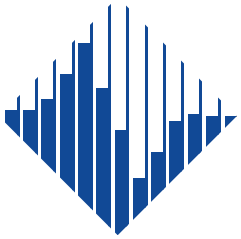


PACIFIC EARTHQUAKE ENGINEERING
RESEARCH CENTER

Long-Term Monitoring of Bridge Settlements using a Vision-Based Embedded System

Henry L. Teng

Khalid M. Mosalam

Department of Civil and Environmental Engineering
University of California

PEER Report No. 2020/26

Pacific Earthquake Engineering Research Center
Headquarters at the University of California, Berkeley
December 2020

PEER 2020/26
December 2020

Long-Term Monitoring of Bridge Settlements using a Vision-Based Embedded System

Henry L. Teng

Khalid M. Mosalam

Department of Civil and Environmental Engineering
University of California, Berkeley

PEER Report No. 2020/26
Pacific Earthquake Engineering Research Center
Headquarters, University of California, Berkeley

December 2020

ABSTRACT

The State of California is highly seismic, capable of generating large-magnitude earthquakes that could cripple the infrastructure of several large cities. Yet the annual maintenance of the State's bridges, such as highway overpasses, is not robust due to budget and staff constraints. Over 1000 bridges were not inspected according to the California Department of Transportation's (Caltrans) 2015 Maintenance Plan. To help engineers monitor infrastructure conditions, presented within is a device recently developed that employs modern sensing, computing, and communication technologies to autonomously measure and remotely report vertical settlements of bridges, such as highway overpasses. Given the limitations of existing measurement devices, we propose a novel vision-based method that employs a camera to take a picture of a projected laser beam. This new device is referred to as the Projected Laser Target Method (PLTM).

This report documents the embedded system design and development of two prototypes. The first prototype implements communication over a local WIFI network using synchronous code to measure distance over time; this PLTM is deployed in a laboratory setting. The second device under study implements communication over a Bluetooth Low Energy system using asynchronous code and communication over 2G cellular networks using synchronous code, with the aim of determining its accuracy in the field. This report evaluates the performance of the field-suitable system in terms of its system reliability, measurement accuracy and precision, power consumption, and its overall system performance.

ACKNOWLEDGMENTS

This work would not have been possible without the financial support of the Pacific Earthquake Engineering Research Center (PEER) and Caltrans through the PEER Lifelines Research Program contract 65A0549 Task Order TO-4: Bridge Vulnerability and Design, Sub –Task 7C: Foundation Settlement Monitoring System. We would like to express our gratitude to Tom Shantz, Senior Research Engineer at Caltrans. We thank the following University of California (UCB) staff for their technical input throughout the course of this project: Amarnath Kasalanati, Associate Director of PEER; Phillip Wong, Senior Development Engineer; Professor Michael Riemer; and Branden Ghena, an Electrical Engineering & Computer Science Ph.D. student. We also thank PEER and the Department of Civil and Environmental Engineering of UCB for providing us with the resources and space to complete this project.

This project greatly benefited from the participation of several members of the project team: we thank Albert Qu for his extensive development and implementation of the computer vision algorithm utilized in our prototype; Wyeth Binder for his design and fabrication of the final prototype enclosures; and Alex Mead for his preliminary work on this project. The authors also appreciate the editorial effort by Ms. Claire Johnson, PEER Report Editor. Any opinions, findings, and recommendations expressed in this material do not necessarily express those of the funding agency, PEER, or the Regents of the University of California.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Background and Previous Work

1.1 STATE-OF-THE-ART AND MOTIVATION

Bridges are a critical component in the day-to-day business of communities. It is a key responsibility for state and federal agencies to maintain the safety of bridges under their jurisdiction. As of 2015, California's Department of Transportation (Caltrans) reported in their 2015 Five-Year Maintenance Plan [Caltrans (2015b)] that they were responsible for the maintenance of 13,100 bridges; this includes highway overpasses. In the same document, key statistics shed light on the need for improvements to the state-of-the-art technologies as it applies to bridge maintenance. Therefore, a brief discussion is presented below will define what is currently considered"state-of-the-art" bridge maintenance.

1.1.1 State-of-the-Art in Bridge Maintenance

In order to maintain bridges, knowledge of the real condition of the bridge stock is critical. Caltrans states that bridge stock conditions are identified through regular bridge inspections performed by contractors on-site, as mandated by federal regulations. These federal regulations are enforced by the Federal Highway Administration (FHWA), a subdivision of the U.S. Department of Transportation. The FHWA began implementation of its National Bridge Inspection Standards in 1971 [Washer (2011)] following the cataclysmic collapse of the Silver Bridge in Point Pleasant, West Virginia in 1967 [O'Connor (2015)].

As laudable as this effort was, these standards suffer from many shortcomings in implementation and enforcement, and in fact were not even evaluated for their efficacy until 1998, when the NDE (Nondestrutive Evaluation) Validation Center was established by the FHWA. A resulting study on the reliability of visual inspections on highway bridges by the NDE Validation Center [Washer (2011)] uncovered several issues of concern:

- Professional engineers are typically not present on site for bridge inspections;
- In-depth inspections are unlikely to correctly identify many of the specific types of defects for which this type of inspection is prescribed;
- Significant variability in the condition rating given to bridges depending on the thoroughness of the investigation;and
- Few inspection teams perform an in-depth level inspection of bridge decks as part of their Routine Inspection.

Ultimately, these findings all stem from and human bias. With this in mind, Caltrans' Five-Year Maintenance Plan states that by 2015 there was a backlog of over 1000 bridges that had not been inspected by the State, with an estimated $450 million required just to maintain that backlog through the year 2020. State agencies like Caltrans currently ensure their bridges are operating safely through a maintenance program. Another strategy for ensuring safe operation of bridges is through design measures.

1.1.2 State-of-the-Art in Bridge Design

An elevation of a prototypical bridge is illustrated in Caltrans' *Design Manual for Bridges*. [Caltrans (2015a)].

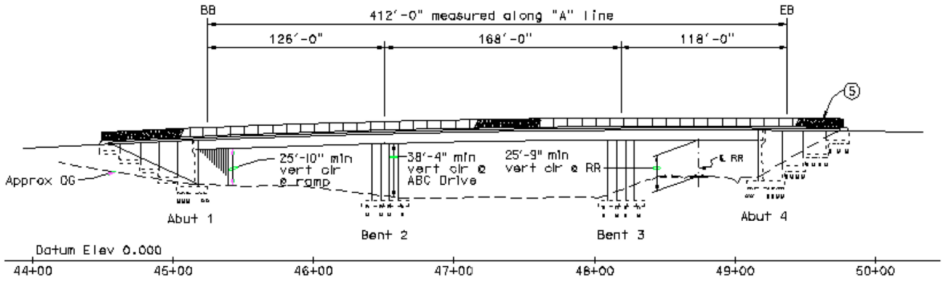


Figure 1.1: Elevation view of example bridge [Caltrans (2015a)]

.

The typical bridge design shown in Figure 1.1 must consider various types of anticipated loads. A key type of loading is that caused by differential settlement, in which one "bent" of the bridge may settle into the ground at differing depths following the construction of the bridge and its foundation, and also during normal bridge operation. To allow for this sort of loading, whose magnitude and direction can be estimated by geotechnical methods (which are outside the scope of this study), the *Design Manual* requires that bridge foundations must allow for a pre-defined differential settlement of 1–2 in. To account for such a magnitude of vertical settlement, sufficiently large safety factors must be applied to the design forces or displacements that are used in the design calculations to supply enough design capacity to meet anticipated loading demands. Enlarging the design factors to account for estimated settlements can be expensive. In the same *Design Manual*, engineers are advised to consider the cost of enforcing settlement limits. If foundation costs are unacceptable, it allows for larger settlements but requires more advanced settlement analyses. The motivation for this project is to have Caltrans reduce design factors and accept less conservative foundation designs that would be less expensive overall to build but would still provide an adequate level of safety.

Allowing for *occasional settlement* would be a fairly reasonable approach in managing the inevitable settling of bridges post-construction. Caltrans' maintenance plans recognized that the cost-benefit ratio for bridge maintenance is 12:1, i.e., minor damage rehabilitation costs 12 times as much as that of preventative measures [Caltrans (2015b)]. This project is aimed at investigating and providing a meaningful proof-of-concept yet deployable solution for engineers and decision makers to detect early-stage as well as long-term settlements. If Caltrans can detect the settlement

early, it can mitigate any potential damage before it occurs. Currently, Caltrans does not operate an existing settlement monitoring program.

With the context of this project established, we subsequently reviewed selected research literature that already exist in the field of infrastructure monitoring to construct the framework of our solution.

1.2 PREVIOUS WORK

Structural health monitoring, as defined by Lynch and Loh (2006), is a new "paradigm" of rapidly identifying structural damage in an instrumented structural system, which can be classified in terms of local or global damage detection methods. Whereas global methods employ numerical models that intake global characteristics of a structure (such as modal frequencies) that indicate possible damage, local methods seek to detect damage by screening structural systems at the individual element scale. In the context of this project, our scope covers only local-based damage detection and is aimed at measuring the relative settlement between two points on a bridge, and reporting that settlement to decision makers so that a conclusion can be made concerning the maintenance state of that bridge.

1.2.1 Traditional Sensors

Quantifying structural responses (be it local or global) is a critical step for structural health monitoring, and common responses that are employed in structural health monitoring are often acceleration, strain, displacement, and tilt [Khuc and Catbas (2017)]. This project focused on measuring the settlement of bridge foundations, i.e., displacement, a not-so-trivial task.

(a) Linear Wire Potentiometer (b) LVDT (c) Strain Gage (d) Dial Gage

Figure 1.2: Contact-type traditional sensors.

Traditionally, engineers have employed displacement sensors such as Linear Variable Differential Transformers (LVDTs), strain gauges (followed by integrating strain readings), wire potentiometers, and dial gauges to collect displacement responses [Anderson and Thorarinsson (2009); Khuc and Catbas (2017)]; see Figure 1.2. Although perfectly adequate under laboratory conditions, these contact-type sensors have critical shortcomings for deployment in the field. These classical sensors require stationary platforms near the measurement points to mount the sensors and are limited in their measurement range [Anderson and Thorarinsson (2009); Khuc and Catbas (2017)]. To overcome some of these limitations, non-contact type sensors have been developed in recent years, including terrestrial laser scanning [Park and Lee (2007)], fiber optical sensors [Lienhart and Brunner (2003)], smartphone sensors [Ozer et al. (2017)], global position system methods

[Moschas and Stiros (2011); Yunus et al. (2018); Im et al. (2013)], accelerometers [Arias-Lara and la Colina (2018); Park et al. (2005)], and vision-based systems [Myung et al. (2011); Jeon et al. (2011); McCallen et al. (2017); Wahbeh (2003); Olaszek (1999); Fukuda et al. (2010); Lee and Shinozuka (2006); Feng and Feng (2015); Feng and Feng (2016); Park et al. (2015); Abdelbarr et al. (2017); Khuc and Catbas (2017); and Jauregui et al. (2003)].

1.2.2 GPS, Laser Scanning, and Fiber Optic Methods

The GPS is a satellite-based radio navigation system whereby specialized satellites orbit the globe and continuously transmit digital radio signals that contain information about the location of the satellites and exact times of transmission. Then, GPS receivers on the ground track these signals sent by the orbiting satellites and by calculating the time difference between the encoded time stamp and the received time to determine distance; see Figure 1.3. When multiple receivers perform this action together, triangulation can ascertain an exact location, including elevation [Yunus et al. (2018); Im et al. (2013)].

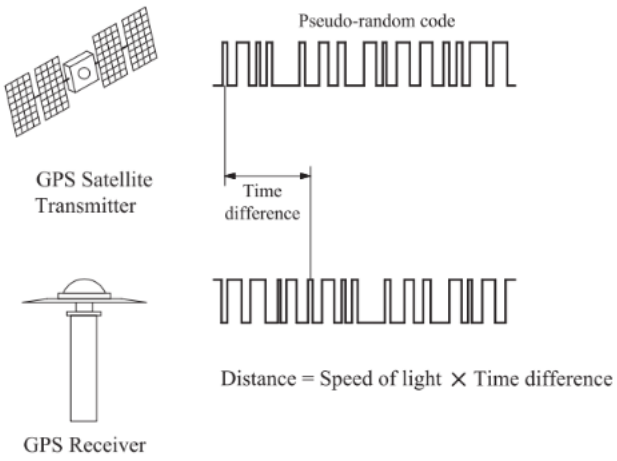


Figure 1.3: Calculation of distance between GPS satellite and receiver [Im et al. (2013)].

The main advantage of GPS is due to its universal presence; as long as a GPS receiver can receive the transmitted signals from orbiting satellites, a distance calculation can be made; however, the latency involved in receiving these signals and interpreting them typically limits sampling rates to below 100 Hz. In addition, GPS receivers also may not receive signals if mounted in enclosed or partially enclosed spaces (such as underneath bridges) [Yunus et al. (2018); Im et al. (2013)]. Nonetheless, controlled laboratory experiments have resulted in measurements down to 3-mm accuracy in the vertical direction [Im et al. (2013)]. Unfortunately, for the purpose of early-onset settlement estimates, sub-millimeter accuracy is desired.

Another new non-contact method that hinges on a time-of-flight calculation is terrestrial laser scanning (TLS). Here, a three-dimensional (3D) coordinate is extracted by measuring the time it takes for a laser pulse to travel from the emitter to an object and then return. Then, a coordinate transformation must be made to convert the coordinate from the local TLS axes to the global structural axes; see Figure 1.4. This method, while able to achieve sub-millimeter accuracy in

laboratory experiments, is hampered by the expense of installing TLS, and the inability to deploy the equipment remotely and run it autonomously [Park and Lee (2007)].

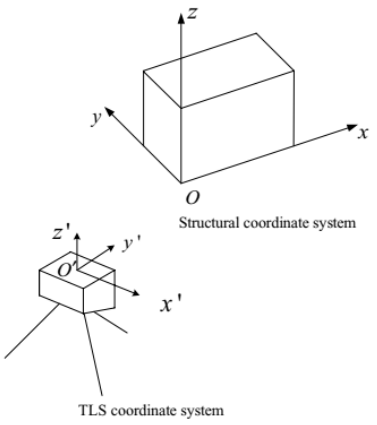


Figure 1.4: TLS coordinate systems [Park and Lee (2007)].

Another innovative technology that has gained traction in structural health monitoring is fiber optic-based sensors. Various studies have investigated the merits of embedding fiber optic cables into spans of bridges and measuring the deformation of those cables, which then gives an estimate of the deformations experienced within the structural element under study. For instance, Lienhart and Brunner (2003) embedded eight fiber optic cables in a newly-built bridge; see the white curves in Figure 1.5.

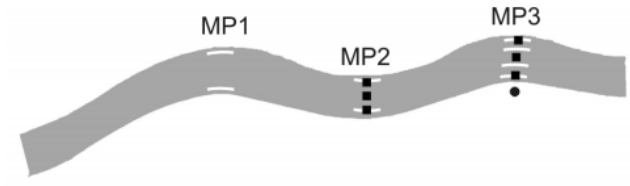


Figure 1.5: Position of sensors in three measurement profiles [Lienhart and Brunner (2003)].

Lienhart and Brunner (2003) used Michelson interferometers to split a beam of light down the length of a fiber optic cable. After recombining them, the team calculated the differences in travel time between the two beams of light. This was subsequently related to travel distance, which reflected any asymmetric deformations in the cable since its installation.

Wang et al.(2015) employed a large deployment of fiber optic cables to examine long-term settlement along sections of the Chinese high-speed railway. By using Fiber Bragg grating sensors, they measured relative settlements between foundations along spans of the elevated railway [Wang et al. (2015)]. as illustrated in Figure 1.7, the Fiber Bragg grating sensor is based on the concept that light traveling between media of different refractive indices may reflect *and* refract differently depending on the media interface. Deformation can be calculated by studying which wavelengths are reflected or lost through a section of fiber optic cable; see Figure 1.8 [Wang et al. (2015)].

Although both methods were able to obtain sub-millimeter precision in their deformation estimates, significant drawbacks in implementing fiber optics for monitoring purposes make it

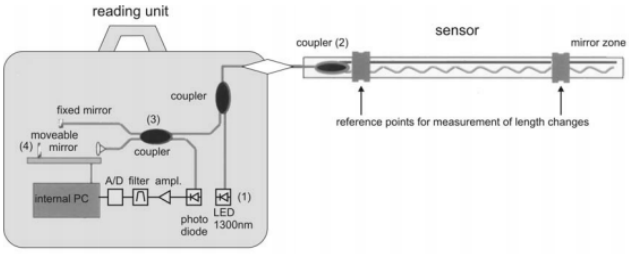


Figure 1.6: Michelson interferometers used by Park and Lee (2007).

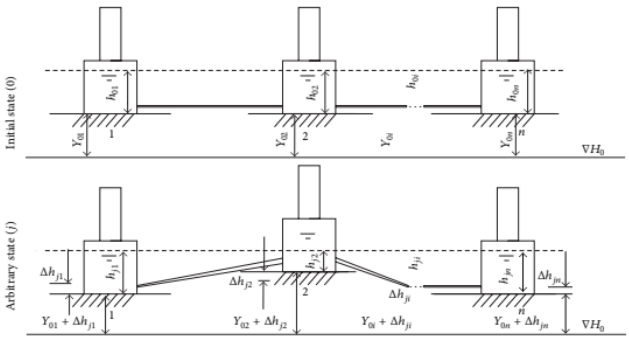


Figure 1.7: Schematic of idealized settlements of railway foundations [Wang et al. (2015)].

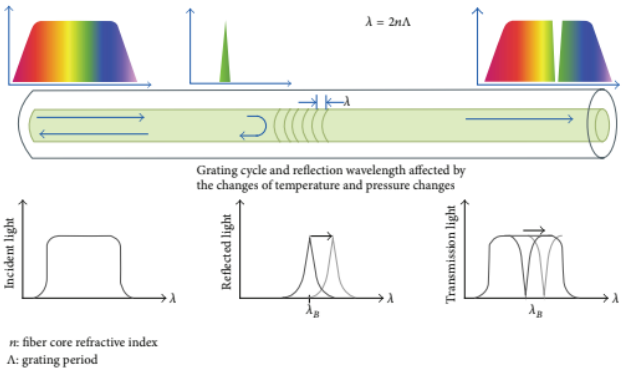


Figure 1.8: Fiber-Bragg grating sensor [Wang et al. (2015)].

infeasible for long-term deployment . The sensing equipment and fiber optics are expensive and potentially hard to mount and power in remote locations. In addition, the fiber optic cables must be embedded into the structural element for the best results; thus, they would not be appropriate for monitoring the deformaton of existing structures [Wang et al. (2015); Lienhart and Brunner (2003)].

1.2.3 Displacement Estimation from Acceleration Data

Another popular method for determining displacement is by double integrating acceleration data, which are collected using an accelerometer. Indeed, accelerometer records can be obtained without

a reference (unlike an LVDT, for instance), and, depending on the specific sensor, can obtain very high resolution readings. Such accelerometers are often used in dynamic testing of structures due to their ease of installation and relatively low cost and noise (depending on the sensor) [Park et al. (2013)].

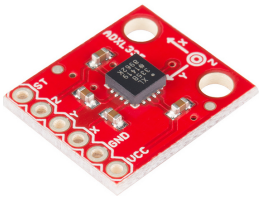


Figure 1.9: Typical MEMS accelerometer.

However, the process of double integrating acceleration to obtain displacement is fraught with error due to the inclusion of low-frequency noise and imprecise initial conditions [Park et al. (2005); Park et al. (2013)]. For instance, when Park et al. (2005) double integrated laboratory-obtained acceleration data with various initial velocities, they achieved vastly different estimated final displacements; see Figure 1.10. Another method used to selectively filter out certain bands of frequencies from the acceleration data involves a high-pass filter to eliminate low-frequency noise [Park et al. (2013)]. This method often leads to the forcing of end displacement values to zero, which is desirable for dynamic analysis but not for static displacement measurements [Arias-Lara and la Colina (2018)]; see Figure 1.11. In Figure 1.11, the top three plots show the effects of various high-pass filters applied on the acceleration time history, whereas the bottom three plots employed either statistical methods or initial condition varying methods [Arias-Lara and la Colina (2018)].

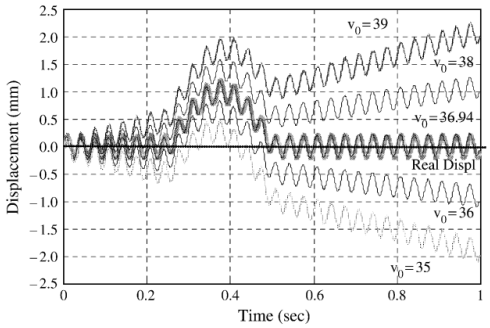


Figure 1.10: Effects of varying initial velocity [Park et al. (2005)].

1.2.4 Vision-Based Methods

As discussed above, various issues have made such technology inappropriate for the goals of the program discussed herein, including the necessity of reference points for contact-based sensors, the need to embed fiber optics into structures when they are built, and the restrictions for mounting GPS receivers with unobstructed paths for communication with satellites. In addition, overall high equipment costs and difficulties in mounting are additional issues that make such technologies

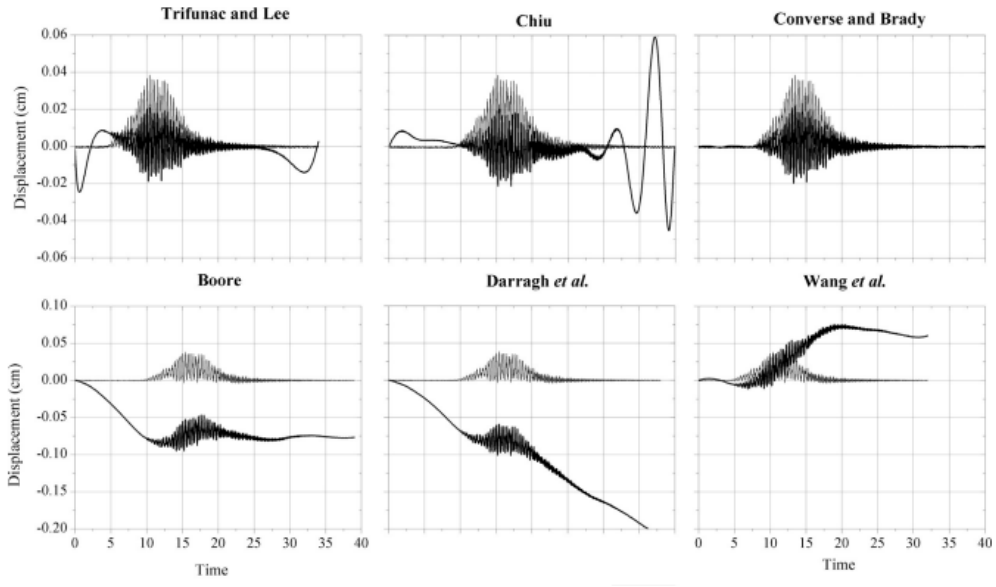


Figure 1.11: Effects of high-pass filter [Arias-Lara and la Colina (2018)].

poor candidates for long-term deployment for monitoring settlements. Recent work in vision-based methods have addressed many of these issues.

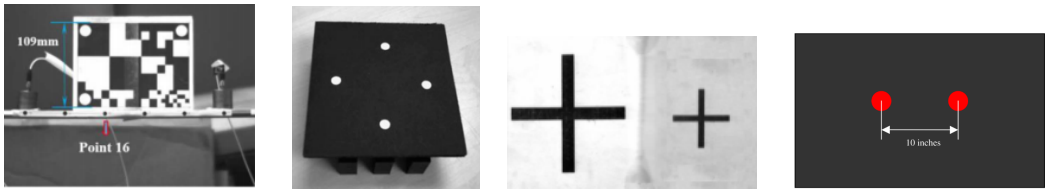
A brief explanation of the high-level goals of this project will put into context why vision-based methods of displacement measurement were considered. Early detection of bridge settlement requires taking only static measurements without regard for the dynamics of the bridge. These dynamics are more indicative of the global response of a bridge (e.g., a frequency shift due to extensive cracking of the deck) as opposed to a local response (e.g., one bent of a highway overpass has sunk 0.5 mm relative to another). Ultimately, all vision-based methods require three critical components to measure the displacement of a point or region of interest: a camera or optical device (hence, vision), something to take a picture of and keep track of over time (hence, feature), and a computational unit to process the photos for the desired features (hence, processor). We will first examine two subsets of these methods: target-based and non-target-based.

Target-Based Methods

Target-based vision methods, which keep track of a target over time, have been investigated extensively in the literature. Many of these studies employed a physical target, often with a pattern. This pattern can be merely an arrangement of circular dots, either printed or painted [Fukuda et al. (2010); Lee and Shinozuka (2006); Park et al. (2015); Jauregui et al. (2003)] on the target surface, or even illuminated with LEDs per Wahbeh et al. (2003). These patterns can also be rectangular, per Abdelbarr et al. (2017), Feng and Feng (2015) and Feng and Feng (2016), or even in the shape of a cross hair per Olaszek (1999). A few of these are shown in Figure 1.12.

These vision-based methods often employ some sort of a motion tracking scheme whereby the camera/camcorder records a sequence of images focused on the target of interest. The initial frame or image is analyzed to extract a coordinate of a certain feature; in subsequent frames or

images, the same feature is located and its location is tracked through time. This initial coordinate acts as a reference frame. In some studies, like those done by Feng and Feng (2015), this algorithm of tracking a certain feature in a sequence of images is called template matching, making motion-tracking possible; see Figure 1.13. Note that "ROI" represents the feature that is being tracked, or a "region of interest"



Rectangular, Feng and Feng | Dots, Fukada et al. | Crosshair, Olaszek | Dots (LED), Wahbeh et al.

Figure 1.12: Example templates for vision-based methods.

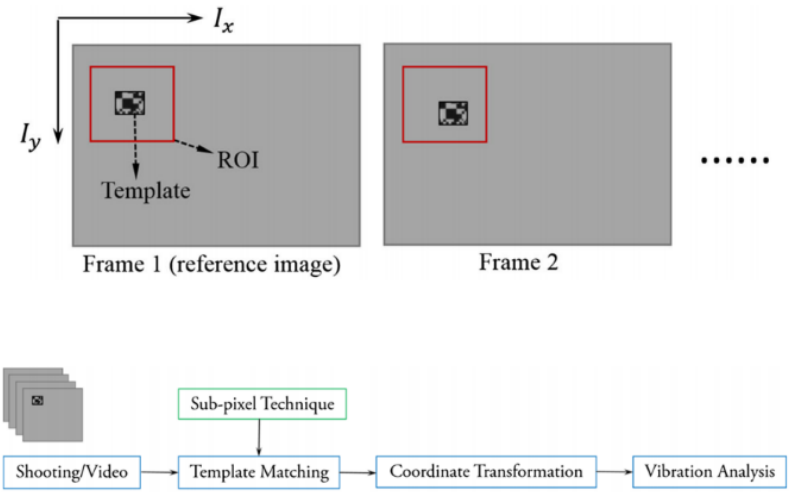
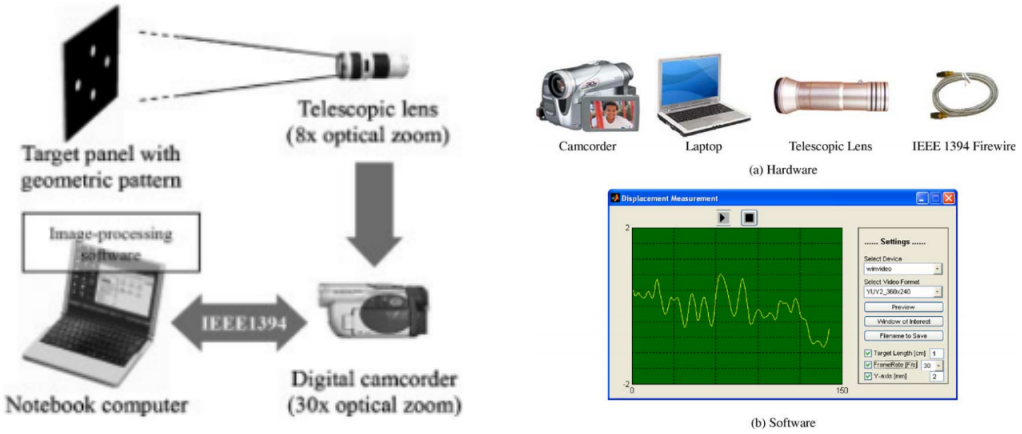


Figure 1.13: Procedure for implementing motion tracking [Feng and Feng (2015)].

By varying the rate of image capture, this motion-capture method can used for either dynamic or static measurements. In the case of the majority of the studies reviewed in this report, dynamic measurements were taken, thus requiring a high-frame capture rate; see Wahbeh (2003), Olaszek (1999), Fukuda et al. (2010), Lee and Shinozuka (2006), Feng and Feng (2015), Feng and Feng (2016), Park et al. (2015), Abdelbarr et al. (2017), and Khuc and Catbas (2017). Due to hardware limits and Nyquist's theory, this limited the range of dynamic response that these systems could capture. For instance, Fukuda et al. (2010) could only achieve reliable readings up to 8 Hz, and Lee and Shinozuka (2006) could only achieve reliable readings below 3 Hz. This of course constrains the use of vision-based methods to low-frequency structures, which includes bridges.

Many of these studies had equipment setups where a camera was mounted a distance away from the target, and the images were either recorded onto local storage [such as in Wahbeh (2003)], or onto a computer for storage and processing [Fukuda et al. (2010)]. In the former case, the data acquisition and data processing stages are separated, with the investigator required to process and analyze the data off-site after the measurement period. In the latter, if the computing unit not only

acquires the images but also processes them, some semblance of real-time measurements can be attained. Various studies, such as those by Lee and Shinozuka (2006) and Fukuda et al. (2010), used laptops and camcorders to acquire images, and telescopic lenses to zoom onto a particular target from afar; see Figure 1.14.



(a) [Fukuda et al. (2010)] (b) [Lee and Shinozuka (2006)]

Figure 1.14: Experimental setup for real-time measurements using target based vision methods.

Although considered"cost-effective" by Fukuda et al. (2010) per the setups in Figure 1.14, this is no longer true due to advancements in micro-controllers and cameras. In addition, these immobile setups are much more suitable for laboratory experiments; indeed, they were only tested in the laboratory or in the field for short duration of times with no long-term deployment.

Studies by Olaszek (1999) and Wahbeh (2003) have provided great insight that benefited the development of our project. We therefore discuss these two projects in brief. Olaszek (1999) investigated the dynamic characteristics of bridges by mounting pairs of cross-hair targets onto the under-structure of the bridges. He developed a specially manufactured optical device to concurrently zoom in and capture images of the two far-away points, utilizing one as a reference point to the other; see Figure 1.15(a). That way, if the camera unit were to be perturbed (perhaps by wind), the data from the reference image could be used to correct for any camera translation. The image analysis algorithm employed cross-edge detection in which edges of the cross hairs of the images were determined [see Figure 1.15(b)], from which the centroid could be determined by regression. This photo-grammetric approach with edge-detection image analysis achieved submillimeter accuracy at long ranges (up to 100 ft) but required the careful consideration of external factors, with the most important factor being lighting.

Wahbeh (2003) used a camcorder with digital zoom that could shoot images of large black targets (28 in. tall $\times$ 32 in. wide), that also were mounted underneath a bridge; see Figure 1.16(a). These targets had two dots of known distance from each other and were lit up via an LED. A sophisticated signal processing technique employing Gaussian regression curve fitting was applied to find the center of this high-intensity spot in the images [see Figure 1.16(b)], which required off-line (and not real-time) processing but achieved sub-millimeter precision. The lack of a communication pathway for data to be transmitted from the camera unit to the computational unit,

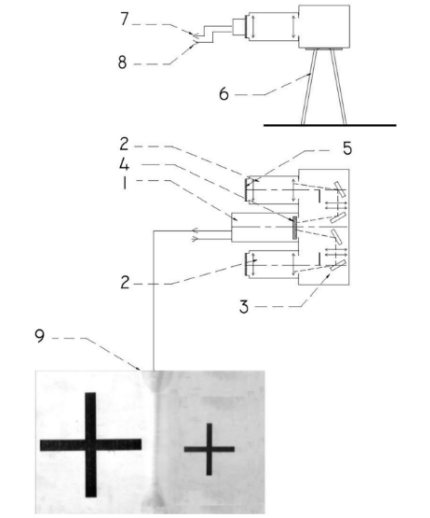
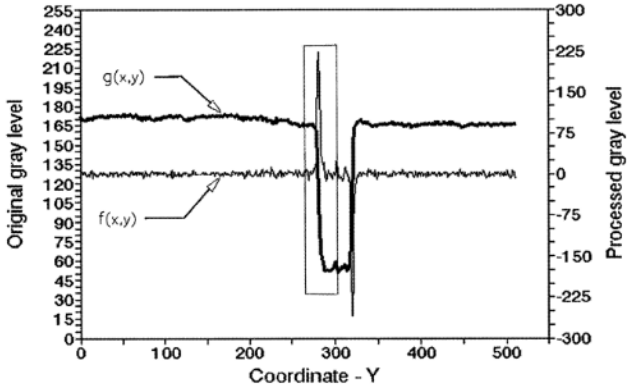


Fig. 2. Schema of two-channels system optical set: 1, camera; 2, telephotolens ($f = 500$); 3, mirror and lens system; 4, CCD camera matrix; 5, optical filters; 6, stand; 7, video output (to the computer); 8, power; 9, image of two patterns observed using the double lens system.

(a) Dual-Camera Unit



(b) Edge Detection Method

Figure 1.15: Camera equipment and image analysis methods [Olaszek (1999)].

combined with the prohibitively large targets, ultimately made this infeasible to deploy; however, it provided good insight into what was required to make a system convenient to deploy in the field on a semi-permanent basis.

(a) LED Targets (Mounted)

(b) Raw Image of LED Target

Figure 1.16: Target Mounting and Target Images [Wahbeh (2003)].

Non-Target-Based Methods

To do away with exacting installation of targets, some studies have explored non-target vision methods of displacement measurement. A study by Khuc and Catbas (2017) implemented an advanced image processing algorithm that applied Gaussian filters at different scales. From the differences between various Gaussian filtered versions of the same images, local extrema were

identified as points of interest that are then matched between adjacent photos in a sequence of photo; see Figure 1.17.

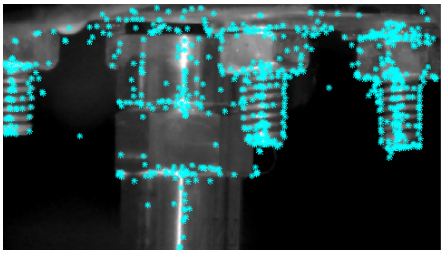


Figure 1.17: Key feature points extracted from images [Khuc and Catbas (2017)].

At this point, the motion-tracking scheme matches the general procedure mentioned in the previous section on target-based methods, with the consecutive position of points in a sequence of images being tracked and subtracted from the previous through time to obtain a displacement history; see Figure 1.18. Khuc and Catbas (2017) acknowledges that although their method could be used to measure both static and dynamic displacements, they did not consider long-term deployment of their technology on an actual site beyond taking proof-of-concept measurements. Ultimately, this device obtained sub-millimeter precision in laboratory settings, but it was also hindered by ambient lighting conditions when deployed outside, perhaps even more so than target-based methods due to their reliance on quality images of distinctive backgrounds for feature detection.

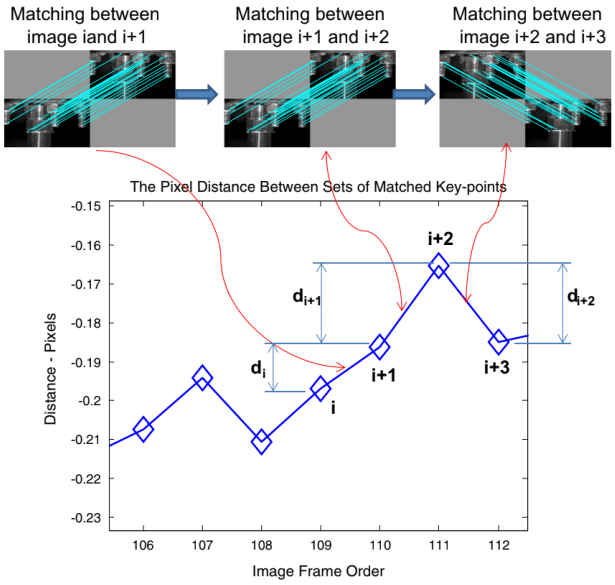


Figure 1.18: Motion tracking procedure [Khuc and Catbas (2017)].

Structured Light

The success of these target and non-target based vision methods are dependent on environmental conditions. Ambient lighting conditions could successively cause feature detection, making motion

tracking impossible to perform. For instance, Jauregui et al. (2003) had to use construction lights to illuminate three targets mounted underneath an overpass in order for their targets to be identified successfully from 60 ft away. To side-step this limitation, one team, Myung et al. (2011), employed a method called structural light. Structural light employs two modules, each comprised of a canvas screen, one camera, and lasers; see Figure 1.19. Each module was mounted a distance across of each other. This distance was only limited by the range of the lasers. The cameras were aimed at their own canvas (not across the spanned distance), and the number of lasers pointed in each direction could be chosen to obtain enough information to reconstruct the 6-DOFs pose of one module relative to the other.

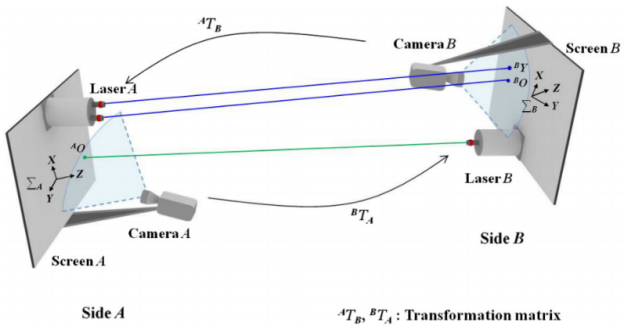


Figure 1.19: One unit of a paired structural light system [Myung et al. (2011)]

The kinematic equations that were derived were extensive, requiring specific geometries of the initial mountings of cameras and lasers to be precise and known; however, once derived, these equations are solved easily. As implemented by Myung et al. (2011), they can be incorporated into a Kalman filter to account for any imprecision in mounting or camera noise. Ultimately, this system was able to achieve sub-millimeter precision, but only in a laboratory setting as it was not deployed on an actual structure. Myung et al. (2011) envisioned a series deployment where the canvas of one module would serve as the canvas of another, forming a chain of paired units, as shown in Figure 1.20.

Figure 1.20: Proposed deployment of paired structural light, [Myung et al. (2011)].

A subsequent paper by Jeon et al. (2011) identified some limitations of the system developed in Myung et al. (2011). That system, while able to estimate 6 DOFs of displacement (translational and rotational), is limited by the canvas size. The cameras are always aimed at the canvases, so the laser points from one module must land upon the canvas of another. Simple perturbations of a

laser emitter on one end could lead to large translations on the receiving canvas. For instance, if two modules were distanced 100 m from each other, with a 30 cm $\times$ 30 cm block of canvas, the screen would only capture for $\pm 0.086^\circ$ in rotation of the laser emitter due to simple trigonometry [Jeon et al. (2011)]. To account for this, Jeon et al. (2011) employed servo motors to control the rotation of the laser emitters, the feedback of which would be fed into the Kalman filter employed in Myung et al. (2011).

Laser-Based Optical Sensor

Vision-based systems need not employ a camera. McCallen et al. (2017) developed a displacement measuring system that employed position sensitive detectors (PSD). The PSD system operates on the concept of a photoelectric effect, so when a laser beam shines upon the semiconductor material of the PSD, electrons are prompted to flow and these resulting currents can then be measured. A sensor array was then custom fabricated for the project, in which individual PSD sensors were embedded into a board in a staggered pattern (see Figure 1.21), such that in practice, measurement precision of 1.0 mm was obtained.

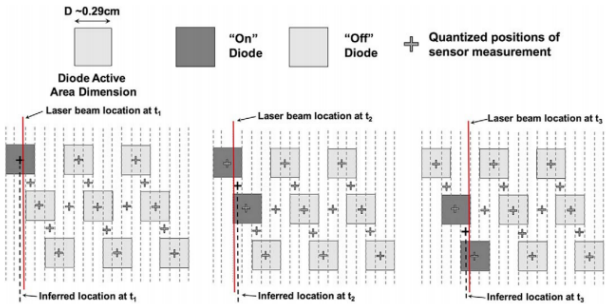


Figure 1.21: PSD sensor array and inferred displacement through time [McCallen et al. (2017)].

This precision is limited by the spacing of the PSD sensors. Although much too imprecise for detecting early-stage settlement, by mounting the sensor array on the floor and shooting a vertical line laser beam onto the board from the ceiling, it was enough to measure interstory drift in a building [McCallen et al. (2017)]; see Figure 1.22.

1.3 PROPOSED SOLUTION

1.3.1 Scope of Work and Specifications

The objective of this study was to develop a system that monitored the settlement of foundations of a bridge with some degree of regularity and report this information wirelessly to decision makers. Such a system would help Caltrans detect potentially damaging foundation settlements in the first year of a bridge's lifespan. To implement such a system, the project employed lasers, optical sensors, and accelerometers. Most importantly, several specifications were defined. To design an effective system, it must meet the following requirements:

1. Accuracy of 1 mm or better;
2. Battery life of 1 year or more;

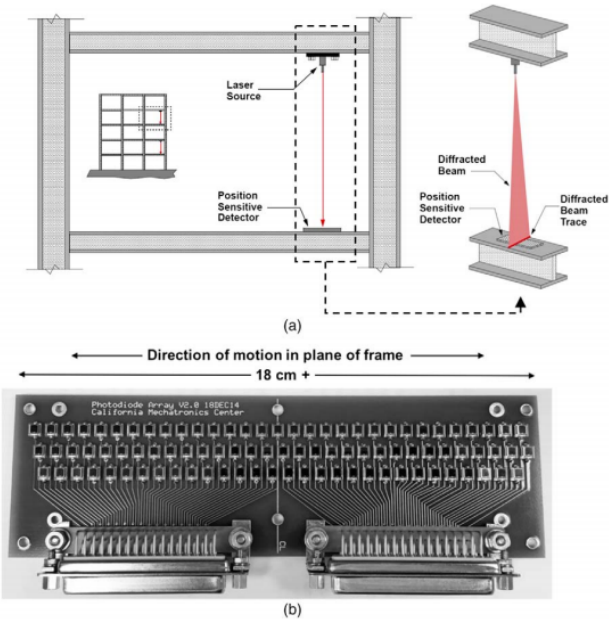


Figure 1.22: (a) Schematic of PSD system; and (b) PSD sensor array [McCallen et al. (2017)].

3. Ability to take 1 measurement per day;
4. Ability for Caltrans to access data remotely;
5. Rugged for field use by Caltrans; and
6. Cost effective and easy to use.

Based on the literature review, further consideration of the use of traditional contact-type sensors was abandoned. The requirement for installation to a physical point of reference is simply too constraining for ease of installation and flexibility in deployment. The use of GPS, laser scanning, and embedded fiber optic devices were also abandoned, primarily due to expense of the equipment and undesirable installation/deployment consequences.

Given the goal of this project in developing a reasonably low-cost, yet precise measurement system, we opted for a vision-based method. Although Feng and Feng (2016) developed a cost-effective system for their target-based vision-based method, their system still required the use of a camcorder with a specific telescopic lens to zoom onto a far-away target. The system also required a PC to read in data from the camcorder in real-time [Feng and Feng (2016)].

We believed this interpretation of "cost-effective" to be outdated and were certain that we could develop an embedded system using commercially available micro-controllers to interface with a sensor suite comprising easily obtainable laser diodes, image sensors, and accelerometers. We felt confident that we could design a reliable state machine for the various micro-controllers chosen for our prototype, implement that design through multi-threaded code, and design a robust image processing algorithm to estimate the position from obtained images. With these objectives in mind, we proposed a key design departure from the typical aforementioned vision-based systems, both target or non-target based.

1.3.2 Projected Laser Target Method

A primary limitation acknowledged by Olaszek (1999) and Khuc and Catbas (2017) was the sensitivity of vision-based methods to environmental conditions. Vision-based systems inherently rely on the extraction of information and features from images, which stresses the importance of quality images. This is no easy task, and Olaszek (1999) even mentioned that the mere direction of the sunlight complicated his image processing and produced errors in his measurements. To that end, we adopted a system that shelters the image sensor in a far more controllable environment; see Figure 1.23.

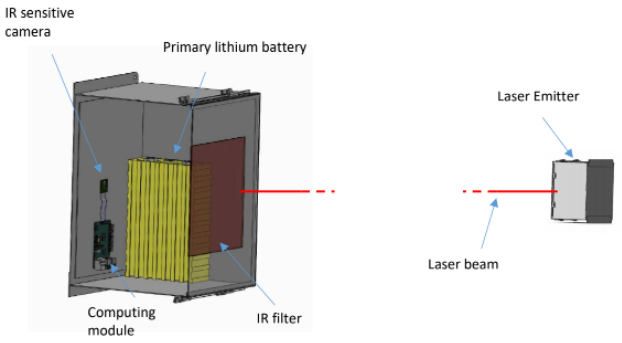


Figure 1.23: Schematic of proposed system.

This proposed design placed the image sensor into an enclosure that blocked out all outside light short of the opening located directly in front of it. This opening holds an infrared filter (a piece of red-tinted acrylic), which blocks out all ambient light except for red light. The laser emitter was a red laser corresponding with the red IR filter that shines its beam onto the IR filter from afar. The beam is visible from within the chamber by the light sensor, which takes a picture of the interior face of the acrylic panel.

This design essentially takes the fixed target used by several teams—such as Feng and Feng (2015), Wahbeh (2003), Wahbeh (2003)—to serve as a point of measurement and breaks it into two components: the canvas and pattern, much like the structured light approach per Myung et al. (2011). The canvas, represented by the IR panel, is brought to within a foot of the image sensor to ensure that quality images of the canvas can be taken without an expensive zoom lens. Much like in Myung et al. (2011), the pattern is not painted directly on the canvas but is projected onto it by an external laser emitter mounted a distance away from the canvas; see Figure 1.23. Our system departs from Myung's system by using an opaque (except for the red light) acrylic panel instead of a solid white canvas. Instead of extracting position information through an affine transformation of coordinates based on pictures taken of laser points projected onto Myung's white canvases, our solution borrows from the method used by McCallen et al. (2017) who employed a discrete array of light-sensitive diodes. Instead of measuring how many intervals of diodes are traversed by a line laser, as was done in McCallen et al. (2017), our solution measured how many pixels were traversed by a cross-hair laser. We were then be able to employ edge-detection computer vision methods, as employed by Olaszek (1999), to determine the centroid of the cross-hair laser beam.

The advantages of this setup lie in the design of the camera module, in which the controlled camera chamber freed us from using expensive cameras and zoom lenses to take high-quality

photos of far-away targets. The limitations of our setup were as follows: (1) the laser emitter must be of adequate strength and focus in its output such that the beam is of adequate intensity and clarity when imparted on the acrylic panel; and (2) while the acrylic panel may filter aberrant external light, thereby reducing environmental effects on the captured image, the panel does so by "averaging" the exterior light. The effects of light penetrating this acrylic will be studied further, as there are most certainly limitations on the conditions of light that may result in unacceptable captured images.

We adopted a similar motion-tracking scheme to that used in many previous vision-based systems; see Khuc and Catbas (2017) in Figure 1.18. Instead of measuring differences in positions of pictures sampled at high frequency (for purposes of dynamic displacement tracking), we tracked the motion of the pattern across pictures taken once or twice a day for the purpose of long-term monitoring of static displacements.

To implement this tracking procedure, various measurement tasks must be performed in sequence. To perform these tasks, communications must be established between the camera unit and the laser unit. In addition, another communication line must be established to carry measured data to an off-site server. We proposed an embedded system that will reliably perform the aforementioned duties.

1.3.3 Embedded System Design

Vision-based displacement measurement systems are known for their flexibility in installation (compared to contact-type sensors) and can produce readings accurate enough for static measurements as well as for dynamic studies of structures. Many previous studies, such as those by Feng and Feng (2016), were limited in scope to laboratory experiments with no field deployment. Studies that did deploy their systems onto actual real-world structures did so with much difficulty, such as with the large panels and expensive equipment of Wahbeh (2003) and Olaszek (1999), respectively. Finally, these studies did not address the application of their systems for long-term monitoring purposes.

Some studies have endeavored to produce systems that are field-worthy. For instance, Washer employed electrolytic tilt sensors to measure both short-term and long-term movements of bridges [Washer (2010)]. While these sensors could not directly measure settlement, the system that Washer developed leveraged the pre-meditated mounting arrangement of tilt sensors to estimate an over all vertical settlement. The most important takeaway from Washer's study was the one-plus year deployment of his system on a test bridge in Rome, New York, during which the system collected data without interruption, with reasonable power consumption ($<$70 W) supplied by a power-line and back-up battery, along with remote data accessibility through a web server [Washer (2010)]. Unfortunately, Washer's report discusses little of the electronic design required to obtain these performance metrics, as seen in a diagram of the system in Figure 1.24.

To best construct a framework for our embedded system, we adopted the hardware design of the wireless sensing units developed by Wang et al. (2007), which included a sensing interface, computational core, and wireless communication, all of which together ensured limited power consumption, long peer-to-peer communication ranges, and local data processing capabilities. Wang's team chose specific hardware components to meet these goals, as detailed in Figure 1.25.

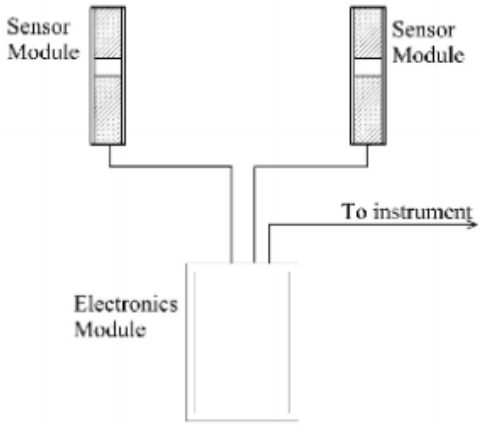


Figure 1.24: Schematic diagram of sensors and electronics modules [Washer (2010)].

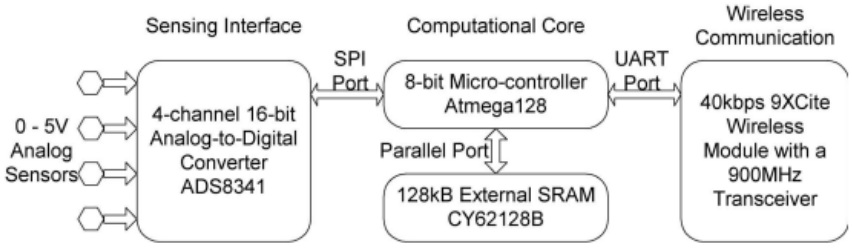


Figure 1.25: Hardware design of the wireless sensing unit [Wang et al. (2007)].

Wang's team aimed to design a wireless monitoring system that supported real-time data acquisition from multiple wireless sensing units. This necessitated the development of multi-threaded software to implement simultaneous data collection, data interrogation, and wireless transmission. In addition, a unique data communication protocol was developed to enable real-time and near-synchronized data acquisition from the multiple sensing units.

The scope of the research reported herein did not require a network of sensing modules that must measure and report time-sensitive data. Our project required taking one or two measurements per day, with no priority for time synchronization of data taken from multiple modules. In other words, if each deployed module wirelessly reports back their daily information in a loosely-defined "timely" manner, we have already made an appreciable improvement in the state-of-the-art of long-term bridge settlement monitoring. Such a flexible scope also accommodated the use of commercially available computing modules that implemented the exact same communication channels and hardware components as detailed in Figure 1.25.

Our project was not novel in pushing the limits of any hardware or software capabilities; that job has been done well by groups such as Wang et al. (2007). What distinguishes our project is that we applied the fundamental embedded system framework used by teams such as Wang et al. (2007) to extend vision-based measurement methods into feasible long-term monitoring solutions for use-cases such as bridges and highway overpasses.

Undeniably, the framework and analysis used in Wang et al. (2007) was inspirational and

useful in organizing and designing an effective embedded system. For instance, because Wang's team approximated the current and voltage consumption of their wireless sensing unit, they were able to specify a reasonable power source for their deployable system. In addition, their design of multi-threaded embedded software for their wireless sensing unit demonstrated that it was possible to perform parallel tasks simultaneously on the same computational core. This was directly applicable to our project. Certain tasks, such a collecting data, manipulating data, or communicating data, could be done concurrently instead of consecutively. To design the sequence, logic, and flow of these various tasks for the sensing unit to perform, it was proven to be most helpful in designing a state machine that could clearly describe the expected behavior of the unit given a current operational task and various inputs. An example of a state machine as designed by Wang et al. (2007) is shown in Figure 1.26.

With these takeaways, we applied many of the same concepts to our project: to write multi-threaded code to implement sensor interfacing, data interrogation, data transmission, and module communication on our chosen hardware. We analyzed the power consumption of our own wireless sensing unit so that we could ensure that our system is deployable for long-term monitoring with minimal maintenance requirements. To ensure that our software operated as intended, we used state machine methods to visualize and analyze our operational states and program flow

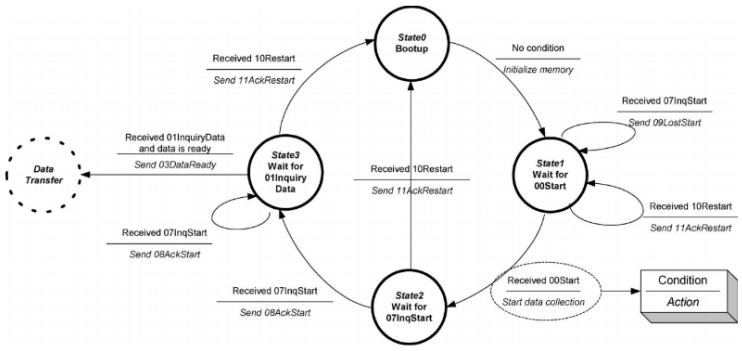


Figure 1.26: State diagram for the wireless sensing unit [Wang et al. (2007)].

1.3.4 Development Stages of the Project

We divided the project timeline into two primary phases; the first phase focused on the development of a laboratory-deployable system and the second phase on the development of the field-deployable system. The division of the project into these two phases was purposed such that in developing the laboratory-deployable system, we would become familiar with key skills that are needed to prototype an embedded system that met Caltrans' needs.

Such skills involve programming with whatever coding language could be compiled onto our chosen hardware to interface with various sensors using the input/output capabilities of our chosen micro-controller, and generally understand how to quickly debug and iterate code versions on the chosen hardware. Moreover, in addition to developing the hardware and software package, time was required to design an adequate experimental setup for calibrating and testing our system in a laboratory setting. To perform such experiments, we also had to learn how to use 3D CAD software and a 3D printer to design and fabricate mounting devices for our computing and sensing

hardware. Note: the skills gained in the first phase were crucial to the timely completion of the second phase of the project.

The second project phase involved the application of skills learned during the first phase to make the final prototype for Caltrans: a field-deployable system. Beyond re-designing certain features of our hardware and software package, this called for designing a robust enclosure that could securely protect all mounted hardware from the elements and ensure the viability of projected laser target method.

A possible mounting scenario is illustrated in Figure 1.27 in which the red dots signify the location where one module out of the paired system will be mounted (either the camera module or the laser module) in the context of an idealized highway overpass with two spans.

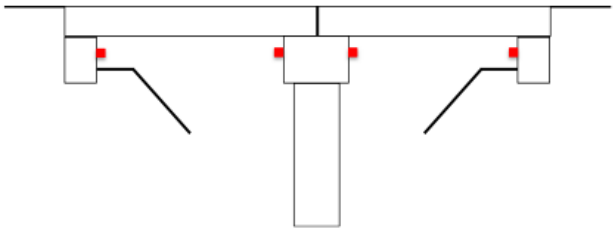


Figure 1.27: Mounting arrangement.

2 Project Phase I: Lab-Deployable System

The first iteration of our system was intended as a beta version to examine features that may be adopted in the final prototype for Caltrans. Ultimately, although this Phase I prototype became more suitable for lab deployment, a similar design workflow was developed that could be repeated for Phase II:

1. Consider the intended user and use case to determine features and key items in the prototype;
2. Specify what hardware components would be used and how they should be integrated together in a system architecture; and
3. Develop an embedded system capable of being deployed in the laid out system according to state machine concepts, including its communication protocols and procedural code.

2.1 SYSTEM FEATURES AND ASSUMPTIONS

Before any development of the prototype could begin, consideration of the possible use cases of our system was necessary to determine what features would be useful for the intended user; an engineer at Caltrans. In addition, various limitations became very obvious at the onset and required making key assumptions to validate our projected laser target approach.

2.1.1 Use Case Considerations for Feature Selection

We envisioned the following use case for the Phase I system: After the device is installed, there should be no need to touch the device for a considerable time—on the scale of months or years. Beyond the accuracy of the reported measurement, the engineer will also require that the system be reliable, i.e., measurements should be taken at regular intervals with little to no latency. The system should automatically take measurements with the assurance that there would be trivial small latency in measurements. Given this blueprint, the first issues to tackle were as follows:

- How would the system know when to take measurements?
- Will the engineer send a command to the device?
- Will the engineer hard-code in a timing frequency?

These questions will be addressed in our ultimate design.

2.1.2 Limitations and Assumptions

Our projected laser method measures one-dimensional (1D) displacement by taking pictures of the interior face of the acrylic screen; therefore, it is blind to tilt effects stemming from both the camera and laser modules. A detailed discussion on the contribution of tilt of either module is included in Appendix A. In short, we assume that neither module will rotate after the initial mounting, but module tilting using digital accelerometers will separately measure and report for each unit.

2.2 SYSTEM ARCHITECTURE

These issues of camera tilt and laser tilt are exterior to the embedded system design, but the measurement tasks and time synchronization items are not. We devised a prototype system architecture that will work along with the assumptions made to limit camera and laser tilts, as illustrated in Figure 2.1.

The camera module and laser module are illustrated abstractly in Figure 1.23. They are broken up into their constituent hardware and are connected by pathways that represent information flow. Consider, for example, the WiFi-Router; the pathway leading to its left leads to the camera module, and the pathway leading to its right leads to the laser module. The pathway leading above it leads to the database, server, and website.

This schematic features all lines that carry data. For instance, the relay circuit component, while ultimately controlling power flow through the system, is controlled using polling data lines between the GPIO ports of the Raspberry Pi and the GPIO ports of the Arduino. We will go into further detail on the chosen hardware and details of the above schematic below.

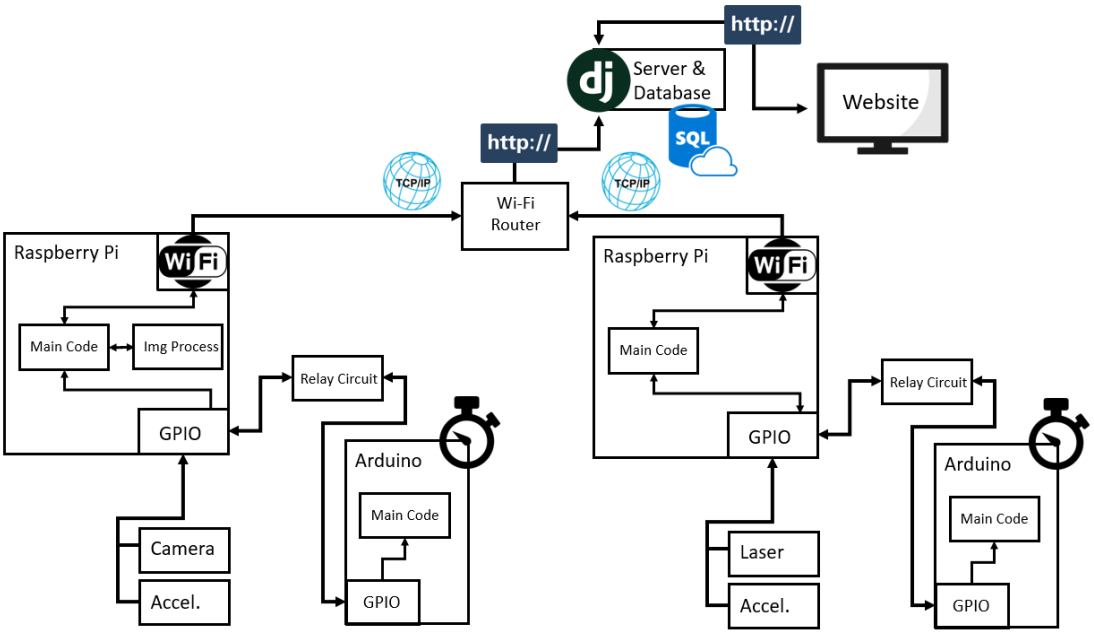


Figure 2.1: Phase I system architecture.

The system architecture can best be explained in terms of the three-component framework of a wireless sensor device, as constructed in Wang et al. (2007), which contains a computational core, a sensing interface, and a communication component. We will explain the proposed system architecture in terms of this framework, which is better illustrated in the context of a generalized structural system in Figure 2.2. Note: we do not have an actuation interface as there is no need to actually perform any action back on the system. We are merely reporting measurements to a remote server. This server has been implemented and will be briefly discussed in this report as it relates to the outputs in Phases I and II.

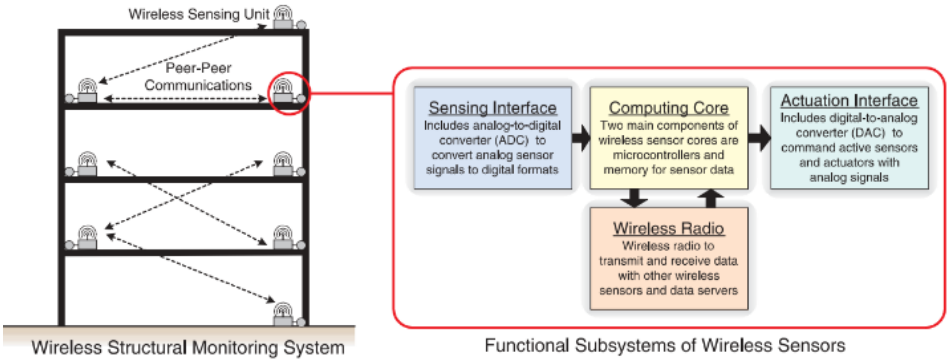


Figure 2.2: Functional elements of a wireless sensor for structural monitoring applications [Lynch and Loh (2006)].

2.2.1 Computational Core

Each module contains two computational units: a Raspberry Pi and an Arduino. These two computing modules were chosen for this project as they both have a large hobbyist following, with immense online documentation generated by the engineers and developers of their respective companies, in addition to a tremendous amount of open-source resources. As stated in Section 1.3.3, the ultimate purpose of this project is not to push the envelope on the efficiency nor effectiveness of the computational core of the wireless sensing module, but rather to extend the functionality of modern-day electronics and computing to make the long-term monitoring of bridges more feasible and deployable. Nonetheless, why we chose these two computational units extends beyond merely their commercial and popular appeal.

The computational core must perform several tasks specific to this phase of the prototype. They are:

1. Interface with various digital sensors;
2. Communicate wirelessly with the other module;
3. Implement measurement tasks in a programmatic manner as defined by a state machine;
4. Implement complex image processing and computer vision algorithms;
5. Perform time synchronization; and
6. Control power delivery to various components in the computational unit.

For clarity, a subset of Figure 2.1 is extracted in Figure 2.3 showing the composition of the computational unit which consists of two boards. The Raspberry Pi will perform the first four tasks enumerated above, and the Arduino will perform the last two tasks enumerated in the above list.

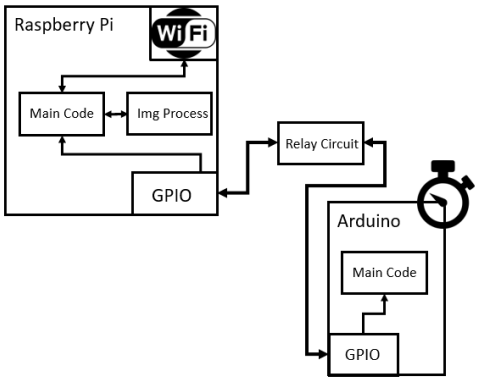


Figure 2.3: Schematic of entire computational unit.

Raspberry Pi

A Raspberry Pi can perform most of the requirements listed above with reasonable power consumption. The Raspberry Pi is a "system on chip" (SoC) that acts as a mini-computer. It does most operations that one would expect from a typical computer. It contains an operating system (OS), which is a derivative of Linux called Raspbian. The various tools available because of the OS include high-level programming with helpful modules such as Python, as well as useful Linux tools through the command line. Because of these benefits, we chose to adopt the Raspberry Pi as our primary computation unit. In addition, the Raspberry Pi has extensive online support and documentation to support this project.

We chose to use the Raspberry Pi 3B+ model because it has built-in Bluetooth Low Energy and Wi-Fi; see Figure 2.4 . For more details on the various peripherals, data capabilities, power consumption, and general discussion on our reasoning behind choosing the Raspberry Pi 3B+, refer to Appendix B.

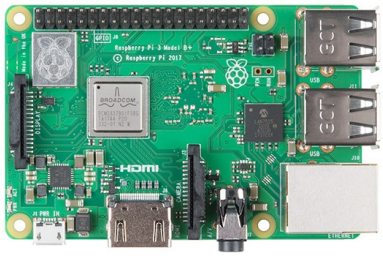


Figure 2.4: Raspberry Pi 3B+ Google Images.

Arduino

As opposed to the Raspberry Pi, which is essentially a computer, the Arduino is much simpler; it is a micro-controller. In fact, a SoC like the Raspberry Pi may actually have a micro-controller as one of its components. That being said, the Arduino, with its simple input-output functionality, is capable of performing the last two duties listed above. It can read in signals on its input pins, make a decision based on those inputs, and output a command. In addition, the Arduino has sufficiently low-power consumption with a very large hobbyist following, much like Raspberry Pi. Therefore, we selected the Arduino as another component of our computational unit. For more information on the Arduino, our selection of a specific model of the Arduino (the Arduino UNO Rev3, Figure 2.5), and the various features of the Arduino board, refer to Appendix C.

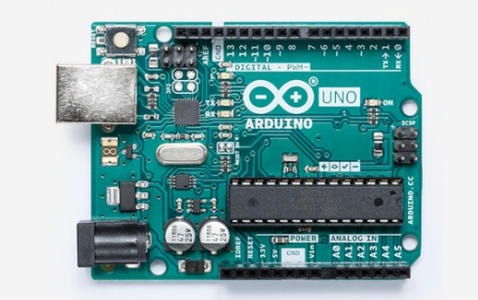


Figure 2.5: Arduino UNO Rev3.

2.2.2 Sensor Interface

We refer back again to the framework of a wireless sensor network of Wang et al. (2007). Having discussed the computational core of our system, we now introduce the sensor interface that it must engage with. As seen in Figure 2.1, each computation core must interface with an accelerometer. The camera module will also engage with an image sensor/camera. The laser module will also engage with a laser diode.

Digital Accelerometer

The chosen accelerometer is an MMA8451 digital accelerometer, which includes the MMA8451Q accelerometer by Freescale Semiconductor installed in a breakout board and distributed by Adafruit. The tri-axial accelerometer can either run at 14-bit or 8-bit resolution, has adjustable full-scale range of up to $\pm$ 8g, and can be interfaced using I2C. With a current consumption of up to 165 μA and input voltage of 1.6 to 3.6 V, the power consumption of the MMA8451Q is essentially trivial to the total system power consumption.

The accelerometer will output how much acceleration it sustains in three Euclidean axes. We are using these accelerometers to measure tilt, i.e., pitch [see Figure 2.6(b)]. To compute tilt from triaxial acceleration readings, we used concepts of vector algebra to obtain the pitch. Given A_x, A_y, A_z, the measured accelerations in the three axes x, y, z, respectively, we can calculate for pitch:

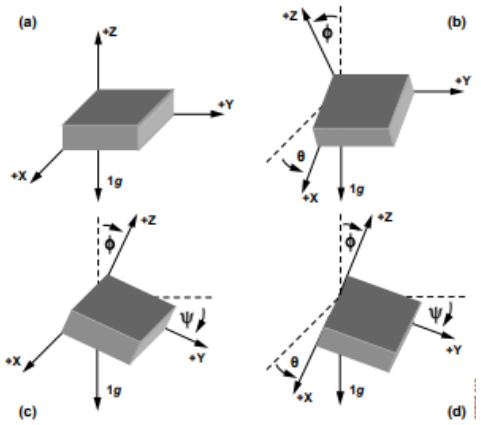


Figure 2.6: Angles for independent inclination sensing [1].

$$\theta = tan^{-1}\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \tag{2.1}$$

From empirical tests (moving the accelerometer into various positions and keeping it constant), it is incapable of measuring tilt to the precision required to correct for module tilt influence on the settlement estimate. This is why we configured the physical setup so that the modules cannot tilt following installation but use an accelerometer to report back a tilt reading of precision of order of magnitude 1^o. Although this configuration is restrictive, monitoring such tilt will help engineers decide on the need of a more sophisticated settlement measurement or assessment, such as sending engineers to the site.

Camera/Image Sensor

The camera module's computational unit must also interface with an image sensor/camera that is compatible with the Raspberry Pi. We chose to use the Raspberry Pi Camera v1, which uses the image sensor OmniVision OV5647. From Raspberry Pi Foundation documentation, the v1 camera has 5 megapixel still resolution, with maximum sensor resolution of 2592 $\times$ 1944 pixels [2]. The sensor's horizontal and vertical field of view, 53.50 ± 0.13^o, 41.41 ± 0.11^o, respectively, will be important when designing the hardware enclosures. In addition, this camera is compatible with the camera port of the Raspberry Pi 3B+, which connects to the camera itself via a ribbon cable (see Figure 2.7), and transfers data over a CSI interface, employing I2C/SPI data protocols [3].

Laser Diode

Finally, our laser module's computational unit must interface with a laser diode. The physics behind the operation of a laser diode are beyond the scope of this report. Of concern herein is the power draw for a laser diode, which is indicative of the laser's effective range. Again, we emphasize that this is a key limitation of the effective range of our proposed projected laser target method. We performed various laboratory tests to explore this limitation. We tested three laser diodes of various power ratings: a 5 mW, 30 mW, and 100 mW laser (see Figure 2.8); laser diodes

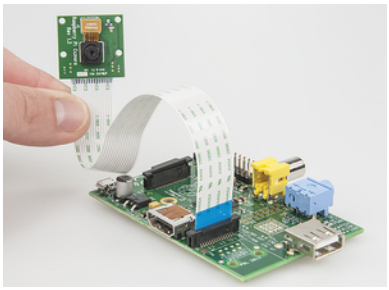


Figure 2.7: Raspberry Pi camera port with ribbon cable.

were provided with lenses to create a cross-hair beam. The power consumption of the chosen laser was a key consideration.

With the sensing interface already specified, we now have a computational unit interfaced with a sensing suite. This is illustrated in Figure 2.9, which is extracted from Figure 2.1 for clarity.

Figure 2.8: 100 mW laser with included lenses and mount.

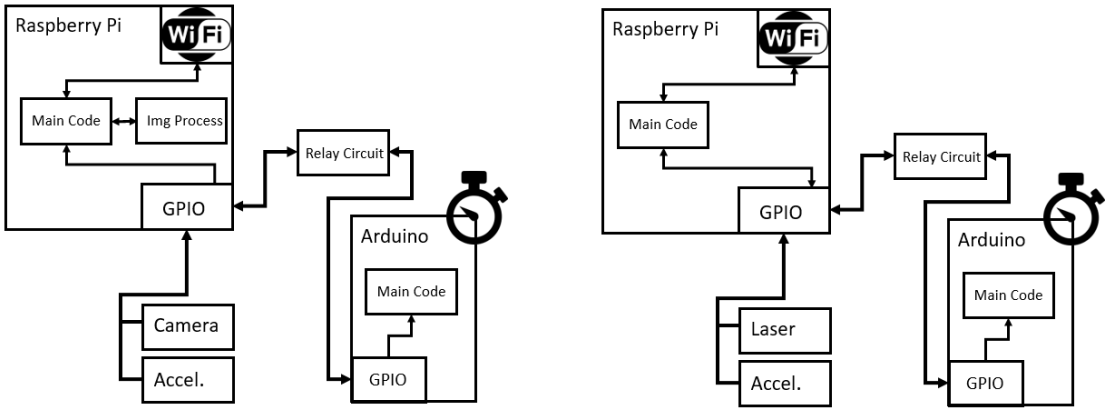


Figure 2.9: Schematic of computational unit with sensing interface.

2.2.3 Communication

We now must consider the last aspect of the Wang framework's wireless sensing device: wireless communication. Our device must be able to send its measurements to the engineer. In addition, our system requires having both a laser and camera module, each of which controls specific sensors; therefore, the modules will have to communicate commands to each other. For instance, the camera module may have to command the laser module to turn on the laser.

When considering wireless communication, common considerations include: how much data you can send, how fast you can send it, and how reliably you can send it [Lee and Seshia (2017); Wang et al. (2007); Park et al. (2013); and Lynch and Loh (2006)]. Wireless sensor nodes send data over radio frequencies; the specific band and technology are then up to the choice of the developer. For instance, a system may employ Bluetooth (short-range radio), Zigbee (also short-range radio), or 802.11 (commonly used for Wireless Local Area Networks, WLAN) technology to send their data [Lynch and Loh (2006)]. A great deal of existing literature and studies exist in the realm of wireless sensor networks (WSN), with emphasis placed on the specific "network." Indeed, these networks can take the shape of a variety of topologies, as illustrated in Figure 2.10.

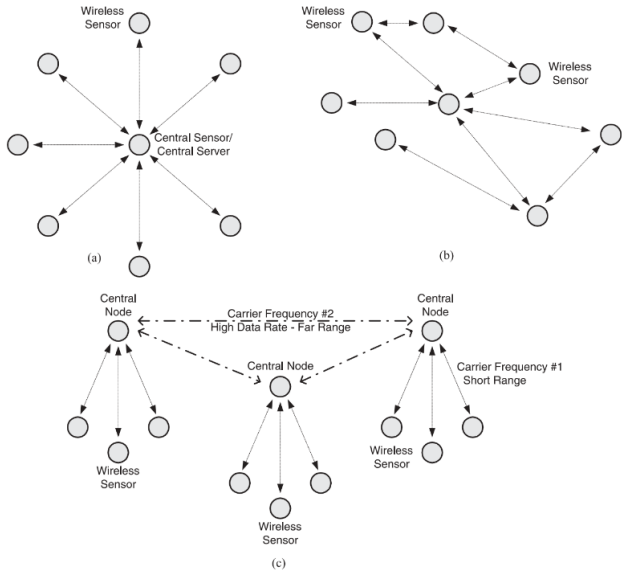


Figure 2.10: Wireless network topologies for WSNs: (a) star; (b) peer-to-peer; and (c) two-tier network topologies [Lynch and Loh (2006)].

These various topologies have their own advantages. For instance, whereas the the star topology may be a very simple network to implement and visualize, the range of the network is limited to the wireless technology's range from wireless sensor to central server. If one wanted to spread various nodes of limited range about a large structure, they could perhaps employ a peer-to-peer network, which would then require more intricate mapping of each node in relation to each other. Investigating the various implementations of WSNs is outside the scope of this study. We were concerned in the Phase I laboratory-deployment level to have a single pair of modules communicate with each other, and then have one of the modules communicate the final measurements to the remote server. Granted, while this may resemble a two-tier network topology, this classification

was not a design choice for this project.

Wireless Router

As our system in Phase I will only be launched in a single room (i.e., a large single-room structural testing facility), the range that our modules would have to communicate across to each other is well within the limits of a WLAN. Typically, the range of a WLAN is limited by the wireless router, which can have ranges of up to 150 ft indoors [4]. We chose to use a spare wireless router that met our prototyping requirements: a Netgear WGR614v7 wireless router. Each module can connect to the local area network established by the wireless router through its 802.11 Wi-Fi chip and find any other device on the network provided they know the other device's IP address. Once connected on the WLAN, devices can then communicate with each other using some communication protocol such as TCP/IP. These details will be covered in a subsequent implementation section.

The communication pathways between the two modules and the modules and the remote server is now established. Our system now resembles the complete schematic in Figure 2.1. The server and website were written and deployed, and can be abstracted outside of the wireless sensor framework, as illustrated in Figure 2.2 by Wang et al. (2007), and Figure 2.1 detailing our proposed system architecture. In other words, we have designed the embedded system nearly independent of the server, database, and website. Our only requirements were that the inputs for the server are output by the wireless device, and that the communication pathway is compatible for both parties per the design of the proposed system. Further details about the server/website can be found in Appendix E.

Figure 2.11: Netgear WGR614v7 wireless router.

2.3 IMPLEMENTATION

With all components specified, the next step was to design a system that integrates all of the separate parts together to perform the required measurement tasks. We decided to employ the state machine concept to better design and visualize the program flow to develop a robust embedded system that will behave in an expected manner and be reliable. A state machine is a model of a system that can be described as having discrete states. These states are the operational conditions of a system at a particular point in time. These states accept certain inputs and will respond with certain outputs and/or actions [Lee and Seshia (2017)].

For instance, consider a thermostat that controls an HVAC system that must either turn on or off the heating. The behavior of this thermostat can be modeled as a state machine; see Figure 2.12. In this figure, the states are defined as follows: $states = \{heating, cooling\}$, with a set-

point temperature between 18 and 22. The machine will intake the temperature as an input at a given state, and output a command of either turning on the heat or turning off the heat. The output and action is associated with a transition from one state to another. Note: if the machine is in the cooling state, the machine will only perform an action if the $temperature \leq 18$. This is reflective of the advantage of using state-machine concepts when designing the behavior of a machine in that one can clearly state the inputs and outputs of a machine with respect to certain states of operation.

Note the notation of the state transition where the left item, the input, and the right item the output are separated by a "/". Another action that can be done in addition to an output is the setting of a shared variable. This can be denoted as a second line underneath the [input] / [output] line, where a variable x can be assigned a value: $x := value$.

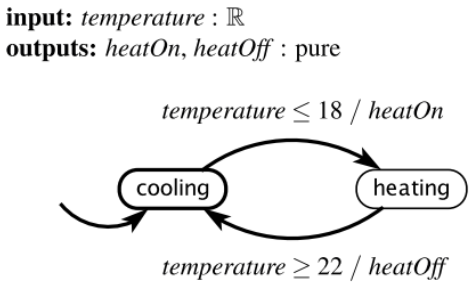


Figure 2.12: State machine model of a thermostat [Lee and Seshia (2017)].

2.3.1 State Machine Design and Implementation

We have described the various duties of the computing components in each of the camera and laser modules in the system in Figure 2.13. These duties will then be implemented more fully in subsequent subsets of the functional block diagram in Figure 2.13.

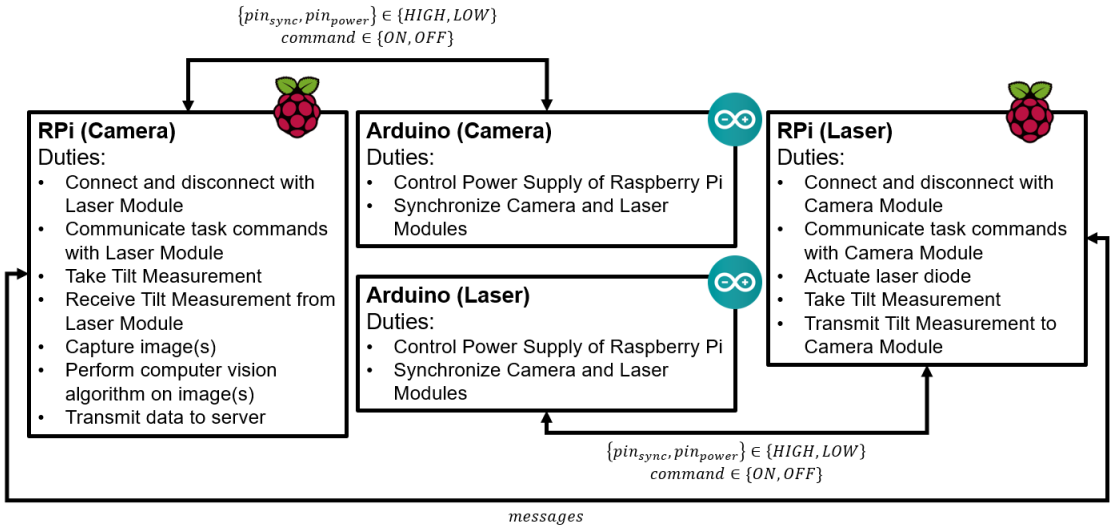


Figure 2.13: System-level functional block diagram and component duties.

One key point worth emphasizing is the extensive list of duties required of the camera module. The camera module, namely the Raspberry Pi in the camera module, must be capable of compiling all of the required data to send to the server, including the extracted centroid value of the captured image ($[p_x, p_y] = [pixel, pixel]$), along with two tilt measurements ($[pitch_{camera}, pitch_{laser}]$). This seemingly arbitrary choice to designate the camera module as the so-called "master" or "central" device, and the laser module as the "slave" or "peripheral" device, is made to establish a directionality in the communication protocol developed for the passing of data and command messages between the camera and laser modules. This communication protocol and the impact of this chosen directionality will be explained later in this section.

The proposed functioning of this system is as follows:

1. Engineer pre-defines settings and hard-codes them into the computing unit. Settings include measurement intervals and synchronization intervals;
2. Technician installs unit into field and boots up the computational unit;
3. Once powered on, the system operates autonomously; and
4. The system will take measurements according to the defined intervals and synchronize the laser and camera modules according to the defined intervals.

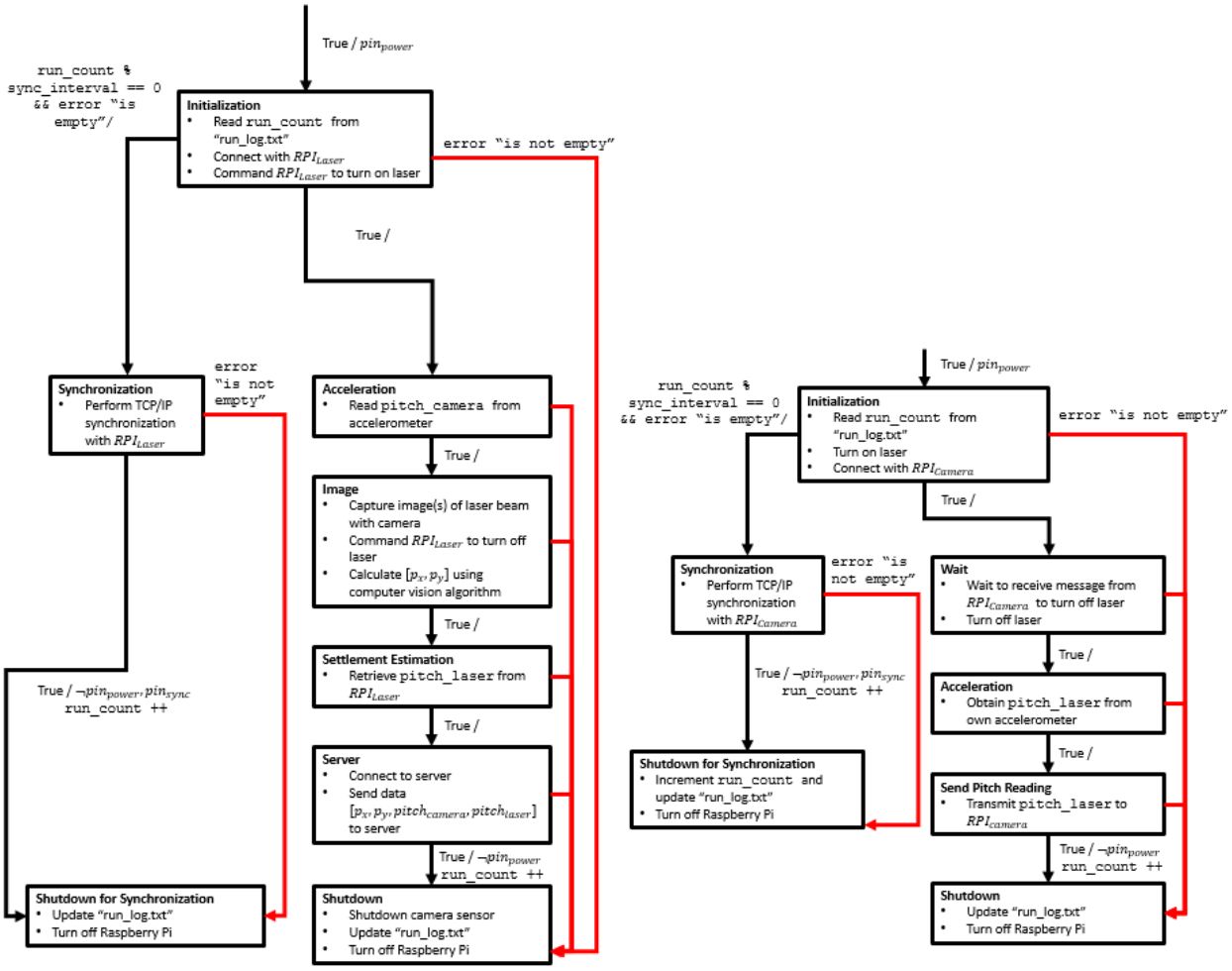
Raspberry Pi Design and Implementation

The design of the Raspberry Pi in both the camera and laser modules is shown in Figure 2.14. These state diagrams illustrate the finite number of states possible per each Raspberry Pi at any one instance of time and define the transitions required to move from one to another. We will explain the program flow through each diagram to illustrate the expected robust operation of each module with respect to each other.

Before narrating the programs of each module, note that there are three points of interest that each state machine addresses:

- Non-fatal handling of unexpected errors;
- Proper synchronization of tasks between the laser and camera modules; and
- Power control by another micro-controller, the Arduino in this case.

These two Raspberry Pi's illustrated in Figure 2.13 must work together to complete their measurement tasks; therefore, their operations are discussed together. Both Raspberry Pi's are connected via wired GPIO pins to an Arduino, which is poised to turn on and off the power supply to each respective Raspberry Pi to conserve overall system power consumption. Therefore, at boot-up, each Raspberry Pi must perform **Initialization** tasks, including driving a "True" signal through a GPIO pin being read by their corresponding Arduino. This alerts the Arduino that the Raspberry Pi is on. Also, each Raspberry Pi references a text-file, called `run_log.txt`. This file contains a single integer `run_count`: a counter that starts at 0 and encodes the current boot-up that the module has performed that is only incremented at the end of the program. Each Pi also maintains a settings file, which contains hard-coded information that the main program references. One of these variables is `sync_interval`, the number of runs that must elapse before a clock



(a) Camera module Raspberry Pi

(b) Laser module Raspberry Pi

Figure 2.14: State diagrams for Phase I deployment of the Raspberry Pi.

synchronization must be performed between each module. Both Pi's at boot-up also check if the current run is a multiple of `sync_interval`; if it is, each Pi will perform clock synchronization tasks instead of measurement tasks. If it is not, each Pi will proceed onto measurement tasks. The laser module, as part of **Initialization**, will also turn on the laser diode.

Next, the Pi in the camera module, RPI_{camera}, will access its accelerometer to calculate a pitch value and save it to `pitch_camera` in the state **Acceleration**. Then, it will proceed to take the required images using its camera in the state **Image**. Following this, the RPI_{camera} commands the RPI_{laser} to turn off the laser diode. The RPI_{camera} then also performs the computer vision algorithm in this state to extract the centroid of the laser beam, $[p_x, p_y]$. All the while, the RPI_{laser} has been in its **Wait** state as the capturing of the images takes an appreciable amount of time. The RPI_{laser} therefore is designated its own idle state, waiting to receive the command from RPI_{camera} to shut off the laser diode. After receiving this message, the RPI_{laser} calculates its own tilt value `pitch_laser` from reading its own accelerometer in its state **Acceleration**. After **Acceleration**, the RPI_{laser} then proceeds to transmit `pitch_laser` to the RPI_{camera} in the state

Send Pitch Reading. All the while, as reading an accelerometer does not take long to execute, the RPI_{camera} has moved onto its **Settlement Estimation** state without entering a separate Wait state, and receives the `pitch_laser` reading from RPI_{laser}. After RPI_{laser} sends out its pitch reading, all of its duties have been completed, so it proceeds to its **Shutdown** state. In doing so, it drives its designated GPIO pin "LOW", or "False" to indicate to the Arduino that the Raspberry Pi will shutdown momentarily. Before issuing a shutdown command, the RPI_{laser} will update its `run_log.txt` with an incremented `run_count`.

In the meantime, the RPI_{camera} has moved to state **Server**, whereby it connects to the server, sends the compiled data $[p_x, p_y, pitch_{camera}, pitch_{laser}]$ to the server, and then proceeds to **Shutdown**, which is identical to the **Shutdown** state for the RPI_{laser} except that the camera module must also shutdown its camera sensor. If the camera is not shutdown properly, the resources that are shared between the camera board and the Pi are not de-allocated properly, and the Raspberry Pi is at risk of suffering from abnormal memory segmentation faults, which are typically fatal and may cause the Raspberry Pi board to freeze.

Figure 2.14 shows additional state transitions in red. In the execution of each state, if any significant error is caught (e.g., if the camera is not accessible), the entire system transitions to its **Shutdown** state. This ensures that the unit does not prematurely exit from the program execution and shutdown improperly, which is considered a "fatal" handling of the error.

An important question is now posed; if one unit shuts down, how will the other unit know to shutdown? As will be discussed in a subsequent section on the communication protocol for this system, there exists certain states that require the passing of messages between the two units. For instance, in order for RPI_{laser} to transition from **Wait** to **Acceleration**, it must receive a command from RPI_{camera} to turn off the laser. If RPI_{camera} has somehow encountered an error prior to sending out that command in **Image** and shut down, RPI_{laser} will never receive this message. To prevent RPI_{laser} from hanging and entering a phenomenon called deadlock in which the system no longer responds to the provided inputs and stalls, we consider this lack of a message to be an error in itself, and our communication protocol will raise the error to the state machine. This error then results in RPI_{laser} proceeding to **Shutdown**.

Lastly, we note that if `run_count % sync_interval == 0` for both modules exiting their respective **Initialization** states, they will both transition to a **Synchronization** state. In each of these states, the Pi's will perform time-synchronization procedures to correct the clock offset that may have accrued between boot-ups. This time synchronization procedure will be explained in more detail in Section 2.3.2. Each Pi then proceeds onto their respective **Shutdown for Synchronization** states.

We have therefore addressed each of the three concerns mentioned earlier in this subsection. These state machine designs allow for the Raspberry Pi's to implement their measurement tasks while also non-fatally handling unanticipated errors, correcting for clock drift, and communicating with the Arduino to implement power-saving operations. We can next proceed to discuss in further detail the state machine model for the Arduino.

Arduino Design and Implementation

The Arduino plays an important role in the functionality of the computation unit of the proposed monitoring system: it controls when to power the Raspberry Pi on or off. First, we designed a state machine for the Arduino, as illustrated in Figure 2.15.

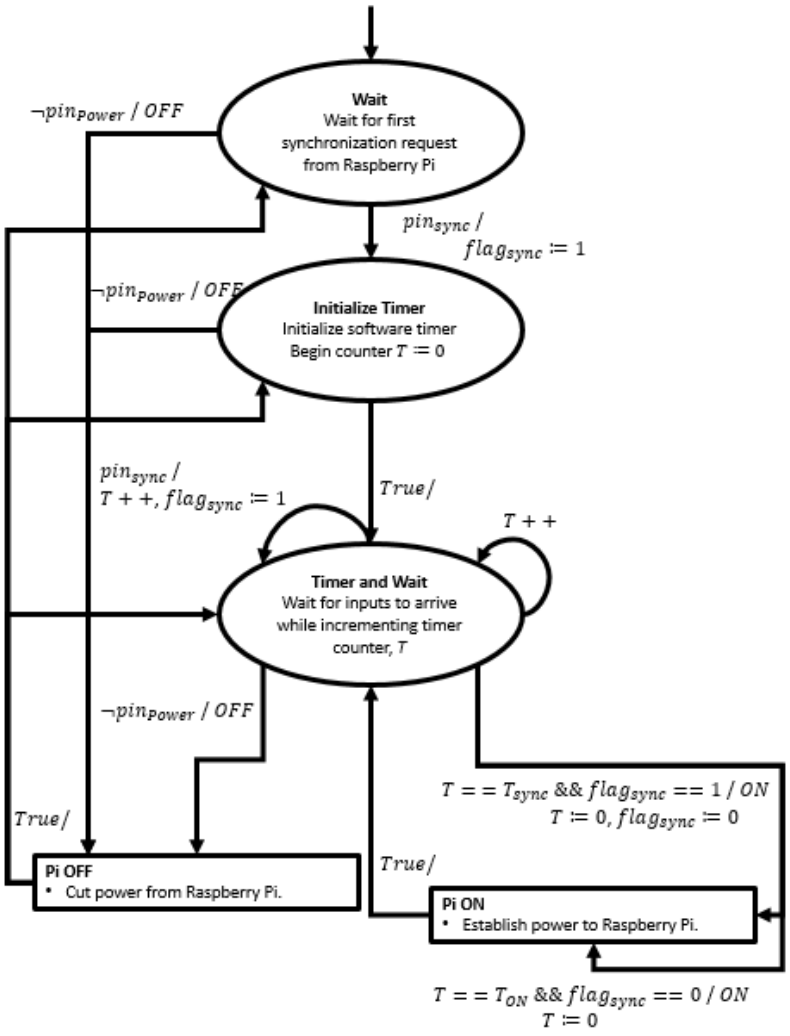


Figure 2.15: State diagram for Phase I deployment, Arduino.

This state machine is time-triggered instead of event-triggered. A time-triggered system reacts based on an internal timer, while an event-triggered system reacts to external stimuli based on interrupts [Lee and Seshia (2017)]. When implemented in code, the Arduino "polls" the state of certain input/output pins and reacts accordingly. A discussion on the difference between interrupts and polling is provided in a subsequent section.

Note: the state machine model for the Arduinos in both the laser and camera modules are identical as their respective "slave"—the Raspberry Pi's—ultimately provides the same inputs to accomplish the exact same goals. These goals are as follows: (1) inform the Arduino when the Raspberry Pi is powered "on" and "off"; and (2) inform the Arduino if and when the Raspberry Pi

wishes to perform time synchronization. This information is transmitted to the Arduino via wired GPIO pins. As discussed below, it is desirable to synchronize the clocks of both the camera and laser modules when they are both booted-up for the first time.

The Arduino boots-up and begins in state **Wait** as it waits for a signal from its Raspberry Pi to synchronize. This signal is transmitted via a connected GPIO pin to signal to the Raspberry Pi that it needs to pivot from "LOW" to "HIGH" when a time synchronization is desired. Once this signal is received, the Arduino will initialize the software timer (based on a physical hardware oscillator) and begin a counter variable `T` at zero. The machine then automatically transitions to the predominant state of the diagram, **Timer and Wait**, in which the Arduino will wait for various inputs or conditions in order to transition to other states and perform actions. These rather intuitive transitions are:

1. If `T` matches a value `T_sync` and a condition variable `flag_sync`, the Arduino will turn on the Raspberry Pi;
2. If `T` matches a value `T_ON` and `flag_sync` is not active, the Arduino will turn on the Raspberry Pi;
3. If the Arduino receives a "LOW" pin_{power} signal (i.e., $\neg pin_{power}$) from the Raspberry Pi, the Arduino will cut the power going to the Raspberry Pi (*Note: $\neg$ is a logical "not" symbol that signifies the absence of a signal, or if a signal is "LOW" or "False" versus "HIGH" or 'True"*);
4. If the Arduino receives a "HIGH" pin_{sync} signal (i.e., pin_{power}) from the Raspberry Pi, the Arduino will turn the condition variable `flag_sync` to be active; and
5. If no other transitions are made, `T` will be incremented.

The incrementation of `T` ensures that the conditional statements of `T == T_sync` and `T == T_ON` in the first two enumerated transitions above will eventually be satisfied.

If the machine has transitioned from **Timer and Wait** to either **PI OFF** or **PI ON**, the program will immediately return back to **Timer and Wait** following the execution of **PI OFF** and **PI ON**. In this way, we ensure that the incrementation of `T` is interrupted as infrequently as possible. In addition, when the Arduino transitions to **PI ON**, not only is the command made to turn on the Raspberry Pi, but the counter variable `T` resets to zero, as well as any condition flag such as `flag_sync`. When the Arduino transitions to **PI OFF**, `T` is not reset. The reasoning behind this protocol is best illustrated in an example program flow.

First, assume that `T_sync` is significantly less than `T_ON`, `T_sync = 2 minutes` and `T_ON = 12 hours`, meaning that when the Pi is synchronizing with the other module, it will shutdown and wake up again after only 2 minutes of elapsed time instead of waiting a full 12 hours. Next, assuming that we are booting the monitoring system for the first time, `run_count = 0`. Therefore, we know that the Raspberry Pi, according to its state diagram in Figure 2.14, will perform time synchronization instead of measurement tasks.

Upon boot-up, the Raspberry Pi will send pin_{sync} to the Arduino, thus allowing the Arduino to transition to **Initialize Timer** from **Wait** and also set `flag_sync := 1`. After initializing the software timer and `T := 0`, the Arduino then transitions to **Timer and Wait**. If the Raspberry

Pi had already finished its synchronization tasks and shutdown, a $\neg pin_{power}$ signal would be sent to Arduino, thus allowing the Arduino to cut the power to the Raspberry Pi with the command **PI OFF**.

We have now arrived at a state where the Arduino is in **Timer and Wait**, and the Raspberry Pi has shutdown, i.e., its power has been cut. The Arduino will establish power back to the Raspberry Pi once a pre-defined time `T_sync` has elapsed, as noted by one of the transitions to **PI ON** during which `T` is reset. The Arduino now is back in **Timer and Wait**, and the Raspberry Pi is now performing measurement tasks instead of synchronizing with the Arduino. Eventually, the Raspberry Pi will shut itself down and will send $\neg pin_{power}$ to the Arduino, which will trigger a transition to **PI OFF** and back. Note that this time, `T` has not been reset and `flag_sync == 0`. Therefore, when `T_sync` has elapsed, the power is not re-established to the Raspberry Pi. The Arduino waits until `T_ON` has elapsed and then re-establishes power.

We have explained the expected execution of the Arduino state machine in relation to the expected execution of the Raspberry Pi state machine. In doing so, we have now established that the Arduino is expected to power the Raspberry Pi on and off in accordance with the task required of the Raspberry Pi, be it time synchronization or measurement.

2.3.2 Time Synchronization

Time synchronization is required because of error that a timer accumulates as it counts through time, typically due to imprecision of the hardware of the timer itself [Lee and Seshia (2017)]. For instance, in the case of the Arduino UNO, its timer features are implemented using the 16 MHz crystal oscillator located on its board; literally a quartz crystal oscillates at a known frequency, which gives the micro-processor a notion of time. In addition, there is an initial existing offset because the two modules cannot be booted simultaneously; this offset must be corrected. Each Arduino unit has no concept of actual real-world time without additional outside information, e.g, from the Internet. It only knows how much time has elapsed "locally" from some initial point in time, say, when a software timer is initialized.

Clearly there are two sources of time error: (1) accumulated clock drift due to hardware imprecision; and (2) an initial clock offset. To correct for this time error, we implement a synchronization protocol called the Precision Time Protocol (PTP) [Lee and Seshia (2017)], which is illustrated in Figure 2.16.

Implementing the PTP will solve for the accumulated error between two separate clocks, assuming that one clock is a "master," and the other is the "slave": essentially we synchronize the slave clock relative to the master clock. This protocol calculates this time error by passing encoded time stamps between the two clocks and assumes that the time of flight in both directions is equal, which is a fair assumption when trying to obtain a reasonable level of synchronization precision. Referencing Figure 2.16, it should be noted that the actual messages passed between master and slave are encoded time stamps. For instance, t_1 is the time that t_1 was sent from the master. It is then received by the slave at a time t_2 (in the time context of the slave). The error term e accounts for the difference in time references, which is exactly the time error we want to solve for. Ultimately, the error can be solved using the following equation:

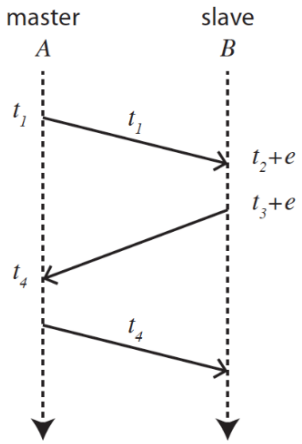


Figure 2.16: Precision time protocol (PTP) [Lee and Seshia (2017)].

$$e^* = (t_2 + e) - t_1 - \frac{r}{2} \tag{2.2}$$

where e^* is the exact clock error if the communication latency, or travel time, is symmetric between the master and the slave, and r is the round trip travel time, defined as follows:

$$r = (t_4 - t_1) - [(t_3 + 3) - (t_2 + e)] \tag{2.3}$$

To implement PTP synchronization, the Raspberry Pi's will pass these various messages between each other over WLAN, and after the error e^* is computed, both Raspberry Pi's will pause for program execution for a unique period of time using the Python `time.sleep()` method such that when they both resume program execution, they are indeed synchronized. It is expected that one Pi will have "waited" longer than the other to account for this error. When a Pi resumes execution, it will immediately command the Arduino with pin_{sync} to signal both the desire to synchronize as well as when to synchronize, and then shutdown. Its Arduino will receive this signal and immediately initialize its software timer and counter. The other Pi will send its own pin_{sync} signal eventually to its own Arduino to initialize its Arduino's software timer and counter, and then also shutdown. Both of these signals would have been sent at the same time, thereby initializing the timers of both Arduinos at the same time. Therefore, when the Arduinos re-establish power to their respective Pi's after waiting for the same `T_sync` to have elapsed, the Pi's will reboot at the same time.

2.3.3 Communication and TCP/IP

Still to be addressed is how messages are passed between the Pi's. Across a WLAN, we can employ TCP sockets to pass messages from one device on the network to another. The Python module `socket` offers users high-level functionality of TCP/IP communication. Transmission Control Protocol (TCP) is advantageous as it is reliable; packets sent according to this protocol that are dropped are automatically detected and re-transmitted by the sender. In addition, TCP has

in-order data delivery, meaning that data is read by the recipient in the same order it was sent by the sender [5], thus removing any worrying about issues such as packet loss, out of order data arrival, and other issues. The Python `socket` module then implements this protocol using a notion of sockets by adopting the framework of a server and client (master and slave) device; see Figure 2.17. Note in Figure 2.17 how the server end must initially create the socket object, bind its own IP address to that socket object, and "listen" for connection attempts from other client devices. In our system, we designated the camera module as the master device (or the server in this case) and the laser module as the client. Once this connection is made via a three-way handshake, the messages are passed using send and read methods, and after one party decides to end the transaction, the TCP socket is closed.

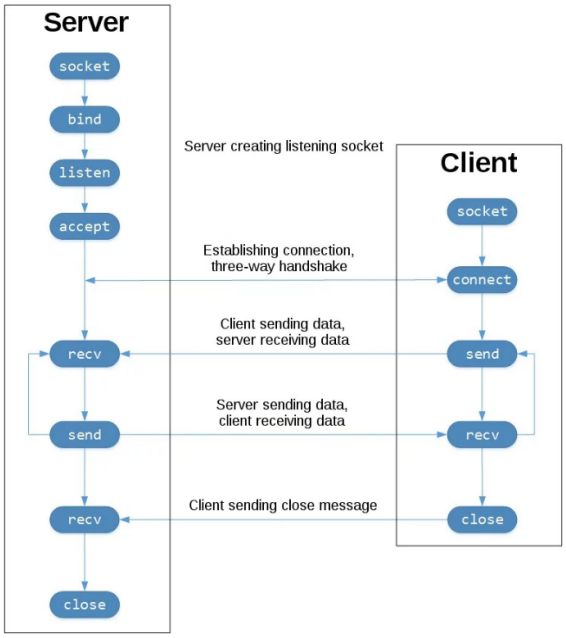


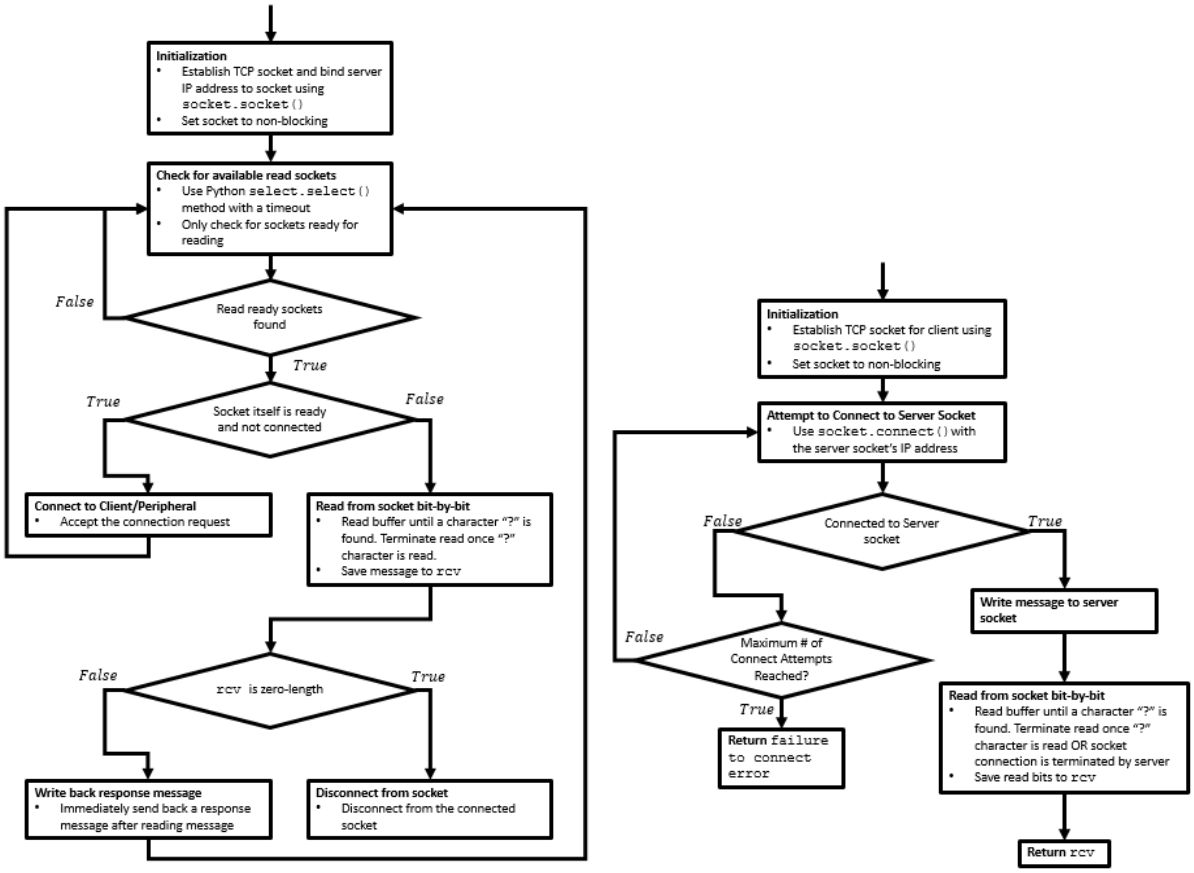
Figure 2.17: TCP socket flow [3].

During measurement tasks, our Raspberry Pi's will communicate according to a protocol that uses the TCP socket flow as illustrated in Figure 2.17. To construct this protocol, we employed a helpful property of socket communication: blocking or non-blocking sockets. When a socket is set to "non-blocking," it means that the socket will not wait indefinitely on certain "receiving" methods for something to arrive. For instance, the server socket must "listen" for incoming connection requests from the client socket in order to make a connection. If set to "blocking," this socket will wait indefinitely for that connection request, which runs the risk of placing the invoking program (in our case the primary state machine program) into deadlock. Deadlock is a phenomenon in which the system no longer responds to inputs and stalls because it is waiting for some event to occur, often when such event will actually never occur.

To prevent this, we set our sockets (client and server) to be non-blocking. For example, a non-blocking server socket will only check for a connection request once and in that one instance. If a client connection request has not arrived, the server socket will return to the invoking program without making a connection. Another example is that from the client end when the client is

reading a message from the socket buffer. Instead of waiting indefinitely for a message to arrive in the buffer, the socket will only try once instantaneously (or for a certain amount of time called a timeout) before returning. This useful property allows one to employ these socket methods in program loops to construct their own communication protocols, such as the one developed for this project. Our protocol accommodates a very specific communication protocol whereby the server will only wait for a designated amount of time for a connection, and the client will only query for a connection for a designated maximum number of times. If a connection is created, the client will immediately send out a message, and the server will immediately begin reading its buffer. Once the server socket reads the message, it immediately sends out a response. Once the client has sent out its message, it will immediately begin reading for this server response message. This protocol is illustrated in Figure 2.18.

We made extensive use of the Python module `select()`, which monitors input/output ports. For example, if `select.select()` is given a non-blocking TCP/IP socket as an argument, it will return lists containing communication channels that are ready to read from, ready to write to, or have reported an error.



(a) Server (a) Client

Figure 2.18: Phase I functional block diagrams and communications protocol.

2.3.4 Power System

Because we have no intention of field-deploying this system, we were not concerned regarding power consumption. A more rigorous discussion of power consumption will be made in discussions of Phase II.

The last fundamental questions in developing the Phase I system are how the Arduino physically receives the pin_{power} and pin_{sync} signals, and how the commands to turn the Raspberry Pi "On and Off" are implemented. To implement this power delivery system, a prototype circuit was fabricated using various components including a Low-Signal (5 VDC) Latching Relay; see Figure G.1. There are various components and signal lines that are important to note in Figure G.1 and are tabulated in Table G.1. Of importance is the use of pull-down resistors as well as the choice of using the latching relay. We installed a 1N457 diode across all of the relays in our circuits.

Pull Down Resistors

As will be mentioned in a subsequent section on implementation challenges, the power delivery system relies on the signal lines carrying pin_{sync} and pin_{power} to be accurate. The Arduino will cut off power to the Raspberry Pi when pin_{power} drops from HIGH to LOW; if the signal does so when the Pi has not yet completed its tasks, this could cause severe ramifications on the operation of the computational unit and may damage the micro-controller itself.

An example of how this signal may have become corrupted is when the Raspberry Pi first boots up. It is known from the BCM chip datasheet that all 54 GPIO pins on the Raspberry Pi 3B+ have a pre-set pull up or down resistor built into them [6]. This ensures that when the pins are not being driven, their voltages are not "floating"; however, it takes time for these built-in resistors to activate at boot-up. While this is usually not a problem, we installed pull down resistors on the two signal lines to ensure that even during boot-up, the signals are pulled low even if the pull-down resistors in the Pi have not taken effect yet. This ensures that pin_{sync} is low until the Pi decides to either drive it high to indicate a synchronization request or keep it low. This also ensures that pin_{power} is always low until the Pi drives it high to indicate that the program is running, and then drives it low to indicate that the main program has terminated. For typical low-voltage applications such as this one in which logic values are defined below 5 V, an adequate pull down or up resistor is $5\ k\Omega$ [7].

Latching DPDT Latching Relay

To actuate the closing and opening of a switch that carries the power for the Pi, we used a latching relay. A relay can be thought of as a simple switch but with coils that are energized/de-energized that close or open that switch. These relays can be latching or non-latching; the latter meaning that in order to maintain the current state of the switch, the coil must be constantly energized. If the coil is de-energized, the switch automatically reverts back to its default state. Since we wanted more control over our power delivery, we opted for the latching relay in which there may exist two relay coils, one of which is to be driven high and the other low. At that point, the switch "latches" and does not revert back to its previous state. The only way for the switch to revert is to drive the high coil low and the low coil high. This sort of behavior is illustrated in the internal schematic of the double pole-double throw (DPDT) latching relay we selected for our circuit; see Figure 2.19.

An additional concern, although subtle, is important. The coils in the relay act as a solenoid in which the current carried in the coils generates a magnetic field, which drives an electromagnet in the relay to close or open the switch. When the coil is de-energized, the magnetic field induced by that solenoid collapses, essentially creating its own electro-motive force (EMF) according to Lenz's Law [9]. This EMF travels in the direction opposite to that of the original driving current, hence the name "back-EMF." Depending on the voltage and currents in play, this back-EMF could be large enough to damage the electronics. To capture this back-EMF, we use a "flywheel diode," as illustrated in Figure 2.20. When the driving current is active, the diode is reversed biased to the flow of the current and will not allow any current to pass. When the switch opens, the diode polarity matches the direction of the back-EMF, diverting the back-EMF away from the driving electronics.

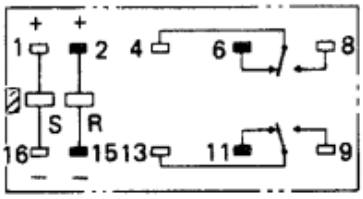


Figure 2.19: Terminal schematic for DPDT relay: G6AK-234P-ST-US-DC5 [8].

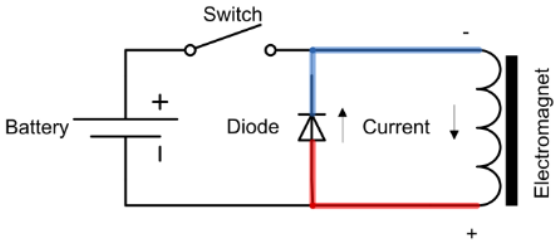


Figure 2.20: Back-EMF suppression using a diode [9].

2.3.5 Computer Vision Algorithm

The computer vision algorithm must take in captured images and output an estimate of where the center of the cross-hair beam is on the image in units of pixels. The algorithm must be robust in that it can accommodate light artifacts (e.g., streaks of light or smudges on the screen), and it must be able to return a position that is of sub-millimeter precision. Indeed, this is a requirement for the entire monitoring system; to return a settlement measurement of sub-millimeter precision but with all components of the monitoring system considered: the computational unit, sensing interface, and communication pathway. Their role is to ensure that the proper images are captured and given to the algorithm so that it can generate output. Presented below is a block diagram that explains at a high level the function of this computer vision algorithm.

Note in Figure 2.21 that it requires two images: one of the laser beam and another of the baseline ambient conditions. This way, the algorithm can remove significant light artifacts by

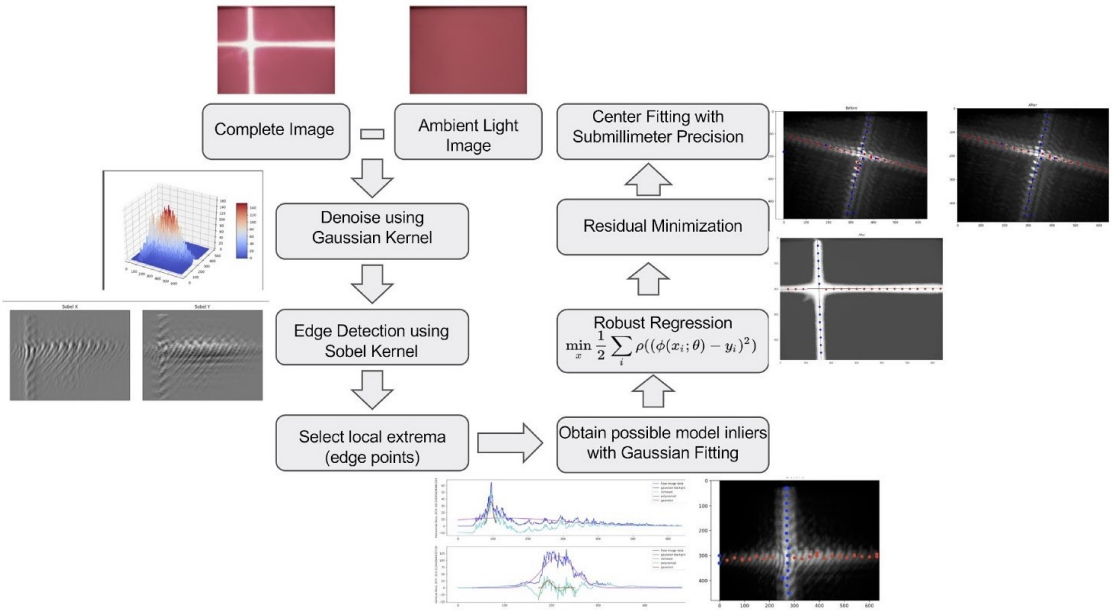


Figure 2.21: Block diagram of developed computer vision algorithm.

merely subtracting the baseline image from the laser beam image. This corrected image is then de-noised using a Gaussian filter, and the gradients of the pixel values are then calculated using the Sobel operator. From these gradients, regression points are found upon which a regression will be made. In essence, an edge-detection method is employed, very similar to that done by Olaszek (1999). In the end, the intersection of the two regression curves is output as the centroid estimate. Also key is that a restriction imposed on this algorithm is that it is constructed to work only on pictures taken at night, as it is safe to assume that ambient light is at a minimum during after dark. Further details on the computer vision algorithm can be found in Appendix E.

2.3.6 Implementation Challenges

Crystal Oscillator Imprecision

As shown in Figure 2.22, the 16 MHz oscillator located on the Arduino has tolerances on the nominal frequency of oscillation. The tolerance of the oscillators used on the Arduino is around $\pm 50\ ppm$ [10]. Thus, if we expect the oscillator to have a frequency of 16 MHz, the actual frequency may deviate by:

$$F_{deviation} = \left(\frac{50PPM}{1,000,000} \right) * F_{Ideal} = 800Hz \tag{2.4}$$

This deviation can have serious implications regarding time-keeping by the Arduino across long time spans, say, 12 hours, which is a reasonable measurement interval in terms of monitoring. During testing of this system, the software timer ran for over 30 hrs, resulting in the two Raspberry

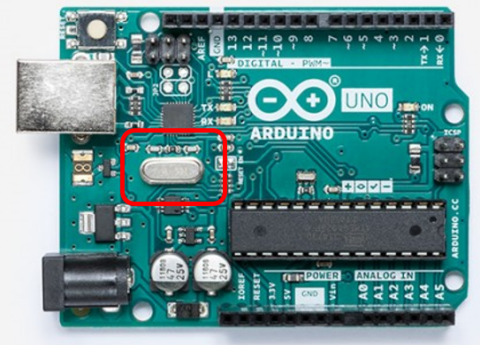


Figure 2.22: Location of crystal oscillator on Arduino UNO Rev 3.

Pi units waking up with a delay of 52 sec, i.e., one of the Arduino clocks had accumulated 52 sec of drift over 30 hrs. To correct this, one method was to adjust the value of the timer's compare match register.

The logic behind this solution is as follows: the Arduino's ATmega328 micro-controller [11] has 8-bit and 16-bit counters, meaning that each counter will either reach 2^n or a compare match value, and then restart at 0, where n is the number of bits in the counter [5a]. An interrupt can be interpreted as an action(s) that must be done by the micro-controller when a certain event triggers the interrupt. When that event occurs, an interrupt handler "handles" the event and performs the programmed action(s). Because this interrupt mechanism itself is already implemented in the micro-controller, the user need not be concerned. In the case of a timer interrupt, the event that triggers the action(s) to be performed is either the counter reaching its maximum value and overflowing, or when the counter value matches the compare match value, which is stored in the compare match register. For reference, a register is merely a small parcel of memory in a computing unit and is often associated with an address.

Ultimately, the user needs only to apply Equations (2.5) and (2.6) to control the frequency by which the timer interrupts occur [12]. In our case, we wanted timer interrupts to occur every second, so that the Arduino can check to see if a certain amount of time has elapsed (in seconds).

$$Freq_{Timer}[Hz] = \frac{Freq_{Crystal}[Hz]}{Prescaler} \tag{2.5}$$

$$Freq_{Interrupt}[Hz] = \frac{Freq_{Timer}}{CMV} = \frac{Freq_{crystal}}{Prescaler * (CMV - 1)} \tag{2.6}$$

Note that the -1 term in Equation (2.6) exists because the Compare Match Value (CMV) is zero-indexed, and that the Arduino Crystal Frequency is nominally 16 MHz. To set our nominal 1 Hz timer, we use a default *Prescaler* (defined in the Arduino architecture) of 1024 and compute the CMV to be 15,624. Note: it was with this exact value that one Pi accumulated 52 sec of error over 30 hrs. That error is due to the tolerance of the nominal frequency of the crystal. By empirically correcting for this error, we can back-calculate a corrected CMV to be 15,614 instead, which is reasonable as a smaller CMV speeds up the frequency of the timer interrupt.

By tuning the CMV, we were able to correct for this relative time error between the camera and laser modules; this may not have been necessary for this monitoring application. In the context of the communication protocol illustrated in Figure 2.18, we can have the server and client sockets wait longer for the other module to boot-up and connect. Referencing the state diagrams in Figure 2.14, once both Raspberry Pi's have transitioned out of their **Initialization** states, the rest of the measurement tasks or synchronization tasks will be in-sync with each other. That being said, this issue did bring up an important limitation of this implementation of the monitoring system. While we can empirically tune timer properties to reduce clock drift and also employ PTP calculations to synchronize the modules relative to each other, we need some sample of the actual world time to synchronize the units absolutely. A straight-forward way would be to access the Internet. This would require a gateway to the Internet instead of merely tying all of the various components together across a WLAN. Ultimately, for the purposes of this laboratory-deployed system, we deemed the implemented time-synchronization methods to be adequate.

Interrupts or Polling

Another interesting complication that arose during implementation involved the passing of the signals pin_{power} and pin_{sync} from the Raspberry Pi to the Arduino. During testing, it was observed that the signal would often flicker during program run time. While the pull-down and pull-up resistors on the signal lines ensure that the signals are stable when there exist floating voltages before and after the GPIO pins on the Raspberry Pi are fully initialized and driven, one problem they could not resolve were environmental factors. The presence of static electricity and other ambient conditions would sometimes cause the pin_{power} signal to flicker from High to Low, thereby falsely indicating to the Arduino that the Raspberry Pi had shut itself down and was ready for its power to be cut. This resulted in premature shutdown of the computational unit. Granted, this could have been due to poor craftsmanship and soldering of the circuit board, as described in the Power System section of this chapter, but enforcing quality control of electronics can be difficult and required implementing a software solution to address this problem.

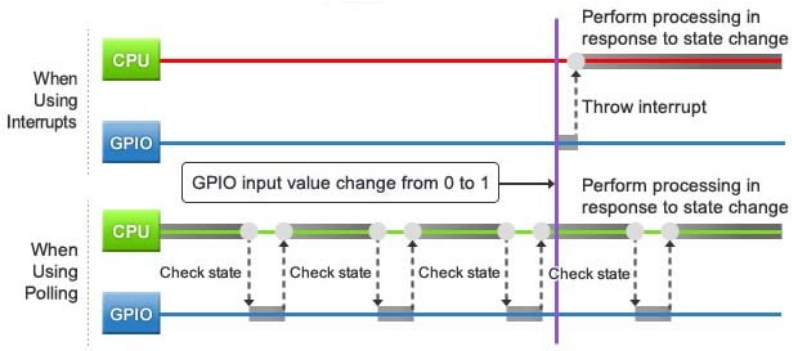


Figure 2.23: Poll v. interrupt mechanism.

Interrupts react to stimuli nearly instantaneously and can be of two types; external or software (internal) [Lee and Seshia (2017)]. For internal software interrupts, the micro-controller typically has some sort of timer programmed to trigger certain interrupts when certain time values are met. This is how the Arduino was programmed to check if `T_sync` or `T_ON` time had elapsed in order to turn on the Raspberry Pi, using a timer interrupt. For external interrupts, the micro-controller is typically programmed to constantly monitor a certain pin for a certain event, be it a logic level (High or Low), or a falling edge (High to Low), or a rising edge (Low to High). Once that event occurs, an interrupt handler handles that event and performs certain programmed actions in response to that event. This is how we initially implemented the Arduino to respond to pin_{power} and pin_{sync} signals as it has very low latency. If the signal itself has abnormalities, such as flickering, these interrupts will cause unexpected and oftentimes unwanted behavior. To counter this, we implemented polling.

Polling is very similar to interrupts, in the sense that the micro-controller is monitoring the state of a pin. Instead of the event triggering a response from the micro-controller, the micro-controller must first detect if the event has occurred and then respond to it. This slight difference is illustrated in Figure 2.23. We can see that depending on when the event actually occurs in relation to when the micro-controller samples the GPIO pin, the latency between the event occurrence and the micro-controller response can be quite high. However, when we implemented the Arduino response to pin_{power} and pin_{sync}, using polling at a sampling frequency of 100 Hz, we were able to eliminate any unexpected behavior due to flickering GPIO pins as the frequency of the signal abnormalities was higher than the polling frequency.

2.3.7 Programmatic Implementation

The actual deployment of the designed state machines and protocols in code was done predominantly on Python. All code described in Phase I will be available in a Github, as well as documented in Appendix H. In addition, it is important to understand how all of the different function files are organized and dependent upon each other. This organization and dependency is illustrated in Figures G.2 and G.3.

Note: the Arduino only has one program flashed onto it at a time. This one program may include certain libraries that may already be saved in memory, but there is only one written program that implements the Arduino state machine. Because of its OS, the Raspberry Pi has a file system that is organized as illustrated in Figure G.2. To ensure that the system runs autonomously, the main codes `RPI_C.py` and `RPI_L.py` on each module are run automatically when the system boots-up. This is done automatically by editing the configuration file in the Raspberry Pi that controls the actions done at boot-up, `/etc/rc.local`. To shutdown the Raspberry Pi programatically, a command line shutdown command, "`sudo shutdown -h now`" must be issued.

2.4 TESTING AND EVALUATION

With our system properly designed and implemented, additional steps still remained before we could run laboratory experiments to test this system. They were as follows: (a) mount the hardware in an appropriate enclosure; (2) calibrate the projected laser target system; and (3) run a laboratory

experiment to evaluate how the completed system operates.

2.4.1 Test Enclosure Design and Fabrication

For laboratory-deployment, the requirements for the enclosures of both the laser and camera modules are different than those if the system were to be deployed in the field. We were not required to design an enclosure that is weatherproof. Therefore, in this phase of the project, we designed a mounting device for both the laser and camera modules in SolidWorks, and 3D printed them using an in-house 3D printer using PLA (polylactic acid) material. These designs were only intended to validate the concept of the projected laser target method. The minimum requirements of the enclosures and mountings required for this system to return accurate results are listed below:

1. The camera chamber must eliminate the majority of the outside light to ensure consistent lighting conditions for the camera;
2. The camera chamber geometry must be compatible with the field of view of the image sensor;
3. The camera module, which includes the camera chamber, should be able to hold the acrylic panel in place;
4. The laser module must be able to hold the laser firmly in place through the duration of the testing; and
5. The laser module must allow for the adjustment of the trajectory of the laser diode.

Various Designs of Camera and Laser Enclosures

With these requirements in mind, we designed two laser enclosures and two camera enclosures, each of which would be suitable for various situations. Pictures of these enclosures are included in Figures G.4, G.5, G.6, and G.7. In Figures G.4 and G.6, we can see various iterations of the camera module. The primary improvement of the camera module v2 versus camera module v1 is that whereas v1 mounts the hardware on the exterior faces of the camera chamber [as seen in Figure G.4(b)], the v2 module features another wall surrounding the camera chamber that encloses the electronics, as seen in Figure G.6(a). In both versions, a lid encloses the camera chamber, with the camera itself mounted on the rear wall farthest from the acrylic screen and facing the screen.

As shown in Figures G.5 and G.6 the laser modules hold the laser firmly in place by fastening the cylindrical laser diode using set screws that penetrate a cylindrical chamber housing the laser diode. This cylindrical chamber is incorporated in a fixed mounting, as seen in Figure G.5, and is set in a ball-and-socket joint in Figure G.6(b). This latter mounting allows the user to install the laser off axis from the camera module (perhaps on a wall), and aim the laser beam at the camera module, which may be located somewhere else in the room. After aiming the laser beam, the user can set the positioning of the laser by tightening the set screws that penetrate the socket component.

It is possible to mount the camera module on various types of specimens so long as the camera chamber itself is kept intact. In other words, one may design an enclosure that mounts the hardware in other orientations, or perhaps includes mounting devices to affix the entire chamber on a certain specimen. So long as the camera chamber, which includes the camera, acrylic panel,

and the walls that define the field of view of the camera, is kept intact, the projected laser target method is viable. For instance, note that in Figure G.5 the entire camera chamber can be affixed onto a tripod for mobility in a laboratory setting, and in Figure G.7 the entire camera enclosure can be affixed onto the top of a concrete cylinder (analogous to a cylindrical structural or foundational element) via a metal frame.

To ensure the efficacy of the projected laser target method, much effort must be put into the camera module. It requires the proper dimensioning of the camera chamber, the proper orientation and mounting of the camera within the chamber, and the accurate calibration of the camera chamber.

Camera Chamber Design

The slanted walls of the camera chamber follow the field of view of the camera, as illustrated in Figure 2.24. Depending on how much vertical and horizontal translation of the projected cross-hair beam we want to capture on the acrylic panel, we are restricted to the horizontal and vertical fields of view of our camera. For the Raspberry Pi Camera v1, the horizontal field of view is 53.5 ± 0.13^o, and the vertical field of view is 41.41 ± 0.11^o [6]. Using these values, we can calculate how far we must install the camera from the acrylic panel in order to achieve a certain horizontal and vertical range of measurement.

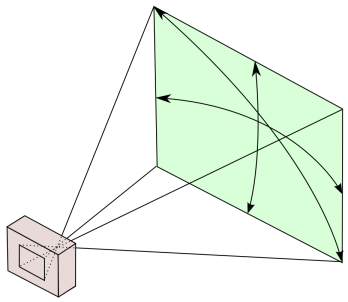


Figure 2.24: Field of view of a camera.

In addition, we can see that the horizontal field of view is greater than the vertical, meaning the horizontal range of the camera is greater than the vertical range. Therefore, as we are interested in capturing settlement, i.e., a vertical displacement, we oriented the camera 90^o rotated, so that the horizontal field of view as per the spec sheet is actually oriented vertically. In order to achieve a range of 4 in. of vertical measurement range, we calculated that we needed to install the acrylic panel 3.97 in. away from the acrylic panel, per Equation (2.7).

$$depth = \frac{h}{2tan(0.5 * \theta)} = \frac{4}{2tan(0.5 * 53.5^o)} = 3.97\ in. \tag{2.7}$$

We note that accounting for the uncertainty in the field of view, the depth would only vary by $+/-0.011$ in., which is trivial when considering fabrication tolerances; therefore, we can neglect this uncertainty and work with the expected field of view calculation. This mounting depth will also result in a horizontal range of 2.99 in. With these calculations, we must be sure when

designing the chamber to mount the camera such that the lens is mounted exactly at the center of the expected field of view. At that point, we can build the rest of the camera chamber and enclosure around this space bounded by the camera and field of view, as illustrated in Figure 2.24.

2.4.2 Calibration of Camera Module

We must now consider how to calibrate the camera module; in this case, calibration means obtaining a scale factor between pixels and an engineering unit of length, say, inches or millimeters. Per Feng and Feng (2016), this scale factor can be determined in two ways: (1) estimated based on the known physical dimensions of the captured object and its corresponding image dimension in pixels; or (2) estimated based on the parameters of the camera and the parameters that relate the captured object in real life to the camera. Ultimately, we chose to adopt the former option, as the latter relies on information concerning how the object being photographed is spatially oriented to the camera as well as the precision of the documented properties of the image sensor, such as its focal length [Feng and Feng (2016)]. If adopted, these intrinsic means of calculating scale factor could complicate the implementation by requiring certain mounting procedures that are unrealistic to perform in the field or having tighter quality control on component assembly and selection, which may add significant cost of the developed system and be prohibitive in massive deployment on a state-wide bridge network. To side-step these complications, we can obtain this scale factor by constructing a calibration curve on the camera chamber through the method prescribed below:

1. Mount the camera chamber and laser emitter a short distance away from each other;
2. Capture a baseline image of the initial location of the laser beam on the acrylic screen;
3. Shift the vertical location of the laser beam;
4. Capture another image of the shifted location of the laser beam;
5. Repeat Steps 3 and 4 with different vertical locations of the laser beam;
6. Extract the beam centroid $[p_x, p_y]$ of the captured images; and
7. Fit a regression line to the expected linear data, which is the calibration curve.

Note: we are interested in the vertical position of the laser beam because we want to obtain the vertical scale factor. Indeed, although the scale factor between the horizontal and vertical axes is expected to be different due to different pixel-image resolution between the two axes, we are only concerned with the vertical resolution.

This calibration procedure is illustrated in Figure 2.25, where three photos were taken at three different offset values of the camera chamber. The laser beam was oriented off-perpendicular from the camera screen to show that although the orientation of the laser is insignificant, its fixed orientation is significant. The slope of the resulting calibration curve is of units $[\frac{pixels}{mm}]$.

In Step 1, we mounted the camera chamber a short distance away from the laser diode to minimize the possibility of laser distortion due to laser diffraction, scattering, etc; the laser beam need not be projected exactly perpendicular to the screen so long as the orientation of the beam does not change through subsequent images. Note: at this stage we did not concern ourselves with the monitoring system and the full camera enclosure. As the camera installation and the camera

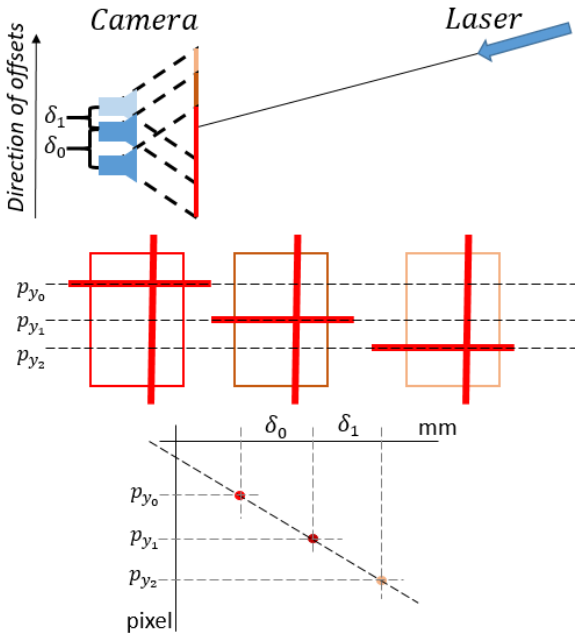


Figure 2.25: Illustration of camera chamber calibration.

Figure 2.26: Gauge blocks for camera chamber calibration.

chamber is invariant between captured images and minimal hardware is installed to capture images, the scale factor can be determined in post-processing in the last two steps of the above procedure.

In Step 3, we "induced" this vertical offset by using gauge blocks. As shown in Figure 2.26, these gauge blocks were machined precisely to a certain dimension, which enabled us to shift the camera chamber up or down by placing it on gauge blocks of different heights. Note: to avoid the propagation of small tilt effects, we did not change the location of the laser beam. In step five, it should be clear that the more photos taken, the more accurate the resulting calibration curve. For the actual results of calibrating the 3D printed camera chamber, as pictured in Figure G.4, please refer to Section 3.3.2.

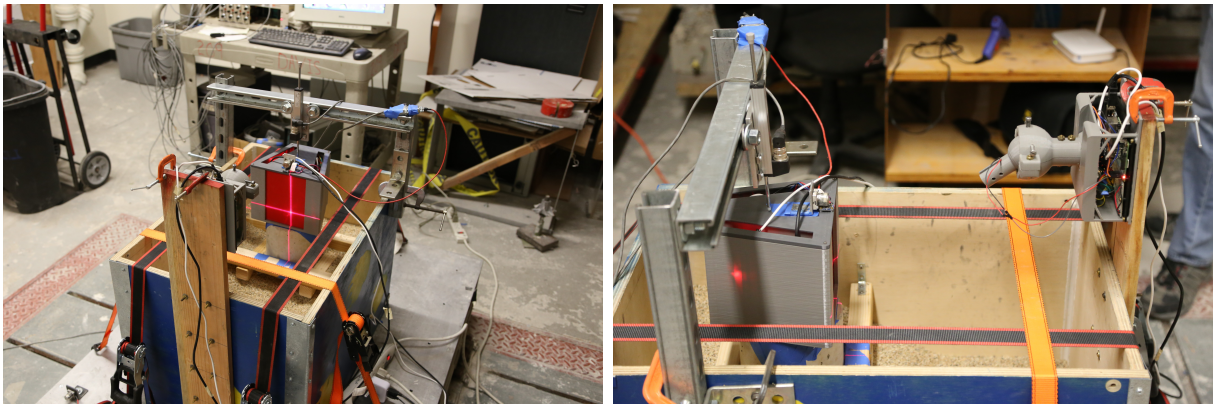
2.4.3 Experimental Setup

To evaluate the functionality of the monitoring system, we ran the monitoring system per the system architecture shown in Figure 2.1. This system was powered via a wall AC to DC adapter over the course of several days with various hard-coded time-synchronization and measurement interval times, ranging from taking a measurement every five minutes up to taking measurements

every 30 hrs. It was through tests of this nature that the two major issues discussed in the section on implementation challenges arose. Because Phase I was not intended to create a final prototype, we did not perform extensive testing and evaluation. The key issue here was to verify our calibration method, which enabled us to deploy our laboratory prototype on a shaking table test of a mock pile. This mock pile test was performed to evaluate the accuracy of the readings when taken out of the context of a carefully setup laboratory experiment, whereby the camera was ensured to not tilt, and the settlement was known to be induced in the vertical direction, only; it was never intended to evaluate the monitoring system.

We re-purposed a wooden box to accommodate several cubic feet of unsorted gravel and sand, which was then installed onto a shaking table. To reinforce this box, metal angles were installed on the edges of the box to ensure that the box itself would not fail when the sand was shaken; see Figure 2.27. In addition, a post was drilled into one side of the box so that the laser enclosure could be clamped to it. After this, gravel was gradually poured into the box from a low height in thin layers using metal scoops to ensure that the gravel was consolidated as little as possible during the loading of the box. Then, to represent a foundation pile, a 6-in. concrete cylinder was chosen and partially submerged into the gravel in the box. We secured the camera module to the top surface of the cylinder using heavy-duty tape. As a reference for the measured settlement, an LVDT was installed onto a frame that spanned across the top of the box and fixed the tip of the slider to the top of the camera module; see Figure 2.27.

The soil box was strapped down to the uniaxial shaking table using ratcheting tie-down straps, and the concrete cylinder was also kept in place using a wooden frame. This wooden frame was meant to limit how much the concrete cylinder would tilt. To measure how much the camera module would tilt, we recorded the accelerometer tilt readings before and after shaking.



(a) Overall view (b) Detailed side view

Figure 2.27: Phase I lab deployment, shaking table setup.

2.4.4 Results and Recommendations

To reset the box meant emptying the contents of the soil box and gradually pouring back the soil into the box layer by layer, a tedious task at best; therefore, not many trials were run and no statistically important trends could be extracted from the results of these tests. One observation

was very noteworthy: the wooden frame used to limit how much the concrete cylinder could tilt did not perform as intended, and the concrete cylinder still managed to rotate as the sand settled unevenly below it during shaking of the soil. This induced rotation seemed to result in appreciable errors in the measured settlements; see Table 2.1.

Table 2.1: Errors from the shaking table test.

Tilt (^{o})	δ_{LVDT} (mm)	δ_{cam} (mm)	Error (mm)	% Error
-0.34	-11.96	-12.37	-0.41	3.4
-4.70	-26.54	-38.89	-12.34	46.5
2.55	-9.17	-0.32	8.85	-96.5
0.71	-6.15	-2.69	3.45	-56.2

From the above table, we see that when tilt of the camera module is small (-0.34^o),with the error between the camera measurement and the reference LVDT measurement below 1 mm. This level of error is acceptable for the projected laser target method. When any significant tilt is introduced, however, this error is no longer sub-millimeter.

To this end, it became clear that the focus in the Phase II prototype would be to investigate various confounding variables that may influence the accuracy of our projected laser target method as listed below:

1. Influence of mounting distance between laser and camera modules on measurement accuracy: Is there a maximum range, and what is it dictated by?;
2. Effects of tilt of the camera module on measurement value: At what threshold does the contribution to perceived settlement from camera tilt become sub-millimeter?;
3. Influence of laser diode parameters on measurement accuracy: How does the focus and strength of the laser diode affect the accuracy of the measurement reading?; and
4. Ambient lighting influence on computer vision algorithm: Is the algorithm adequate for outdoor deployment? If not, how do we address this issue?

Ultimately, Phase I development demonstrated the validity of the projected laser target method but only under certain circumstances and in a laboratory setting. Phase I development also showed various limitations of the currently prescribed use case whereby the engineer must hard-code in synchronization and measurement intervals, and leave the unit alone after mounting it and booting it up. While the camera and laser modules would be able to synchronize relative to each other using PTP methods, the amount of clock drift could be reduced by tuning software timer parameters so that the units will together drift in absolute time. We determined that this lack of reliability in measurement was not satisfactory and must be accounted for in Phase II. In addition, the limitations of the chosen WLAN wireless network needed to be addressed. It is not feasible to mount a wireless router and power it continuously in a remote area; thus a more remotely deployable communication network must be investigated and utilized for Phase II.

3 Project Phase II: Field-Deployable System

The second phase of our system produced the final product for Caltrans that met all their requirements as detailed in Phase I. We again followed the same design flow as implemented in Phase I and in Section 2, beginning with a description of our use case.

3.1 SYSTEM ARCHITECTURE

3.1.1 Use Case Description

In this final iteration, a use case was determined that matched the basic requirements that the system be convenient to install and operate. In addition, we also considered that the system must be power-lean due to field-deployment constraints.

1. Upon installation and initial boot-up, begins in idle state;
2. The system, comprised of both laser and camera modules, stays in idle until a wake-up signal is received;
3. Wake-up signal transmitted from server and triggers computational units to commence measurement tasks; and
4. After measurement tasks are completed, both modules will return to idle states and wait for the next wake up signal.

Note that there is no longer any mention of time synchronization as the synchronized waking-up of both modules has been delegated to the server. The server, as programmed by the engineer, will now implement a custom measurement schedule. This frees the engineer from the restriction of the Phase I prototype for a fixed measurement interval, frees the embedded system from synchronizing the clocks of the two modules, and eliminates the issue of the entire system drifting with respect to absolute time.

3.1.2 Proposed System Architecture

To match such a use case, we proposed the system architecture illustrated in Figure 3.1. We note several departures from the system architecture of Phase I:

1. While the camera module still is comprised of a Raspberry Pi and Arduino, the laser module only has one Arduino;

2. Instead of all messages being passed through a wireless router, there is the addition of a GSM cellular module, which is attached to the camera module, and the addition of a Bluetooth module, which is attached to the laser module; and
3. An additional relay circuit is attached to the laser module.

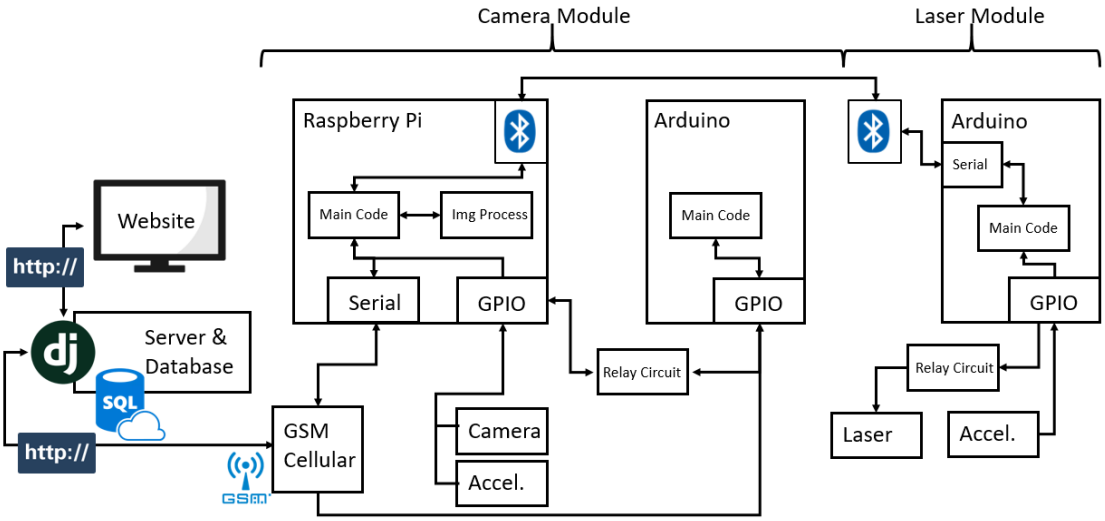


Figure 3.1: Phase II deployment: system architecture.

3.1.3 Communication

We will first discuss the communication layer of our computational unit, as this is perhaps the most radical alteration to the design implemented in Phase I, prompted by the constraints imposed by remote installation of this system. Considering that Caltrans may anticipate mounting this system along some fairly remote highways, we utilized a wireless network that has wide coverage across even remote parts of the state. We do not care about data speeds nor bandwidth, as the amount of data that we are sending is very small and not sent frequently (a few bytes of data sent every measurement interval). Considering common wireless networks, we settled on the 2G cellular network and chose a cell carrier that had reasonable coverage; T-Mobile (or at least a carrier Ting, which uses the same infrastructure as T-Mobile [13]). As seen in Figure 3.2, the T-Mobile 2G infrastructure nominally covers most of the major interstates and highways spanning California. Since T-Mobile uses the GSM protocol for data, we decided to install a 2G GSM cellular module, the Adafruit FONA (Mini Cellular GSM Breakout), as seen in Figure 3.3, which essentially is built around the SIM800H chip. To match such a use case, we proposed the system architecture illustrated in Figure 3.1. We note several departures from the system architecture of Phase I:

- While the camera module still is comprised of a Raspberry Pi and Arduino, the laser module only has one Arduino;
- Instead of all messages being passed through a wireless router, a GSM cellular module was added, which is attached to the camera module, and a Bluetooth module was added, which is attached to the laser module; and

- An additional relay circuit is attached to the laser module.

Next we discuss these aforementioned changes in the context of the framework adopted from Wang et al.: a computational core, sensor interface, and the communication layer [Wang et al. (2007)].

Figure 3.2: T-Mobile 2G coverage in California (orion.freeus.com).

The features of the Adafruit FONA align well with this application, which of course are all merely breakouts from the SIM800H chip. The SIM800H can be controlled with AT commands sent over serial [14]. Using these AT commands, the SIM800H chip can be put into a sleep mode during which it is in a low-power consumption state, yet it is still receptive to calls and SMS messages [15]. An incoming call or SMS message will also trigger another useful feature; a ring indicator. A pinout on the FONA board allows easy access to this feature in which the ring indicator signal will be driven into a low state temporarily from its typical high state whenever a text or call is received. Our system will employ this feature to "wake up" from its monitoring idle state. On top of all of this, it is entirely possible using AT commands to treat the SIM800H as the "heart" of a cellphone; one could potentially write a code that can perform basic cell-phone operations on the SIM800H (i.e., reading, sending, and deleting SMS messages). In addition, it would be capable of connecting to the Internet through a PPP (Point-to-Point Protocol) connection, thereby giving access to whatever online server we may want to push our data to. All that is required to contact the FONA through text or SMS is the phone number associated with its SIM card. Note: the FONA requires that a Li-Poly battery supply the main power to the chip. An additional power pin supplies the logic voltage, but a separate battery other than the primary battery used to power the entire module must be used to power the FONA board.

Now that the hardware necessary to implement the communication pathway between the camera module and the server/database has been selected (Figure 3.1), we now considered the pathway between the two modules and decided to deploy over a Bluetooth low-energy (BLE) connection. The BLE is a wireless personal area network (WPAN) that is different from traditional Bluetooth in that it consumes much less power. The BLE employs the master–slave framework in

Figure 3.3: Adafruit FONA mini-cellular GSM breakout.

which a central device (master) connects to many peripheral devices (slaves). When not connected, the peripheral devices beam advertisement packets and wait for a central device to connect to them. The central device listens for advertisements (in a so-called "discovery" mode). Once a connection has been made, the BLE devices use the Generic Attribute Profile (GATT), which essentially establishes a framework for the organization of information that Bluetooth devices can access [16]. Canonically, the central device will request information from the peripheral, which will then respond with this data; see Figure 3.4.

The peripheral contains all of the information concerning its specific GATT profile (see Figure 3.5), which allows for the flexible implementation of software of various applications for BLE, such as health monitoring, low-energy sensor reading, or in our case, structural monitoring.

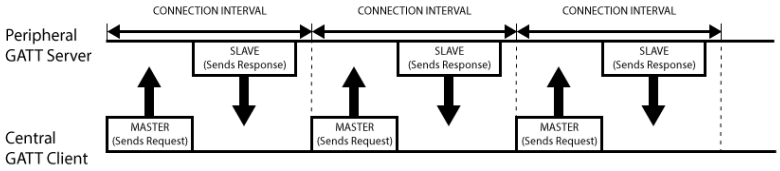


Figure 3.4: GATT client/server transactions, (adafruit.com).

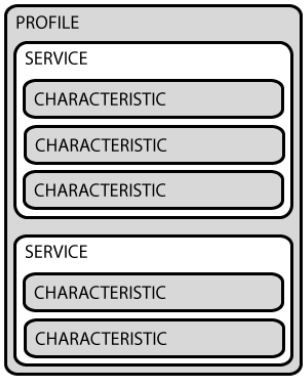


Figure 3.5: GATT profile framework, (adafruit.com).

We chose to implement BLE using the HM-10 BLE module (see Figure 3.6), which abstracts out many of the intricacies of BLE communication and provides two communication pin-outs—TX and RX—that can be interfaced with a serial port. Much like the Adafruit FONA, the HM-10 BLE

module is conveniently controllable through AT commands, which can be sent over serial lines (i.e., the TX & RX pinouts on the HM-10 breakout board). These commands provide the means of customizing the device name of the BLE device, which is useful for the master to correctly find the slave device. The HM-10 also employs its own special GATT profile, which only has one custom characteristic that can hold more than enough bits to pass the short messages and numbers that we anticipate to employ. Once interfaced through a serial connection, messages can be sent or read by writing to or reading from the BLE characteristic.

Of concern is the expected transmission range of BLE devices. Nominally, BLE devices can have an operating range of up to 100 m, but this is of course limited in actual deployment due to obstructions and ambient environmental factors [17]. It is also limited by the BLE power class of the specific transmitter/receiver. Class 1 devices have the highest range, whereas Class 2 devices have a range of 10 m. As expected, most consumer devices employ Class 2 devices due to their low-power usage and adequate range. For the application envisioned herein, we need this BLE pathway (which connects the camera and laser modules) to span across at least 20 m (around 65 ft, which is a reasonable span between mounting points on a highway overpass). In the case of the Raspberry Pi 3B+, the Cypress communication chip employs both Class 1 and 2 devices [18] and the HM-10 BLE module, though not specified specifically as a Class 1 device [19], was found in tests to have a range in excess of 60 ft in environments with unobstructed line of sight between the Raspberry Pi and the HM-10 module.

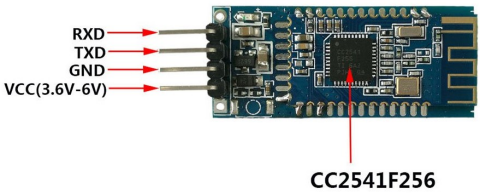


Figure 3.6: HM-10 Bluetooth low energy module.

3.1.4 Computational Core

The computational hardware was not changed as the computational requirements for the measurement tasks have not changed, and the new communication pathways (GSM and Bluetooth) are fully supported by the Raspberry Pi 3B+. The Arduino UNO Rev3 is more than sufficient for the actuating of a laser diode through its GPIO pins, the reading of a digital accelerometer through SPI (Serial Peripheral Interface), and the interfacing with a Bluetooth module through UART serial.

We eliminated the Raspberry Pi from the laser module because of the requirement that this system be "woken" from idle via a signal sent from the server to the cellular chip, which is only attached to the camera module. To be more precise, the cellular chip's ring indicator output is connected to the camera module's Arduino, which is perpetually on due to its low-power consumption compared to the cellular chip and the Raspberry Pi, and the cellular chip's data lines TX & RX are connected to the Raspberry Pi. The camera module's Arduino will receive the ring indicator and wake up its Pi. In order for the camera module to then connect to the laser module, the laser module must be perpetually on and advertising through BLE in order to be connected to by the camera module.

We cannot connect the ring indicator from the camera module to the laser module, as that would require a very long wire that would span the mounting distance between the two modules, nullifying the "wireless" nature of the system. Another option would be to attach a cellular chip to the laser module, but that would then require the server to call two phone numbers instead of one, thus increasing the complexity of the system. Even if this was done to ensure that the Arduino is not perpetually on, the Arduino does not consume enough power to be worth conserving. A lengthier discussion on system power consumption is made in Section 3.2.5.

3.1.5 Sensor Interface

The sensor interface has not changed from the camera, accelerometer, and laser diode suite presented in Phase I. To the micro-controllers, the HM-10 BLE and Adafruit GSM module can be perceived to be "sensors" in that the micro-controllers can interface with them through a data line in order to read and send commands; however, these are not considered to be part of the "sensor" interface as opposed to the accelerometer, camera, and laser.

3.2 STATE MACHINE DESIGN AND IMPLEMENTATION

We have described the various duties of each micro-controller and communication device in both the camera and laser modules; see Figure 3.7. Each of these duties will then be implemented fully in subsequent subsets of the block diagram in Figure 3.7. Note: the duties for the camera module's Raspberry Pi have not changed, and the duties of the laser module's Raspberry Pi in Phase I have been transferred to the Arduino in this version, with reference to Figure 2.13. A significant development when comparing Phase I to Phase II is the new set of signals that the camera module's Arduino must now read, $pin_{power}, pin_{ring}, pin_{batt}$, in which each signal represents:

1. pin_{power}: HIGH means Raspberry Pi is ON, LOW means Raspberry Pi is OFF;
2. pin_{ring}: HIGH means there is no incoming SMS or call to the cellular module, while LOW means there is; and
3. pin_{batt}: HIGH means the cellular module's battery charge is low, while LOW means the opposite.

Now that time synchronization is no longer an issue, the additions of the cellular module and BLE module allow for the wireless deployment of the system. Note: we do not directly program the communication devices shown in Figure 3.7, short of a few AT commands for coding in small tasks, such as renaming the device name of the HM-10, or erasing the SMS messages saved on the cellular module. The majority of the programming and state machine design are for the Raspberry Pi and Arduinos.

3.2.1 Communication Protocols

Once the system is deployed in the field, the communication protocols designed became more complex. The choice to use BLE as the link between the laser and camera modules was a key design shift from Phase I and required the use of an asynchronous communication code instead of synchronous communications. The difference between synchronous and asynchronous code

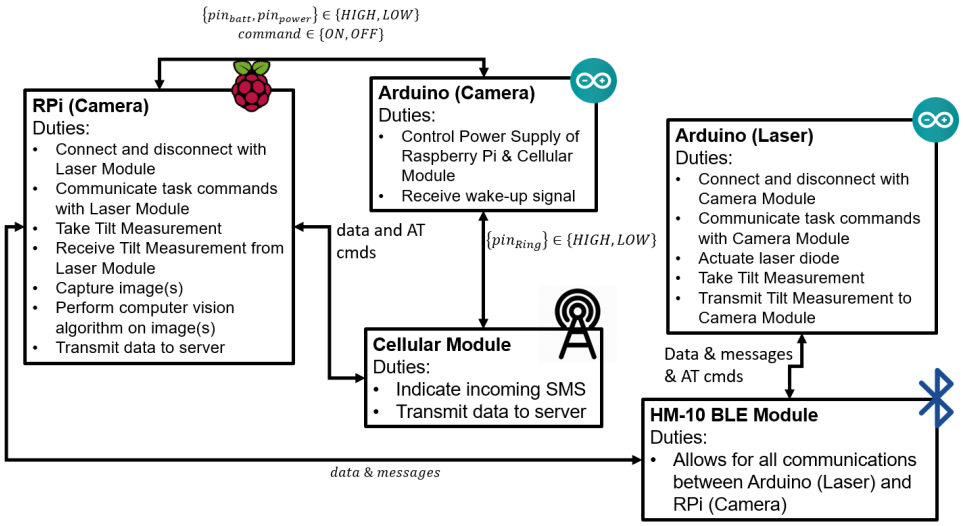


Figure 3.7: Phase II deployment: functional block diagram.

is as follows: (1) synchronous code is defined as a code that runs sequentially, in which one command follows another; and (2) asynchronous code is more flexible, in which many tasks could be happening at the same time [Lee and Seshia (2017)]. As expected, as the number of tasks that can occur at the same time increases and so does the difficulty in predicting the functionality and reliability of the code. Figure 3.8 is an example of how the two different kinds of programming that could be implemented.

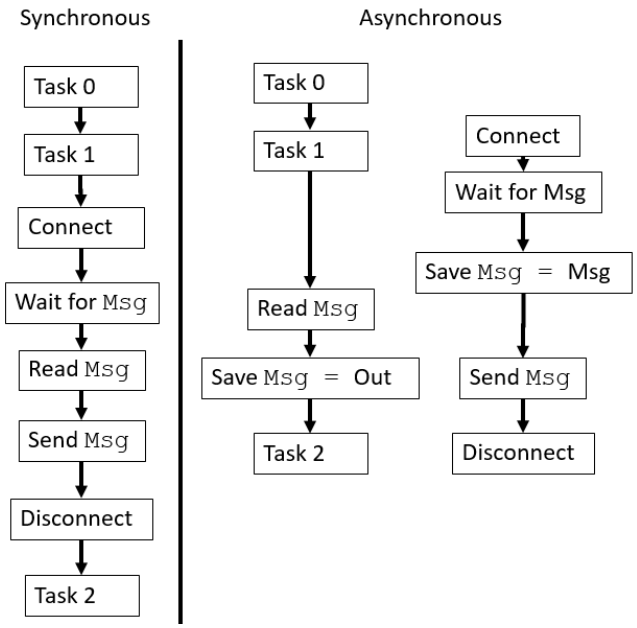


Figure 3.8: Synchronous v. asynchronous communication tasks.

Synchronous Code

In Phase I for any of the micro-controllers—specifically the Raspberry Pi—there is always one main program that implements the main state machine design and directs transitions from one state to another. By itself, this main program is synchronous in which each state (and at a finer level, each command) must be completed before another is executed; however, the code may become asynchronous depending on how the communication protocol is implemented programatically. In the case of the Phase I implementation, as is seen in Figure 2.14, there is indeed only one "thread" of execution whereby there is only one state that the machine can exist in at any point in time.

As shown in Figure 2.18, the communication protocol in Phase I was synchronous: each transaction of communication between the laser and camera modules began with the establishment of a TCP socket connection, which ensured that both modules were immediately available for message passing, which was followed by the message passing until the connection was cut. Each main code was essentially blocked (paused) until the communication transaction was completed; see the left side of Figure 3.8.

This communication protocol required that the mode chosen for communication be synchronous-compatible. For instance, in Phase I we employed TCP sockets, requiring that each socket (client and server) wait for the other until either a connection was made or a timeout was exceeded. This connection could be made quickly, followed by a quick transaction of messages and then disconnection. Due to the ease of simply re-establishing the connection, we were able to implement the communications synchronously, which melded easily into the uni-threaded main program.

Asynchronous Code

For Phase II, it was decided to implement BLE, which requires careful consideration of how both the central and peripheral devices handle both incoming and outgoing sending messages. For convenience, we chose to utilize a Python module that implements BLE communication: `Bluepy`. The module performs the scanning, connecting, and sending/writing of messages via the BLE characteristics, all specifically for the central device. Note: we designated our Raspberry Pi on the camera module as the central device and the Arduino as the peripheral device, thus using the BLE communication protocol fits our hardware and software paradigm.

`Bluepy` operates by instantiating various objects, each of which has various methods. For instance, one can first use this command to "scan" an object, which is able to do just that: scan for nearby objects. Once a nearby device is found, that device will have been associated with a Bluetooth address, allowing the user to create a "Peripheral" object out of the device, thereby implicitly connecting to the object. At this point, the user can write to the Peripheral's characteristic (i.e., writing it a message) or read from it.

Now the issue of synchronicity comes into play. `Bluepy` specifically handles incoming messages by throwing notification events. These events act very much like the interrupts that were discussed in Phase I in that they are meant to alert the attention of the invoking function immediately and then be handled. In the case of `Bluepy`, the invoking function needs to be constantly monitoring the peripheral object for notifications; if a notification is not handled immediately, the incoming data that is associated with the notification is lost.

This required reconsidering using synchronous communication for our prototype. We can no longer have our main code "blocked" until a message arrives. Immediately, the issue of deadlock arises, in which one module may be blocked indefinitely waiting for a message that will never arrive. For instance, this may arise if the invoking function begins monitoring after the message in question has already thrown a notification; the invoking function will search indefinitely or until a timeout is exceeded. To accommodate this issue required employing asynchronous communications; see the right side of Figure 3.8. we use multiple threads to implement the entire system: one thread to implement the state machine itself and other threads to read and write messages. The multi-threaded code used to implement this asynchronous communication design is discussed next.

Asynchronous BLE Communication

We implemented the asynchronous BLE communication required due to the functioning of `Bluepy` and the architecture of BLE through the means illustrated in the functional block diagram in Figure G.8, in Appendix G. The critical aspects of the diagram are discussed below.

First and foremost, this diagram is divided into three main columns: (left) main program (or thread), (center) `ble_worker` thread, which checks and reads incoming messages, and (right) `ble_write` thread, which transmits messages. Each thread represents an independent sequence of execution, but the whole diagram represents implementing these three threads in parallel (i.e., multi-threading). The diagram shows a scenario where the main program (or main thread) performs measurement tasks, followed by the reading of a message, then sending a message, performing more tasks, and then disconnecting from the peripheral. Whenever multiple threads perform tasks that must both read, write, or otherwise share the same resources, extra care must be taken to protect the use of those resources. In this case, the three threads collectively share various global variables: `SETTINGS.DATA` (which holds the received message), `SETTINGS.NEW_MSG` (a flag which indicates if the data in `SETTINGS.DATA` is a new message), `SETTINGS.WRITE` (which holds an outgoing message), as well as the BLE peripheral object itself, which can only be used in one instance at all times. The BLE peripheral object can of course be called by multiple threads, but only by one thread at a time.

To protect and ensure that these threads function properly, we employed various locks and events. Locks are the simplest means of protecting memory resources. If a lock is placed before a section of code, the only way for a thread to access lines of code following that lock are if it acquires that lock. Once a lock is obtained, the lock holder must release the lock before others can acquire it, as only one thread can hold a lock at any time. A thread may either block trying to acquire a lock, or not-block, and merely check/wait for the lock until a timeout. An event is slightly more complex, in that a thread may wait on an event to be set in order to move onto subsequent lines of code. This thread may, as with the lock, exhibit either blocking or non-blocking behavior while waiting for an event to be set. Regardless, when the event is set by the event owner, the thread waiting on the thread can immediately pass through the event and move onto its next commands.

Note in Figure G.8 which locks and events protect access to which resources in the **Initialization** block. Upon the initialization of threads, the main thread will eventually anticipate an incoming message and will attempt to read that message. It does so by attempting to access

SETTINGS.DATA, protected by lock_0. In the gray box on the left side of the diagram, you can see that the thread will repeatedly try to acquire lock_0. The other thread, which is concurrently using lock_0 is ble_worker, which uses it to allow access to SETTINGS.DATA for the Bluepy notification handler to save any incoming messages. If ble_worker is not holding lock_0, the main thread may access SETTINGS.DATA and SETTINGS.NEW_MSG. If SETTINGS.NEW_MSG == 1, it means that the message in SETTINGS.DATA is a newly received message from the peripheral device.

In ble_worker, lock_1 restricts access to the peripheral object, as its methods are required not only in ble_worker, but also ble_write. ble_worker requires the method wait_for_notifications to monitor the peripheral object for notifications, and ble_write requires write to write messages to the BLE characteristic. Both of these threads are, therefore, constantly attempting to acquire lock_1 in a non-blocking manner.

Finally, it should be noted that all three threads are daemon threads, meaning that their invoking programs can terminate without waiting for them to terminate. This is easier in our case to implement as we do not have to ensure that ble_worker and ble_write have been killed before the main thread can terminate. That being said, just to clear up any resources that may be required in subsequent measurement tasks, we can choose to terminate ble_worker by setting event_1 from the main thread, which will cause ble_worker to break its reading loop, and disconnect from the BLE peripheral. When that disconnect occurs, ble_write will also terminate due to the de-allocation of the Bluepy peripheral object. Even if any of the two child threads were to throw any errors, they would not be fatal to the operation of the main thread. One detail not included in the diagram is the slight offset in initialization of the two threads; because there are slight delays interspersed in each thread's loops and these delays are so small, there is a very small chance of the threads entering a state where none of them are able to acquire a certain lock due to an alignment of lock release-and-acquire phases.

3.2.2 Camera Module Raspberry Pi

With the communication protocol designed, we can now interleave it into the rest of the main program, or in other words, the state machine for the Raspberry Pi on the camera module. We include a diagram of this state machine in Figure G.9 (in Appendix G). This is the same sort of structure as was seen in the Phase I prototype shown in Figure 2.14. Discounting the time synchronization option, most of the same essential states in the main program are still in place, all programmed to accomplish the duties listed in Figure 3.7. Note the two image states, with an additional state **Image I Comm** to command RPI_{Laser} to turn off the laser beam. In addition, there are two additional threads introduced to implement the BLE communication. One of the key elements we included in this modified design is how the system handles errors and ensures that no data is lost in the case of said errors.

Error Handling and Data Loss Prevention

It cannot be guaranteed that the FONA cellular module will always function properly. Hardware failures do occur. The same can be said for any piece of hardware chosen for this monitoring system. Given this eventuality, it was deemed prudent to ensure that if errors do arise, the field-

deployed monitoring system does not fail, thus requiring lengthy hands-on debugging to resolve the issue. To examine how our implemented code ensures this, we will examine a snippet of code from the main program:

Listing 3.1: Excerpt of `RPI_C.py` main program

```
elif state == "IMAGE_I_COMM":

    try: #try this entire "try" block of code
        #Send out command to Laser Module
        SETTINGS.WRITE = "TURN_OFF_LASER?";
        ble_event.set();
        time.sleep(1);
        ble_event.clear();

        #Expect to receive a message back
        msg = wait_for_msg(locks);
        if msg != None: #if a message is received
            if msg == "LASER_OFF": #if the message is as expected
                state = "IMAGE_II"; #transition to next state
            else:  #if the message isn't as expected
                logging.debug("No Laser Off Ack Error");
                errors.append(12);
                error_flag = True;
                state = "SHUTDOWN"; #transition to SHUTDOWN
        else: #if a message isn't received
            logging.debug("RPI_L Communication Error");
            errors.append(13);
            error_flag = True;
            state = "SHUTDOWN"; #transition to SHUTDOWN
        msg = None;
    except: #if a much more general error occurs
        logging.debug("IMAGE_I_COMM: ERROR");
        errors.append(33);
        error_flag = True;
        state = "SHUTDOWN"; #transition to SHUTDOWN

elif state == "IMAGE_II":
    try: #try to take an image
        img_collector = ImgCollector();
        img_collector.capture();
        img_collector.shutdown();
        state = "L_ACCEL"; #transition to next state
    except: #if image capturing failed
        logging.debug("IMAGE_II ERROR");
        errors.append(3);
```

```
error_flag = True;
state = "SHUTDOWN"; #transition to SHUTDOWN
```

The above code implements two states; see Figure G.9: **IMAGE_1_COMM** and **IMAGE_2**. These are, respectively, the state in which the command is sent to the laser module to turn off the laser beam, and the state in which the second image is taken, i.e., representing a state dedicated to communication and another dedicated to hardware interfacing.

Note that the main code differentiates between states by comparing a global variable `state` that contains the current state against pre-defined state names. Within each state "block" of code, we implemented various nested try-except code blocks, which are a useful code structure provided by Python to implement error handling. Essentially, the program will "try" the try-block of code; if an error occurs in the try-block, the entire main program does not fail. Instead, the main program flow will be diverted to the except block. In addition, within the try-block, we can implement if–else statements to catch any unanticipated non-fatal errors, such as whether or not a received message is what we expected, as seen in Line 15. This is a non-fatal error, as compared to perhaps line 34, in which the camera object is initialized. This camera object interfaces with the hardware, so if the hardware has any issues, Line 34 would most likely throw a fatal error. As such, we can program the except-and-else blocks to contain error handling functionalities. Here, we enumerated the various errors that we anticipated based on our partitioning of the code into these try-except blocks. Each numerical error code is appended onto a global list variable, `errors`, so that by the time the main thread has reached the **Server** state in which data is transmitted to the server, we can also send any encountered error codes to the server.

What if the transmission of data to the server fails? What if the cellular connection cannot be made? The prevention of data loss in the event of transmission failure was handled by means illustrated in the block diagram in Figures 3.9 and 3.10. Note that in the context of error handling, error codes are also considered data. Any "data" that we send to the server is organized in fields in the format:

```
data = [run_count, p_x, p_y, pitch_camera, pitch_laser, [errors]],
```

such that we will always attribute each set of measurement results and errors to a given run count, so that the server can easily organize the received information according to which run the data was generated. This would also improve any off-site debugging that may occur.

The Raspberry Pi using the cellular module initializes the cell modem on the local end first and logs in the cellular module, after which the cellular modem negotiates a connection with the far end (perhaps one of the nearby cell towers). The far end, prompted by a call from the cell modem, sends back routing information, which is necessary to establish the complete PPP connection. These negotiations and initializations are implemented on the Raspberry Pi via two files: a "chat script," which consists of preset responses to anticipated far-end messages, as well as a connection configuration file that contains information on the type of PPP connection desired. If a connection is made and verified using a TCP socket connection to `google.com`, the current `data` is transmitted to the server, after which the Pi will access a file `data_log.txt`, in which previous un-transmitted data is stored. If this file is not empty, each line (of the form specified by

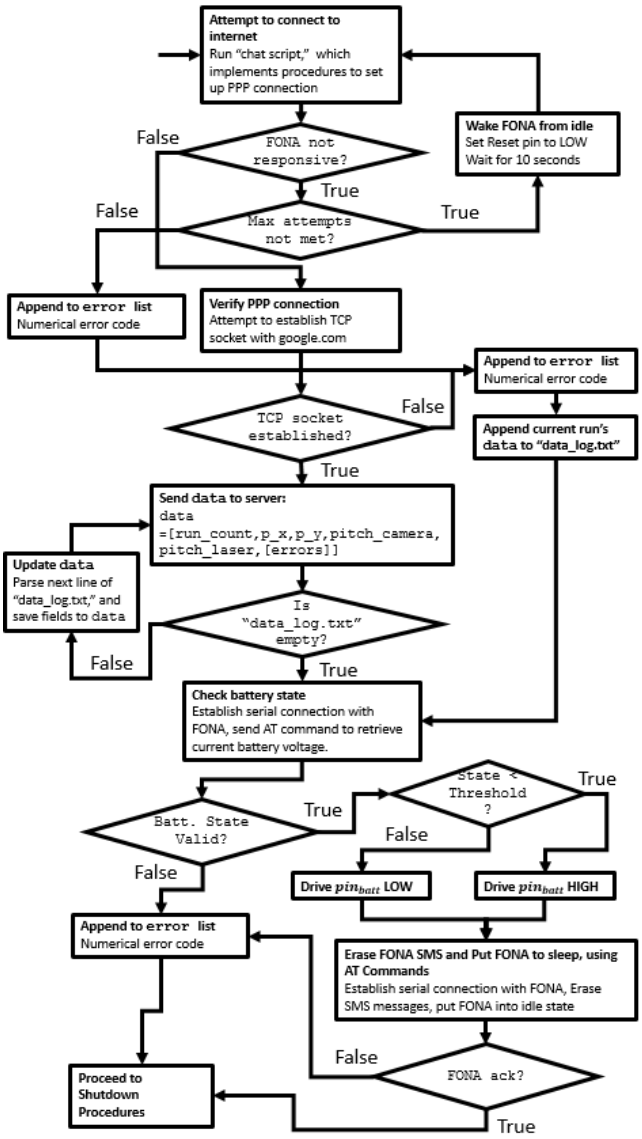


Figure 3.9: Phase II deployment: Raspberry Pi camera module `Server` state diagram.

`data` previously) is transmitted sequentially through the current Internet connection.

Of course, the PPP connection may not work in one shot if the cellular coverage is weak or if the FONA cellular module itself is not responding (because it is in idle mode). Should either of these events occur, the Pi will reset the cellular module and try to connect to the Internet once more. If a sequence of these attempts does not work, the Pi will deem the connection to be futile and append a numerical error code to `errors` (as it does with any other encountered error in the program execution). Note: there still exists tasks that must be performed by the Pi before it shuts down completely, such as checking the battery state of the cellular module and putting the cellular module into its idle state. These tasks themselves may throw errors even as the current `data` still has not been dealt with and are appended to by the main thread. We now address the **Shutdown** state.

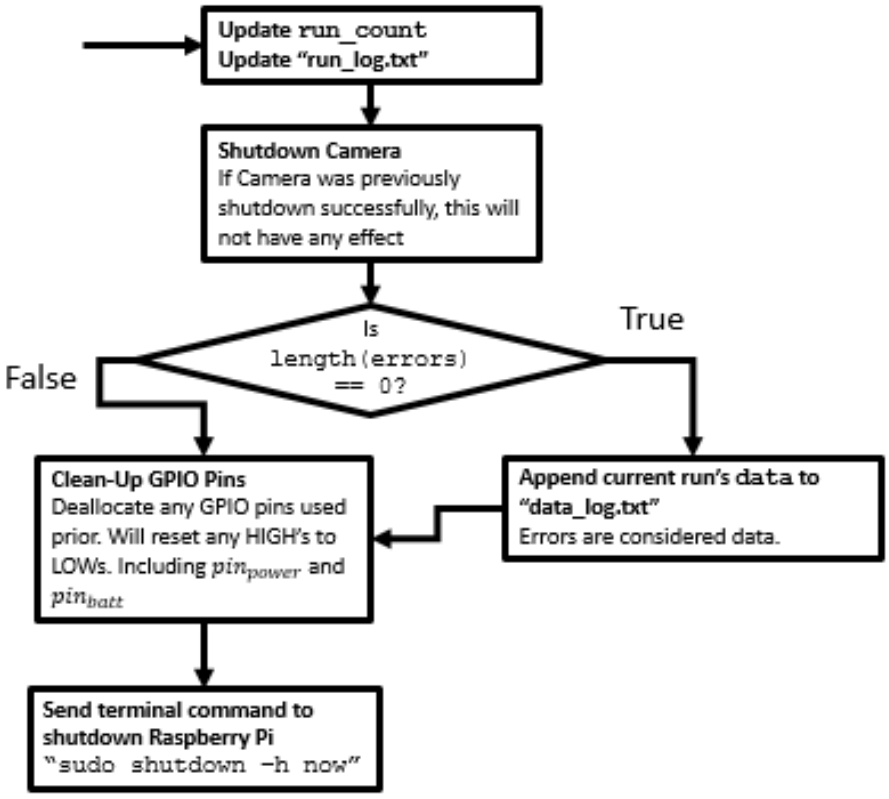


Figure 3.10: Phase II Deployment: Raspberry Pi camera module Shutdown state diagram.

In this final state, we see that one of the last actions done by the Pi before final shutdown is appending the current run's `data` to the text file. This ensures that any data or errors that may not have been sent (including those accumulated since data transmission) will be saved for transmission during the next time a cellular connection is made. The only action whose potential errors cannot be accounted for is the de-allocation of GPIO pins. This task is not expected to ever return an error. If it does, the de-allocation of GPIO pins is performed inherently (albeit not cleanly) by the rebooting of the Raspberry Pi, which occurs in between measurement runs. Therefore, this mechanism ensures that all data that is not transmitted is retained for a future transmission.

As far as handling errors and ensuring that data is retained for subsequent transmission, there are situations where an error may occur in a measurement-critical state whereby the tasks in that state are essential to a successful measurement. For instance, in **Initialization I**, if the camera module cannot even connect over BLE to the laser module, there is no possibility for the correct images to be taken to extract the position of the current laser beam, `[p_x,p_y]`. Therefore, if an error is caught in this state, the error code is saved in `errors`, and the main thread transitions to **Shutdown** where `errors` is saved to `data_log.txt`. The same behavior is implemented in other critical states, as illustrated in red in Figure G.9. We also tabulate the various possible error codes in Appendix G, Table **??**, which also indicates which errors are considered "critical."

3.2.3 Camera Module Arduino

We now discuss the state machine design and implementation of the camera module's Arduino, which must read in signals from the Pi and cellular module to control the power delivery to the Raspberry Pi and cellular module battery. The function block diagram is included in Figure 3.11.

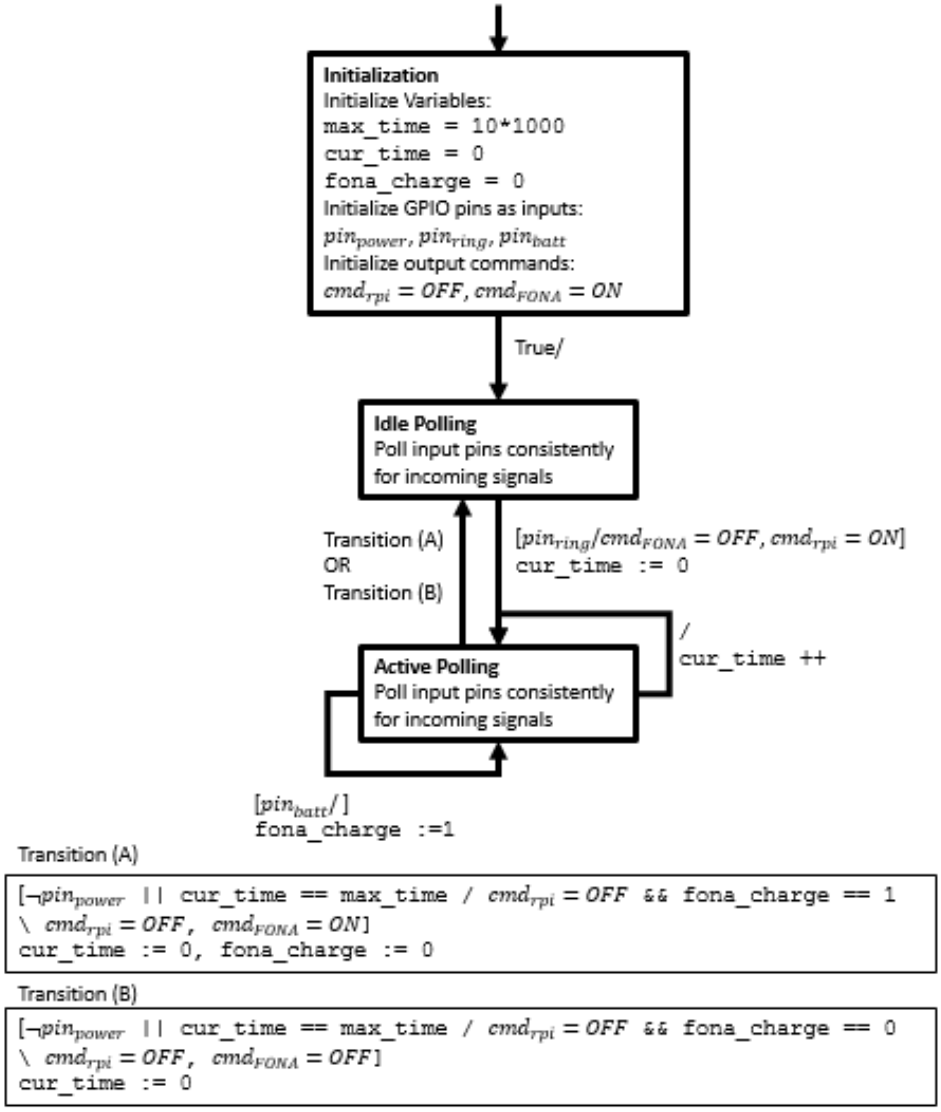


Figure 3.11: Phase II deployment: Arduino camera module state diagram.

In Figure 3.11, note that the Arduino essentially resides in one of two states, **Idle Polling** and **Active Polling**, which are differentiated by whether or not the Raspberry Pi does not have power delivered to it or if it does, respectively. This separation ensures that unexpected signals do not cause aberrant response from the Arduino. These signals can be one of three: pin_{power}, pin_{ring}, and pin_{batt}, which are HIGH when the Raspberry Pi is on, LOW when the FONA receives an SMS or call, and HIGH when the FONA battery is low and needs to be charged, respectively. Their opposite logic states imply the opposite meaning, respectively.

In state **Idle Polling**, the main thread is not incrementing a counter, but constantly polling the Arduino's GPIO pins checking for the incoming signals. The moment that a ring signal is received, it shuts off the power line to the FONA battery ($cmd_{FONA} = OFF$) and then closes the switch on the line, delivering power to the Raspberry Pi ($cmd_{rpi} = ON$). This ensures that the current consumption while the Raspberry Pi is on is below the total current output supplied by the battery (more on this in a subsequent section). While the program is in **Active Polling**, a counter is incremented. If at any point pin_{batt} is received during this state, it means that the Pi has queried the FONA for its battery level and found that it needs to be charged. The Arduino therefore flags this and continues **Active Polling**. When the Pi shuts down, a $\neg pin_{power}$ tells the Arduino to cut off power to the Pi and transition back to **Idle Polling**. If the Pi, for some extreme reason, never shuts down (or at least the signal to the Arduino signifying its occurrence is never received), the Arduino will automatically shut off the power line when a preset timer threshold has been met by the counter. Either way, when the machine transitions back to **Idle Polling**, the Arduino also checks to see if the FONA battery flag has been activated. If it has, it means that the Arduino will shut off the power line to the Pi but turn on the power line to the FONA battery. If it has not, the Arduino will shut off both the power lines to the Pi and FONA battery. As a side note, this behavior means that the FONA battery must have enough battery capacity to power the FONA for potentially several measurement intervals, depending on how frequent measurements are taken. This will be discussed in more detail in a subsequent section.

3.2.4 Laser Module

We can now discuss the state machine design of the Arduino in the laser module, which is shown in Figure G.10 in Appendix G. Note that this diagram is incredibly similar to the synchronous implementation of the Phase I deployment of the laser module, as illustrated in Figure 2.14(b), disregarding the time-synchronization execution branch. This synchronous implementation is possible due to the BLE interface between the HM-10 BLE module and the Arduino.

Recall that on the Pi, the chosen Python module `Bluepy` accommodated incoming messages through interrupt-style notifications, which had to be handled immediately upon arrival. This necessitated multi-threaded communication implementation. On the Arduino, the main thread initializes a serial connection with the HM-10 device over its UART TX-RX lines. By doing so, it sets up a buffer, which can store a finite amount of information at any point and conveniently stores any incoming messages that are sent from the central device to the peripheral. That way, the main thread needs not to persistently monitor the buffer for messages; incoming messages are automatically saved into the buffer.

The additional states **Wait I** and **Wait II** are introduced so that the Arduino may wait a finite time for the Raspberry Pi to (1) take the picture of the laser beam, and (2) take the ambient conditions photo, as required for the computer vision algorithm. The machine may exit these states either when a timeout is met or if the correct command is received from Raspberry Pi (`TURN_OFF_LASER`, and `TURN_OFF`, respectively). These commands are sent by the Pi in its states **Image_1_Comm** and **Laser_Acceleration**, respectively.

Note that each state in Figure G.10 is deemed critical. The error catching in this state machine is very minimal as Arduino's programming language is based in C and does not implement try-

except behavior. In practice, implementing try-except behavior is a computationally expensive operation, requiring the program stack to be copied for each try-except block, with a try-stack block being discarded if it errors and the program diverting to the corresponding except-stack block [20]. There does not exist the on-board memory nor programming resources for this to be implemented on the Arduino.

That being said, we have made the code for the laser module's Arduino as simple as possible, see Appendix H, with many of the states' transitions dictated by timeouts tied to communication transactions. The expected result is that if any state times out or runs into unexpected errors, the machine will eventually revert back into its `Idle` state. In the absolute worst-case scenario, the Arduino would no longer be responsive to connection responses from the camera module; the user would be able to discern that based on error codes transmitted from the camera module. Fixing this would require a technician to visit the module in person and reset the board.

3.2.5 Power System

Power Delivery System

The breadboard schematic of the circuit that implements the power delivery system is included in Appendix G and Figures G.11 and G.12 for the camera module and the laser module, respectively. The power delivery system is notably different between the two modules in this prototype as the system architecture and power demands of either module are very different from each other.

We first consider the power system for the camera module. There are two relays: one for controlling power to the Pi and the other for controlling power to the FONA battery. The two relays are respectively controlled by two GPIO pins, each from the Arduino, the green and brown lines, and blue and orange lines, in that order. The two relays transmit their respective power out to their sinks, represented by the barrel plugs on the left. Also, there are three lines for carrying the signals pin_{power} (yellow wire), pin_{ring} (white wire), and pin_{batt} (purple wire), each of which are pulled up or down by a $5k\Omega$ resistor. In particular, the pin_{ring} line is pulled HIGH instead of LOW as the FONA cellular module actually outputs a HIGH voltage when there is no ring and a LOW voltage when there is a ring. The other two signals are pulled low.

Another noteworthy item on this diagram is the wire connecting GPIO07 on the Raspberry Pi to the reset pin on the FONA cellular module. This line exists so that the Pi can drive the reset pin in order to reset the cellular module to bring it out of idle mode.

Note in Figure G.12 that the power delivery system for the laser module exists purely just to deliver power to the laser diode. This is because the 100 mW laser diode requires about 20 mA of current at 5 VDC, which is greater than what we should draw out of a GPIO pin on the Arduino. The reason why we did not implement such a system in Phase I was that we were actuating a 5 mW laser, which requires four times less current than the 100 mW laser. The other devices in the diagram are the HM-10 BLE module, which is connected to the UART pins on the Arduino, and the accelerometer, which is connected to the SPI pins on the Arduino.

Power Consumption

Now, we consider another key component of this field-deployable system: how much power it consumes. For the system architecture illustrated in Figure 3.1, we were able to empirically measure how much power various components consume. We did this using a USB power meter, like the one shown in Figure 3.12, which plugs into any USB adapter, be it in a battery or from a wall adapter, and measures how much voltage, current, and power is being consumed at any time. The voltage, current, and power consumption of both modules were measured at various "modes" of their operation [21]. These modes can be best described in terms of how we expect the modules to operate when deployed, as listed below:

1. Both modules are connected to their battery power supplies through the power delivery circuit;
2. The laser module's Arduino receives power and is on for perpetuity;
3. The camera module's FONA cellular module is delivered power, but the Pi is not;
4. When an SMS is sent to the camera module's FONA, the power will cut off from the FONA, and delivered to the Pi;
5. The camera module will perform its measurement tasks and shut off;
6. If the FONA is below charge, the Arduino will cut off power from the Pi and deliver power to the FONA;
7. If the FONA is well charged, the Arduino will cut off power from both the Pi and FONA: and
8. The camera module will now wait for the next SMS.

Following the sequence of actions listed above, we measured the power consumption at various points in the sequence; the results of which are tabulated in Table 3.1. Note that many consumption readings were made, and the values tabulated are conservative (overestimates). Considering the power consumption listed in Table 3.1, we chose the V44 USB Battery Pack sold by Voltaic Systems, which has a 12,000 mAh capacity and can be charged by a myriad of solar panels; see Figure 3.13. The solar panel chosen is another Voltaic product, a 10 Watt USB Solar Charger, which outputs 5 VDC current at varying currents depending on cloud cover and how aligned the panel is with the sunlight. If it is overcast, the panel may output less than 0.2 A. If it is clear, and the panel is directed straight at the sun, the panel can output in excess of 1.5 A. On a clear day, with the panel not directed at the sun, it may output about 1.1 A. For powering the camera module, we connected two solar panels in parallel to double the total solar power generation (current-wise).

Before we can confidently say that the camera module can run on its power supply indefinitely, we should consider the FONA battery, which is required by the FONA's manufacturers to deliver enough current to supply for apparent current spikes that the FONA sometimes experiences [22]. The FONA, when idling and is receptive to incoming calls and texts, depreciates the voltage of the battery in a nonlinear fashion. The FONA requires a minimum charge of 3.4 V to operate [23], and the battery at full charge supplies about 4.0 V. We ran the FONA on idle and checked on

Table 3.1: Power consumption at various modes of operation.

Description of Mode	Component Status			V [V]	I [A]	P [W]
Camera Module	Pi On	FONA**	Ard. On			
FONA charging, Pi Off	No	Yes	Yes	5.02	0.59	2.96
Pi On & program running	Yes	No	Yes	5.03	0.62*	3.05*
Pi On & program not run.	Yes	No	Yes	5.01	0.54	2.75
Pi Off & FONA not charging	No	No	Yes	5.06	0.07	0.35
Laser Module	Laser On		Ard. On			
Arduino Only	No		Yes	5.06	0.06	0.30
Arduino & Laser	Yes		Yes	5.06	0.26	1.30

*can be as low as 0.5 A, 2.45 W

**Yes means that the FONA is being charged

Figure 3.12: USB power meter.

(a) Voltaic Systems 10 W solar panel

(b) Voltaic Systems 12,000 mAh USB battery pack

Figure 3.13: Solar panel and battery pack.

its battery state every few hours. The FONA battery's charge level dropped from 4.0 V to 3.8 V in about 24 hrs of inactivity. Then, after about 6 hrs, the charge dropped to 3.45 V, which is very nearly at the minimum charge to power the FONA. It then took a mere 2 hrs to charge the FONA battery back up to full charge using the V44 battery pack. As designed, our camera module will only determine whether or not to charge the FONA battery once every measurement cycle, based on a threshold of 3.8 V. In the worst case scenario, if the battery charge were just above 3.8 V at the end of one measurement cycle, the FONA battery would not have enough charge to power the FONA for another 24 hs to receive the next measurement request from the server.

This necessitated increasing the frequency of measurements taken per day so that we can

check on the FONA battery charge state. Considering the FONA battery depreciates from 3.8 volts to 3.45 volts in 6 hrs, our threshold at 3.8 volts required checking the FONA battery every 6 hrs. Assuming we begin with a fully charged FONA battery (which is a safe assumption as it takes approximately 2 hrs for the battery to fully charge, and the first measurement is not taken within 2 hrs of installation and initial power-up), this should ensure that our FONA battery is never at risk of losing charge below 3.4 V. Note: shortening the measurement period to 6 hrs from 24 hrs had no significant impact on the total camera module power consumption as the Pi is awake only a maximum of ten minutes at a time, which in itself is a very conservative estimate of program run time (the camera module program run time, timed several times, is typically four minutes long). Therefore, we can now calculate some estimates of camera module lifetime based on power consumption and measurement frequency.

Referring to Figure 3.14, we can visualize how we determined the battery capacity of our camera module. Assuming six-hour measurement intervals, program run time of ten minutes, a bi-linear battery voltage discharge rate (Figure 3.14), and a minimum charge threshold of 3.8 V for the FONA battery, we can project power usage across three days; see Figure 3.14.

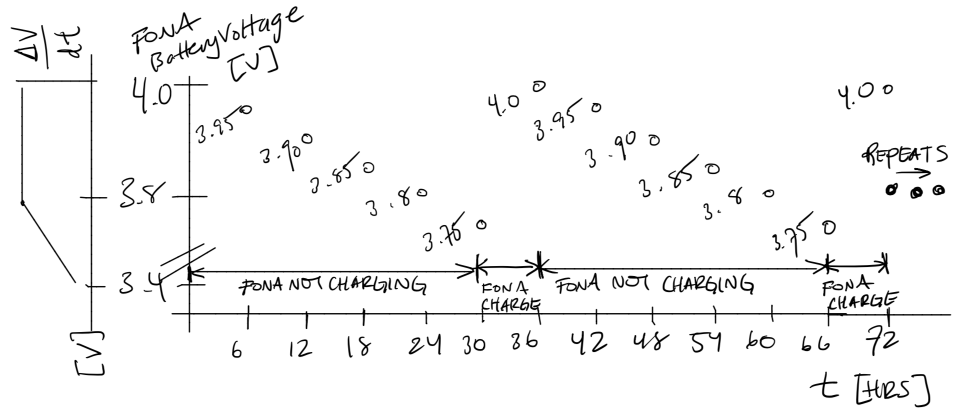


Figure 3.14: Three-day power consumption projection for battery capacity specification.

We have assumed that at hours 24 and 60, the Raspberry Pi reads that the FONA battery voltage is just above the 3.8 V threshold: therefore, the Pi deems the battery charge sufficient to last for another 6 hrs. With this assumption, it is noteworthy to see that the pattern of FONA charge periods and FONA not-charge periods repeats after three days. Therefore, we can analyze the power usage history across three days to determine if the power generated by the solar panels will supply enough power to satisfy the general power demand. First, we must calculate the power demand $E_{3\ Days}$:

$$E_{RPI} + E_{FONA\ BATT} + E_{Arduino} = E_{3\ Days} \tag{3.1}$$

From Figure 3.14, we can see that the Raspberry Pi is on $T_{RPI} = 12 * \frac{10\ min}{60\ min} = 2\ hrs$, the FONA battery will be delivered power for charging a total of $T_{FONA\ BATT} = 2 * (6\ hrs - \frac{10\ mins}{60\ mins}) = 11.67\ hrs$, and the Arduino will be exclusively on for the remaining $T_{Arduino} = 72 - T_{RPI} - T_{FONA\ BATT} = 58.33\ hrs$.

$$E_{3\ Days} = 0.62\ A * T_{RPI} + 0.59\ A * T_{FONA} + 0.07\ A * T_{Arduino} = 12.21\ Ah \tag{3.2}$$

If we assume that the average day has about 5 hrs of peak sun [24] and that each of our two solar panels outputs a very conservative 0.5 amps during peak-sun hours, we can expect E_{solar} over three days to equal:

$$E_{solar} = 2\ panels * 0.5\ A * 15\ hrs = 15Ah \tag{3.3}$$

We see that $E_{solar} > E_{3\ Days}$ even when assuming a very conservative solar panel energy generation and over estimating the program's run time and component power consumption. It is also assumed that the FONA battery will be consuming 0.59 amps over the entire six-hour interval it is delivered power for charging. It may very well be the case that after the battery is fully charged (two hours according to empirical observations), the current consumption will drop off.

Note that we are merely comparing the expected power consumption versus the expected power demand. We are disregarding the fact that we are employing a 12-Ah battery to supply the camera module. This battery acts as a *buffer* so that when there is no power generation (say, at night), or when power generation is sub-optimal (say, when the weather is overcast), the system will still operate on its battery reserve. That being said, we can calculate that from our three-day power consumption projection, that the average current consumption was:

$$A_{avg} = \frac{12.21Ah}{3\ days} * \frac{1\ day}{24\ hrs} = 0.1695\ A \tag{3.4}$$

Therefore, our camera module to run on a V44 12 Ah battery pack for:

$$\frac{12\ Ah}{A_{avg}} = 70.80\ hrs \approx 2.9\ days \tag{3.5}$$

Thus, the camera module is capable of running for nearly three days on reserve power if no power is generated by the solar panel.

Be aware that we have performed the above power consumption calculations on merely the camera module. The laser module, referencing Table 3.1, has significantly less power consumption. Over a single day, assuming a measurement interval of 6 hrs, we anticipate the Arduino and laser to be on together at most $T_{laser} = 4 * (\frac{10\ mins}{60\ mins} = 0.67\ hrs$, and only the Arduino to be on for the remaining $T_{Arduino} = 24 - T_{laser} = 23.33\ hrs$. The expected energy consumption per day E_{day} is therefore:

$$E_{day} = 0.26 * T_{laser} + 0.06 * T_{Arduino} = 1.573\ Ah \tag{3.6}$$

The expected solar power generation in one day, assuming even just 4 hrs of peak sun hours with 0.5 amps of current output using one solar panel, is $E_{solar} = 2\ Ah$ per day. Clearly, $E_{solar} > E_{day}$, i.e., we can clearly sustain the laser module using the one solar panel. In the case of no solar power generation, we can normalize E_{day} across 24 hrs ($\frac{1.573\ Ah}{24\ hrs} = 0.0655A$) and calculate that the laser module can survive on reserve power for $\frac{12\ Ah}{0.0655\ A} = 183.1\ hrs = 7.6\ days$, assuming an initially fully charged V44 battery. In conclusion, the entire monitoring system is capable of surviving on its own power supply and solar panels.

3.2.6 Implementation Challenges

Even with a system architecture as seemingly straightforward as that illustrated in Figure 3.1, it was of no surprise that a myriad of issues would arise in the course of developing the prototype. These issues are discussed briefly in the subsections that follow.

Bluetooth Implementation Resource Conflicts

It goes without saying that even on the Raspberry Pi board, many resources may be shared between many peripherals, be it data or communication related. If any resource is shared by two entities, it could lead to unforeseen errors. unfortunately, we encountered one large conflict while integrating the camera into the Raspberry Pi state machine.

Superficially, it should not be anticipated that a camera, which employs data lines on the Pi board to transmit pixel information to Pi storage [3], to conflict with the UART bus that the BLE chip on the Raspberry Pi uses [25]; however, we frequently encountered a resource conflict issue in the main thread of the Raspberry Pi camera module code. The main thread would scan for peripherals using BLE, as expected, and would eventually find the laser module. It would then connect with it and proceed onto the next states. The main thread would then create a camera object in Python, which is a programmatic way of initializing the camera hardware. A picture would then be captured, and the camera would close. The next state, **Image_1_Comm**, then requires a command to be sent to the laser module to turn off the laser beam. This message would never send, however, and the main thread would proceed to **Shutdown**, indicating an error code that an acknowledgment from the laser module was never received. After restarting the main thread in an attempt to replicate the error, the following error would arise:

```
bluepy.btle.BTLEException:  Failed to execute mgmt cmd 'scanend'.
```

This error, raised by `Bluepy`'s own error handler, indicates that because it cannot find the Bluetooth peripheral, it cannot scan for devices [26]. This error is highly indicative of a resource conflict. In many cases, if a resource is shared inappropriately, errors relating to memory or resource allocation occur, where the processor simply "jams" and can no longer manage the resources in question anymore. In the case of the BLE resource, the Pi was no longer able to even initialize the BLE peripheral when the main thread was terminated.

After extensive debugging using lower level Linux `hci` Bluetooth tools to attempt to somehow uncover where this conflict was occurring, we employed a high-level solution, which was to simply use a separate BLE dongle. After employing a separate Class I BLE dongle, as seen in Figure 3.15, the error was eliminated. Although the resource conflict was avoided, this proved to be an unsatisfactory solution as the initial resource conflict was not resolved. It was ultimately found that the Raspberry Pi reserves GPIO pins 32-33 for Bluetooth [6]. The camera, however, was implemented using another Python module, `PiCamera`, which, without much documentation regarding this setting, utilized GPIO pin 32 on the Raspberry Pi to actuate an LED indicator on the camera breakout board. This was only uncovered after searching through the `PiCamera` source code on GitHub. By merely changing the GPIO pin that `PiCamera` accessed, the entire resource conflict was resolved in implementation.

Figure 3.15: Tripp Lite mini bluetooth USB adapter 4.0 (Class 1).

Raspberry Pi Camera Memory Allocation

In the same vein of memory allocation issues, the same Python method `PiCamera` gave way to another error. This error was unique to this project in that it involved the long exposures that our images required, as discussed in the section on the computer vision algorithm (Section 2.3.5), and will be discussed in a subsequent section on testing and validation (Section 3.3.3). Understanding why this error occurred requires the knowledge that the Raspberry Pi Camera itself is a breakout board of a GPU (graphics processing unit) that interfaces with the image sensor. This GPU already has its own programming implemented in it. For instance, in many automatic capture modes, the GPU runs its own feedback loops to calculate the gains required to magnify each pixel value to generate a coherent image [27].

Unfortunately, the Raspberry Pi Camera is not suited for long exposures. By initializing the camera object as such:

```
PiCamera(resolution = (2592,1944), framerate = Fraction(1,6),
sensor_mode = 3),
```

we set the frame rate to a fractional value of $\frac{1}{6}$, which is considered to be very slow for the hardware. After capturing the image with these settings, one would want to use a `close()` method to "close" the camera. "Close" in this case means the de-allocation of any resources that the Pi had given to the GPU in order to interface with it.

With this slow frame rate, the Pi could not close the camera. The `close()` method would stall, with no other option but to kill the invoking process by using a `sudo` command through another terminal. The Pi would then immediately begin to suffer from unpredictable behaviors. In some cases, the monitor would freeze or begin to show black and white stripes. These indicate memory leak issues, i.e., because an unused resource was not properly de-allocated, another process afterwards malfunctioned due to this former memory not being released properly. Only a system reboot resolves this symptom.

It was ultimately found that this is a long-running issue with `PiCamera`, and that it was possible to resolve this issue programmatically without altering the source code by "tricking" the camera GPU. Since a fractional frame rate of $\frac{1}{6}$ is too slow, one could simply set the frame rate to 1, give the GPU a second or two to implement that change, and then call the `close()` method [28]; this workaround resolved the error.

GSM Cellular Module Implementation Challenges

Two issues were dominant in the implementation of the FONA cellular module. First, the FONA breakout board includes a reset pin, which when driven low (grounded) from its usual high state, triggered a hard reset of the FONA board, namely the SIM800H GSM chip. This reset could not be replicated when the pin was wired to a GPIO pin on the Pi, which would be driven low. When the reset pin was connected to the Pi's ground GPIO pin, the reset would occur. This is a critical issue, as the reset functionality is needed to (a) bring the FONA board out of sleep mode [16], and (b) reset the serial connection following the termination of the PPP connection. It was only after reading an obscure guidebook (via a forum post) not linked to the official Adafruit product page for the FONA [22] that the problem was found. The developers of the Adafruit board had installed a level shifter on the reset pin, which some micro-controllers such as the Pi, sometimes cannot overcome through their GPIO pins. Therefore, the recommended solution was to solder on a plain wire to connect both ends of the level shifter thereby by-passing it, as illustrated in Figure 3.16. This resolved this issue.

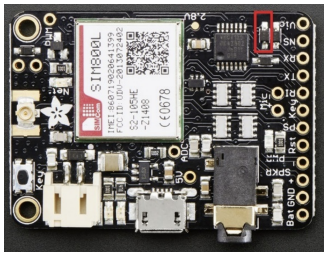


Figure 3.16: Location of FONA reset pin level shifter.

Second, the FONA plays a critical role in the functioning of the monitoring system in that it receives SMS messages from the server to "wake up" and take measurements. When the FONA receives an SMS or call, it will drive a ring indicator low for a few milliseconds. The Arduino will detect this and then deliver power to the Pi. When initially implemented, it was observed that the Arduino would seemingly start delivering power to the Pi when no wake up text or call was sent to the FONA. By using an Arduino with an SD card logger, we were able to poll and record the ring indicator line as we sent an SMS text as well as a call to the FONA.

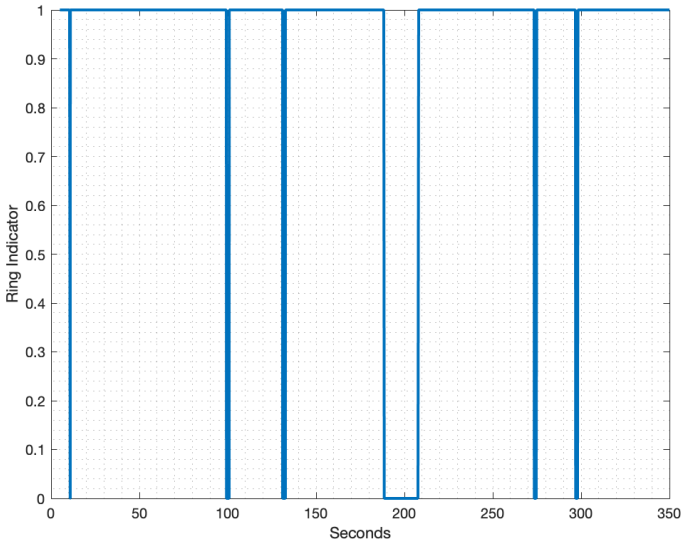


Figure 3.17: Expected ring indicator behavior following SMS and Call.

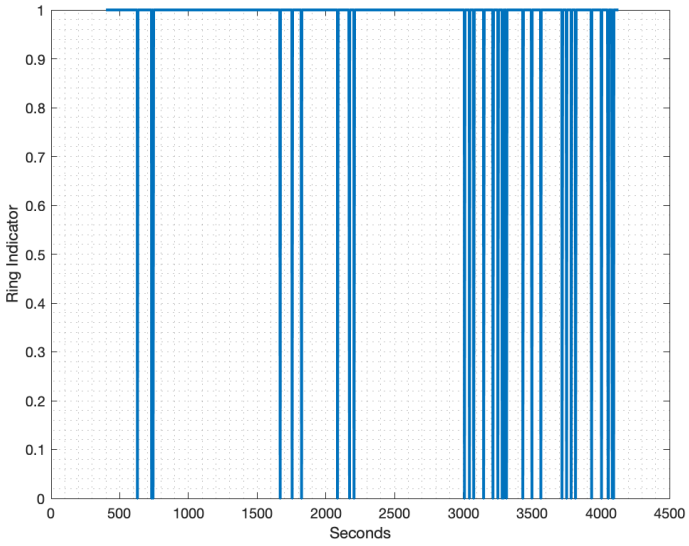


Figure 3.18: Erratic ring indicator behavior.

Figure 3.17 is a plot of the ring indicator state as we performed these actions. In Figure 3.17, we see an initial low dip due to the SMS and then two short dips following which correspond to the FONA reset in order to reset the FONA from idle, and the other to reset the serial connection to the FONA from the Pi following termination of the PPP connection. The same two blips are seen following the long low dip, which is due to the longer-duration call ringing. The rest of this test included erratic ring indicator behavior, indicative of a slew of subsequent SMS messages due to their ring indicator short in duration, despite not actually texting the FONA.

These subsequent ring indications were initially theorized to be due to spam calls, but they are too short to be considered calls; removing the antenna of the FONA board eliminated all of these erratic signals. It was only after digging into the SIM800H manual that the root of the problem was found. The SIM800H can store 30 SMS messages in total [16]. If this storage is full, subsequent incoming SMS will still elicit a ring indication from the FONA, but they will not be saved and not be acknowledged to the cell service provider as received. Therefore, the cell service provider will repeatedly attempt to re-send this SMS message to the FONA. Indeed, it was only after clearing the on-board FONA SMS storage that this erratic ring indicator behavior was eliminated. This task was added into the **Shutdown** state of the Raspberry Pi state machine; see Figure 3.10.

3.2.7 Programmatic Implementation & Organization

As in Chapter 2 for Phase I, all of the code written for Phase II and the files' dependencies on each other are provided in Appendix H and organized in the Raspberry Pi's file system. Figure G.13 illustrates this and is included in Appendix G. Note: this is only for the camera module, as the laser module is now extremely simple with just a singular Arduino `.ino` program flashed onto the Arduino in the laser module.

3.2.8 Bill of Materials

As this is the final prototype that will be delivered in this study, it is essential to present a bill of materials; see Appendix F. It was an expected result, though not an objective, to create a prototype that is reasonably priced and all components used are commonly-available commercial items. The component cost of the final system was approximately 900 USD, accounting for both the camera and laser modules. This price is divided nearly evenly one-third for the embedded system, one-third for the power supply, and one-third for the enclosures and mountings. It is significantly less than the probable cost for the system implemented in Feng and Feng (2016), which included a camcorder and laptop computer.

3.2.9 Enclosure Design and Fabrication

The design, fabrication, and assembly of the field-deployable enclosure for both the laser and camera modules required extensive effort. All materials used are included in the Bill of Materials in Appendix F. A brief description of the enclosures themselves, how hardware is mounted in them, and installation instructions for the system are included in Appendix D. In addition, the team created a video to visually demonstrate the assembly of the laser and camera modules [29]. Various views of the metal enclosures, both internal and external, are presented in Figure 3.19.

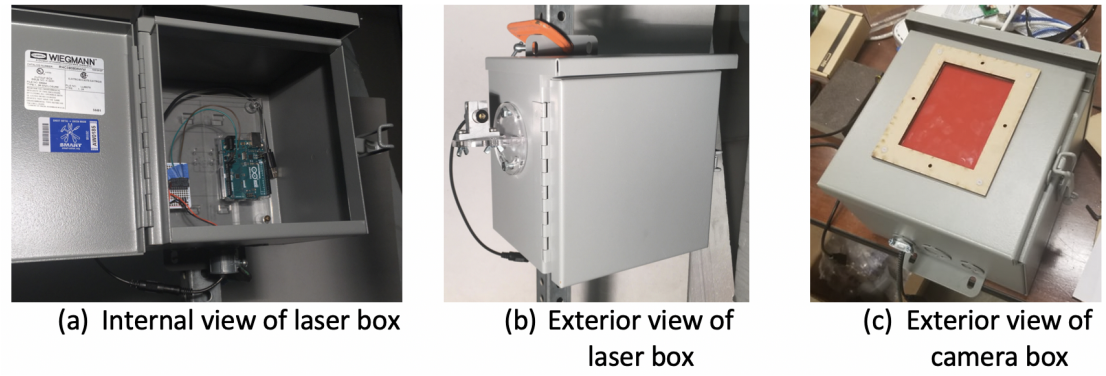


Figure 3.19: Phase II: internal and external views of the camera and laser enclosures.

3.3 TESTING AND EVALUATION

The testing and evaluation of this system can be divided into three parts:

- Evaluation of the embedded system's functionality and reliability;
- Evaluation of the accuracy and robustness projected laser target method; and
- Evaluation of the system performance when deployed outdoors.

3.3.1 Evaluation of System Functionality and Reliability

We define system functionality to be whether or not the system performs as expected. Reliability is defined as whether or not this system can perform those functions repeatedly under a variety of inputs. To evaluate Phase II system's functionality and reliability, a significant portion of this effort was performed in controlled laboratory settings throughout the development stages of each component of the system, and then integrating all components together.

First, we first coded the asynchronous code for the BLE communications and tested the code repeatedly to ensure that it operated standing alone. A simple way of accomplishing this was merely passing random test messages between the Pi and Arduino. We combined that communications code into the context of the state machine, as illustrated in Figure G.9, and then again tried to communicate between the Pi and Arduino to ensure that every state was transitioned through accurately. Next, the FONA module was interfaced with the Pi by itself, which required the writing of the chat script and network configuration file, as well as various Python scripts to implement AT control of the FONA programmatically. This was then independently tested by first attempting to connect to various websites on the Internet and then attempting to configure the settings of the FONA itself. This latter instance was how the issue with the reset pin on the FONA was discovered. We then integrated the FONA into the main code. Next, the Pi was interfaced alone with the camera and then incorporated into the main program, which led to the discovery of the resource conflict between the BLE and camera. Finally, the state machine incorporated the image processing, computer vision, and server push commands.

A study of our code and its various program flow paths led to the enumeration of the various ways the program could fail. The execution error handling capabilities were tested by purposefully

disconnecting certain components or hard-coding in various lines of code to elicit the anticipated error code and checking the subsequent error string in "`data_log.txt`." An example of one of these input sequences is shown in Table G.3 in Appendix G; refer to the error code list in Table G.3. Note: in order to force the Pi to record data to the data log instead of transmitting the data, the FONA was unplugged.

3.3.2 Calibration of the Camera Module

The calibration scheme for the camera module in this phase is very similar to that of Phase I: we obtained a scale factor F_{scale} in units of δ_{pixels} to δ_{mm} by physically shifting the camera chamber vertically relative to the laser beam, using the same method illustrated in Figure 2.25. We performed this calibration on both the 3D printed camera chamber, pictured in Figure G.4, which was developed in Phase I for laboratory testing and the metal camera module developed in Phase II for field deployment. The calibration curve for the 3D printed camera chamber was generated, resulting in $F_{scale} = -680.82 \ \frac{pixels}{inch} = -26.803 \ \frac{pixels}{mm}$; see Figure 3.20. Note: the negative slope is due to the orientation of the camera in the camera chamber such that a positive displacement is perceived as a negative position change in the axes of the camera.

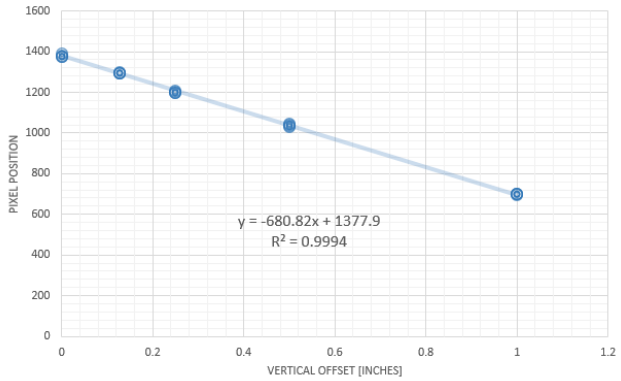


Figure 3.20: Calibration curve for the 3D-printed test camera chamber.

Calibrating the metal camera module required a slightly different physical setup as the metal box is hard to mount on a tripod or even on a table right-side up. The box was laid flat on its back on a carriage on a rail, with the laser pointed vertically onto the acrylic screen, as seen in Figure 3.21.

Our goal was to orient the camera in the camera enclosure to obtain a scale factor that matched the scale factor of the 3D printed test chamber. The hardware mountings in the metal camera enclosure were designed so that the field of view of the camera matched the field of view in the 3D printed test chamber. The equality of these scale factors verifies whether or not the test chamber was successfully duplicated by the metal enclosure. Due to limitations in mounting precision, however, we were unable to achieve a scale factor that precisely matched the scale factor for the metal enclosure. Instead, we mounted the camera to match the test chamber's scale factor as closely as possible and obtained a scale factor of $-628.42 \ \frac{pixels}{inch}$; this unique scale factor was sent to the server to implement.

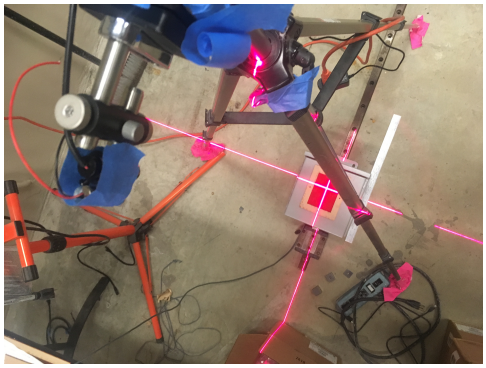


Figure 3.21: Calibration setup for the metal camera enclosure.

This report does not discuss the server and website portion of this system in detail, but we believe that the solution presented in the preceding paragraph deserves some discussion. The server sends a wake-up signal to the system. It also receives measurement values from the system. With these values, the server first calculates the difference in pixel position, $[\delta p_x, \delta p_y]$, of the image by subtracting the previous position values from the most recent measurement, and then applies the scale factor to convert the perceived displacement from pixels to mm. Note: the monitoring systems for multiple deployments on a bridge network do not share the same scale factor; therefore, the server must store not only the phone number of each deployed system's SIM card, but also associate with that phone number that system's scale factor. Refer to Appendix E for more information concerning the computer vision algorithm, server, and website.

3.3.3 Evaluation of the Projected Laser Method

We preface this section of evaluation with the reminder that this report is concerned with the application of embedded system methods to extend vision-based displacement measurement methods as a viable protocol for long-term monitoring. The projected laser target method introduced in the first chapter of this report determined issues of interest in regards to the hardware and enclosures developed so far. In addition, discussed were how the designed embedded system performed and the actions and tasks necessary to implement the computer vision algorithm developed. The hardware and software were evaluated by extensive testing of the functionality and resiliency of the embedded programming of the monitoring system, as well as subjected to the outdoor deployment of the system.

This section considers whether the system described within obtained measurements that met the sub-millimeter precision and accuracy required by Caltrans. In addition, discussions of the results regarding Phase II are largely concerned with the performance of the computer vision code, as well as its interactions with real-world data and limitations beyond the embedded system.

Experimental Setup

The majority of the experimentation was performed using the setup illustrated in Figure 3.22, in which two tripods, one holding the laser emitter, and the other holding the 3D printed camera module, are placed a distance away from each other. Recall that this investigation was concerned

with the limitations of the projected laser target method as implemented by our computer vision algorithm, in terms of the questions posed at the end of Phase I, Section 2.4.4. We will consider these questions in the following sections.

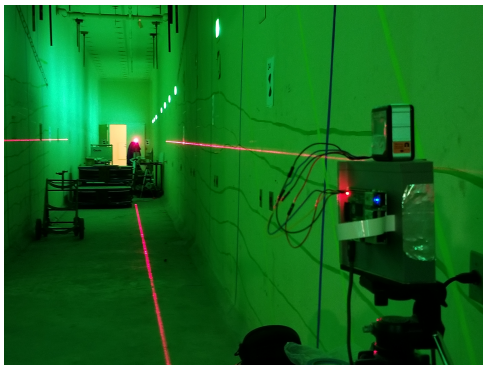


Figure 3.22: Tripod setup for lab testing of the projected laser target method.

Camera Setting Considerations

Before considering the actual act of taking images, we needed to determine the optimal camera settings. Based on documentation of the Raspberry Pi Camera [27], the two main variables to explore are shutter speed (SS) and ISO, which are how quickly the shutter opens and closes, and the sensitivity of the image sensor to light, respectively. The higher the shutter speed value, the longer the exposure time and, therefore, the brighter the photo. The higher the ISO, the higher the gain that the image sensor data will be multiplied by, and therefore the brighter the photo. We approached this issue by taking photos of the laser beam in a dark environment (again, assuming that any photographs will be shot at night to improve the accuracy of the computer vision method) at various SS and ISO combinations. An example of these photos is shown in Figure 3.23.

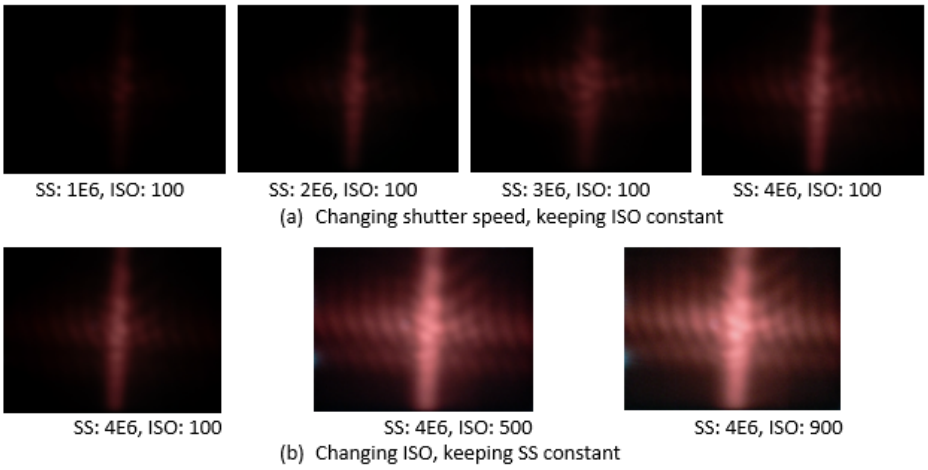
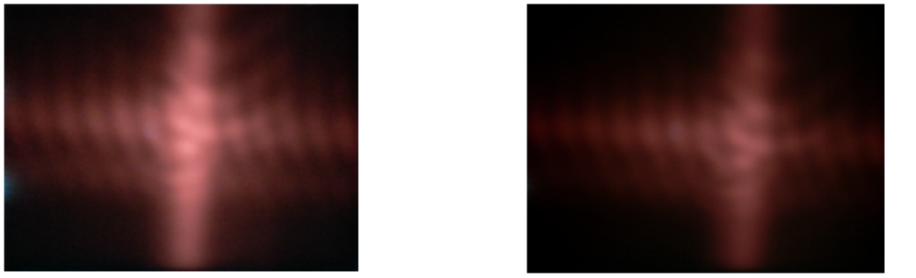


Figure 3.23: Photos of laser beam with varying SS and ISO.



(a) Camera Setting 1: ISO = 900, SS = 3E6 (b) Camera Setting 2: ISO = 300, SS = 3E6

Figure 3.24: Chosen camera settings: 1 and 2.

To determine how "good" the image was, we evaluated how well the computer vision code chose its regression points to match the image of the laser beam, i.e., how "sparse" were the regression points chosen by the computer vision algorithm? For brevity, two camera settings were ultimately found to be the least sparsity in the regression points: $SS = 3E6,\ ISO = 900$ and $SS = 3E6,\ ISO = 300$ or camera settings 1 and 2, respectively. Examples of photos taken at these settings are discussed next.

Figure 3.24 demonstrates why the two camera settings may elicit good centroid estimation resultsm which were due to the limitations of the laser. At the far range (taken at a beam–camera separation of 60 ft), the laser may loose focus and impart a "fat" and "blurred" image on the acrylic screen; see Figure 3.25. This can be corrected by either a more precisely cut lens for the laser diode to create the cross-hair pattern or simply altering the camera settings.

As seen in Figure 3.25, the highest intensity portions of the imparted beam designate the "true" beam, whereas the fainter parts represent lens effects. This method captures changes in the position of the laser beam with reference to an initial image; as the laser beam is "distorted" consistently by the lens between image captures, the same features will exist in the same relative locations in each successive measurement sequence. Note: the camera could be set with high-image ISO gains such that the algorithm will capture the high gradients of the images through its kernel operations, thus finding tight regression points. Alternatively, one could set the camera with lower ISO gains so that the lens distortions are not even captured in the image; the camera merely captures the higher intensity parts of the beam image, with the algorithm able to discern reasonable regression points. We shall discuss which camera setting is more successful in practice in the next section. Note that the captured images in Figure 3.24 are of the laser beam featured in Figure 3.25.

Evaluation of Displacement Measurements

Instead of analyzing images based on the inherent "noise" contained within them, what is key here is the accuracy and precision of the computer vision algorithm in matching actual displacement measurements. In a setup exactly identical to the calibration procedures using gauge blocks, we imposed known displacements on the camera chamber and attempted to match that value with the computer vision code. After setting up the tripods at various distances from each other (5 ft was considered close range and 60 ft was considered far range), we were able to discern the accuracy of our displacement determination methods. We present a subset of our findings in Figure 3.26.

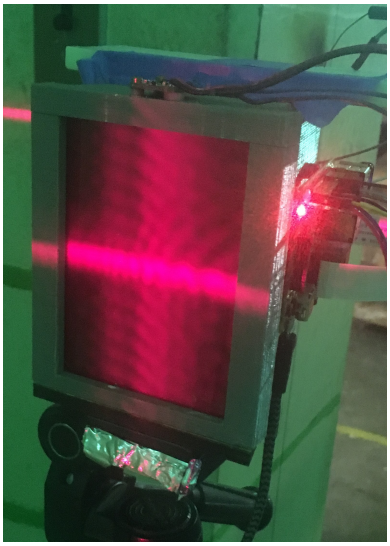


Figure 3.25: Effects of lens distortion at high range, 100 mW laser at 60 ft.

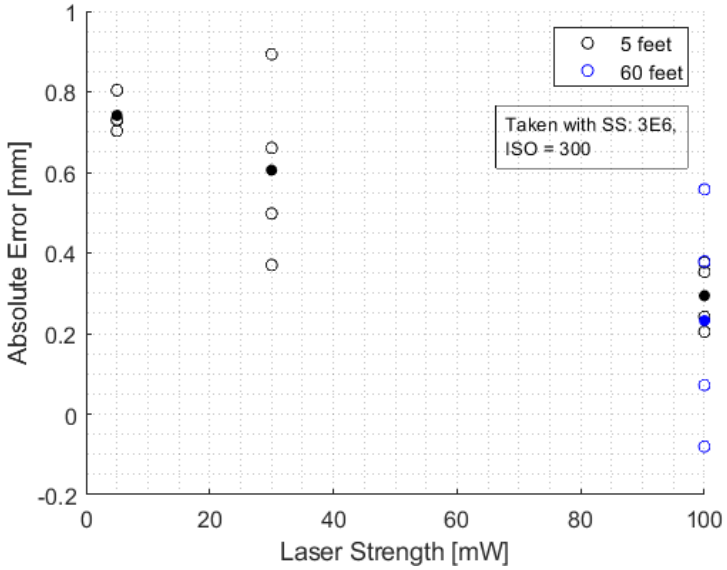


Figure 3.26: Errors of image-based measurements compared against gauge blocks.

We tested the efficacy of three strengths of lasers: 5 mW, 30 mW, and 100 mW. As shown in Figure 3.26, all of these methods obtain sub-millimeter accuracy in that the absolute error ($\delta_{vision} - \delta_{gauge\ block}$) was always of magnitude less than 1 mm. At close range, this is to be expected, but we do see the average absolute error corresponding to each laser strength (the filled dot in each sequence of dots) decreasing as the laser strength increases. This makes sense as the more intense the laser beam, the more discernible its centroid via the algorithm. That being said, at 60 ft, the accuracy of the measurements taken using the 100 mW laser was on-par with the accuracy taken with the same laser at close range. Thus, the computer vision algorithm can elicit sub-millimeter readings at a range of up to 60 ft.

This success hinges entirely on the performance of the laser emitter. The reason why there is no data included in Figure 3.26 for the other two laser strengths at 60 ft is that those images were so faint and blurred that no measurement was possible, indicating the importance of obtaining a laser diode of the appropriate focus and strength. If the laser beam is not focused enough, the images are blurred; see Figure 3.24. If the laser beam is not strong enough, there is not enough light to impart a discernible image on the acrylic screen.

Obtaining clear images is also dependent on the camera setting. If the laser is too weak, why not simply boost up the ISO or exposure time? Because images taken at 60 ft of the 100 mW laser beam proved that this requires careful tuning of those settings. Using camera setting 1 at a long distance, the absolute error was very large compared to darker images taken using camera setting 2. Referring to Figure 3.27, only images taken using setting 2 obtained sub-millimeter accuracy. Thus in the design for our final model, we plan to employ camera setting 2.

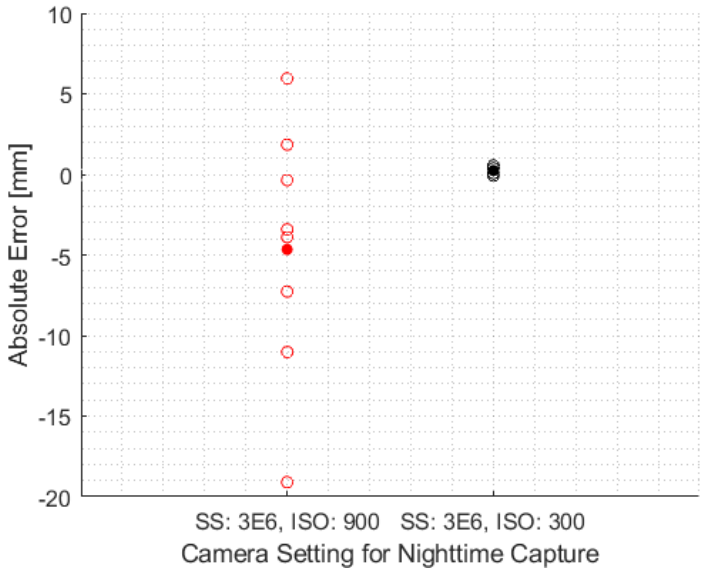


Figure 3.27: Influence of camera setting on errors, with reference to gauge blocks.

Camera Tilt Errors

We next investigated the effects of camera tilt on the displacement reading. We would expect the "perceived" translational displacement to be much less than if considering laser emitter tilt due to the significantly smaller arm of rotation (on the order of inches versus order of tens of feet). By using the same tripod setup as discussed previously, we kept the laser beam in a fixed position and merely rotated the camera module forward and backward to introduce pitch. Then we took images with reference to a zero-pitch image. Figure 3.28 summarizes the results of several trials of this nature.

Due to large imprecision in the camera-laser tripod setup, we cannot take the precision of these results as presented, but we can discern a first-order trend; the influence of camera tilt is nonlinear. This is to be expected as the camera chamber tracks a circle as the pitch grows about its center of rotation, with these snapshots of that path eliciting a curved trend. While this trend is

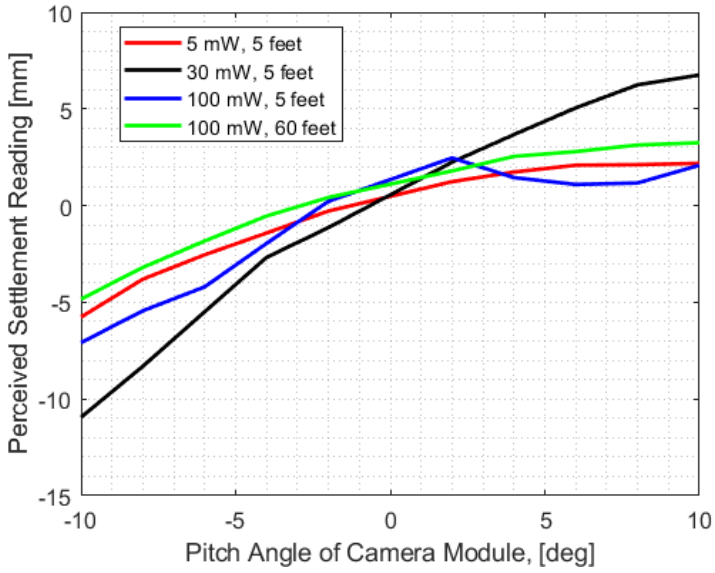


Figure 3.28: Influence of camera tilt on measurement.

interesting, it does not give us much information on how much camera tilt we can tolerate in our measurements. Taking these perceived settlement readings with a practical point of view, the fact that a mere 10^o of pitch may introduce up to an order of magnitude 10 mm of "error" is alarming. Clearly, if our transmitted data from the on-board accelerometers reports a change in pitch value greater than 1 or 2^o, then the estimated settlement measurements can no longer be trusted unless a correction of such undesirable rotation is performed; this topic is outside the scope of this report.

Outdoor versus Indoor Considerations

Another question to answer given laboratory conditions versus *in situ* conditions: Is the computer vision algorithm developed robust enough for outdoor conditions? In particular, we were concerned with aberrant light, artifacts, and non-night-time ambient light conditions. To address the issue of light artifacts, we purposefully left hardware LEDs on in the camera chamber and a few smudges on the acrylic screen, all in an effort to test the efficacy of the algorithm. The image subtraction of a baseline "ambient" photo from the "laser" photo worked very well; see an example of this image subtraction in Figure 3.29.

Figure 3.29 shows a smudge very close to the center of the image, as well as a stray streak of light on the bottom left corner. When the ambient photo is subtracted from the laser photo, these two artifacts were essentially eliminated, resulting in a "clean" photo.

A more pressing concern remained: outdoor light levels. As successful as this algorithm may be in laboratory settings, we identified three key light "levels" that the system may encounter in the outside world, each of which could or should be handled differently if daytime photos are a future objective of this monitoring system.

The first "level" considers images taken at night, where the camera essentially sees a completely black acrylic screen when there is no laser beam shining on it. These pictures result in

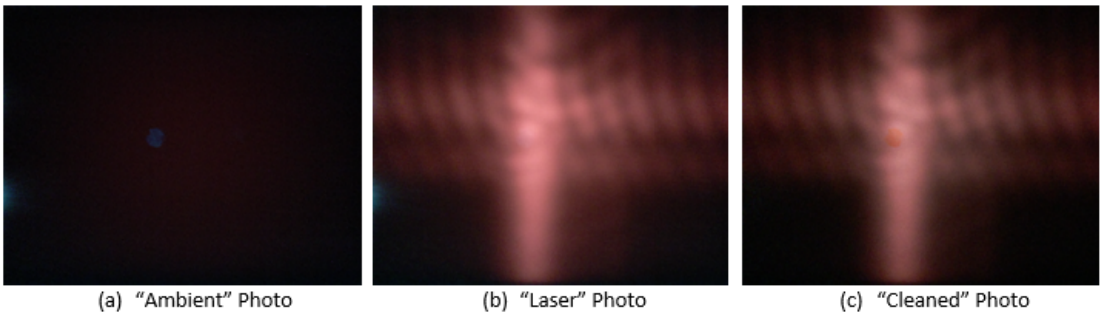


Figure 3.29: Effects of image subtraction.

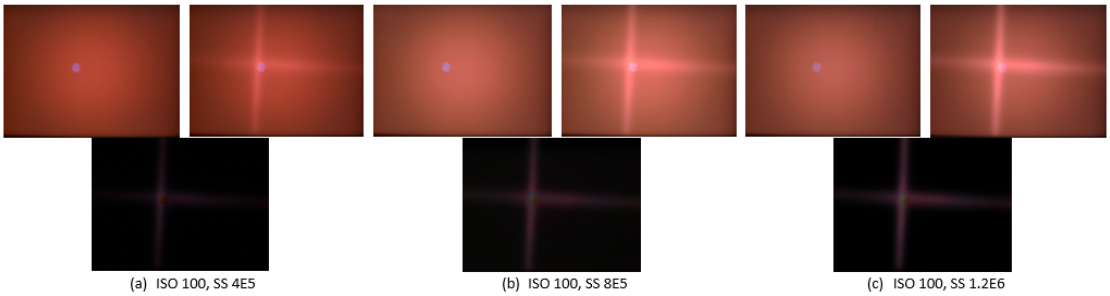


Figure 3.30: Images of various camera settings in the "shade."

good computer vision results that matched the laboratory results. Second, there is also a "shaded" regime, where there is no direct sunlight shining on the screen. Here, it was possible to see the screen lit up fairly evenly across the red screen from the inside. In addition, the red acrylic acts as a fairly good "filter," blurring out patterns and actual visual details such that we are left with a fairly uniform baseline image. Although this is desirable for good estimation of results, how bright and even the screen should be was of concern. If the screen is too bright, then the laser beam will not impart a high enough increase in brightness to be discernible from the ambient light, even when image subtraction is performed. To ensure this does not occur, we adjusted the values of ISO and exposure time to capture a reasonably differentiable image of the laser beam. Images taken with various camera settings in a "shaded" regime are presented in Figure 3.30.

The second row of images in Figure 3.30 shows subtraction-corrected images of the images in the first row, where the image to the left is the baseline "ambient" image to be subtracted from the actual laser image on the right. Depending on the camera settings, the laser beam in the resulting corrected image may be of fainter or stronger appearance. Tuning these settings autonomously proved difficult. We attempted to correlate the brightness in the camera chamber itself with the ambient light conditions outside the chamber by hooking up a photo-resistor to the Raspberry Pi. Unfortunately, it was found that ambient light conditions are much too variable for a single photo-resistor in the chamber to offer enough information for the control of the camera settings. One of the goals of this project was to keep the hardware and software relatively simple, thus avoiding the installation of feedback control methods to auto-tune the camera settings for daytime measurements; therefore, we instead decided to continue with only night-time measurements.

Finally, the last daytime light "level" we encountered was in the case of direct sunlight on the screen. In that case, it was proved to be difficult in aiming the laser beam onto the acrylic screen;see Figure 3.31. Even at close range, the photos taken with this method were not very useful; see Figure 3.32. It is impossible to discern the laser beam at any reasonable camera setting, since the ambient and the laser images were almost identical, leading to a nearly black image after subtraction. If the image algorithm is fed images taken under these conditions, there is no chance that any viable position estimate will be output.

Figure 3.31: Direct sunlight conditions.

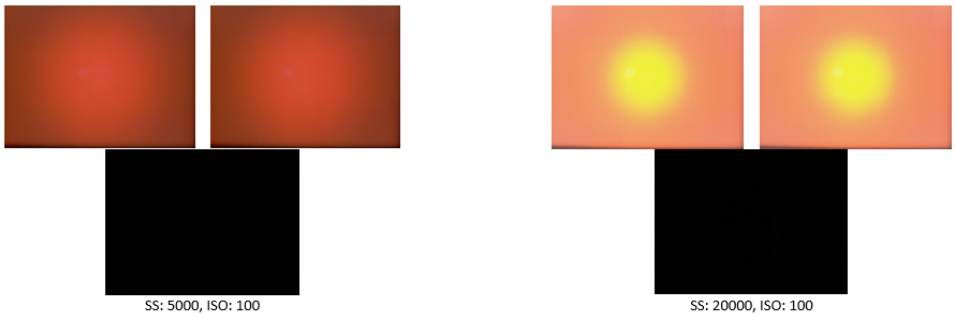


Figure 3.32: Images taken under direct sunlight conditions.

3.3.4 Outdoor Deployment

In order to fully test the functionality of the monitoring system, as well as the practicality of its installation and limitations of operation in the real world, we deployed the system on a bridge located on the campus of the University of California, Berkeley (UCB). This outdoor deployment is discussed in this section as the final step toward validating the developed monitoring system.

Site Selection and Mounting

The bridge selected for the in-field testing of the monitoring system is located on University Drive on the UCB campus, immediately north of the Moffitt Undergraduate Library, as marked in Figure 3.33. As it is located on a major thorough-fare through campus, this bridge experiences frequent vehicle, bicycle, and pedestrian traffic. This bridge had recently undergone retrofitting; the wooden

deck was resurfaced and is now supported by a series of steel I-beams that span the length of the bridge from abutment to abutment; see Figure 3.34.

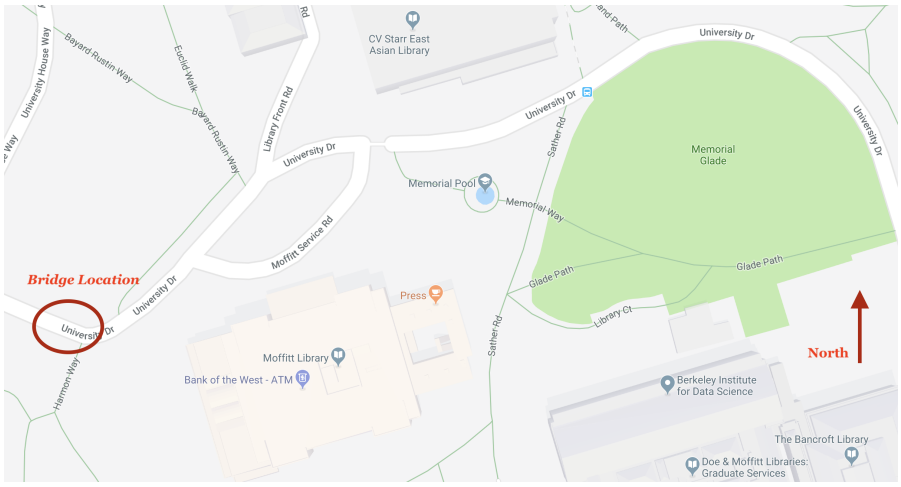


Figure 3.33: Selected bridge for field tests on UC Berkeley campus.

Figure 3.34: Bottom view of selected bridge.

We mounted the enclosed units onto the new I-beams using uni-strut members and C-clamps, as seen in Figure 3.35, placing emphasis on fixing the units as firmly as possible to limit any tilt or slippage of the units. The units were mounted approximately 12 ft apart from each other. The two corresponding solar panels were connected in parallel and were for the camera unit and the laser unit. They were mounted separately from the boxes so that they could receive sunlight; see Figure 3.36. The solar panels were then connected as needed using USB-compatible extension cords to the V44 batteries located in each module.

At this point, with the units mounted in their desired locations, and the solar panels mounted and connected to their respective batteries, we proceeded with aligning the laser beam with the camera screen. To accomplish this, we had to devise a power delivery system that was independent from the power delivery system in the laser module. The easiest way of doing this was by providing

(a) Camera unit

(b) Laser unit

Figure 3.35: Field installation of monitoring units.

(a) Parallel solar panels

(b) Single solar panel

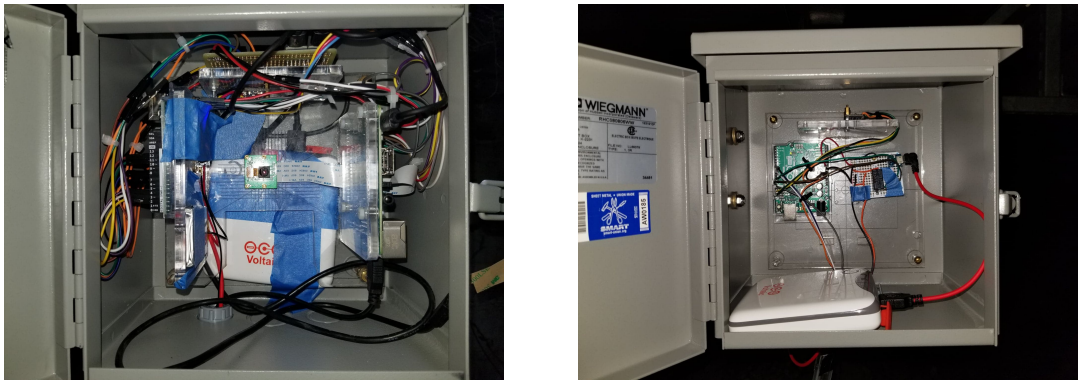
Figure 3.36: Field installation of the solar panels.

a separate power supply (e.g., battery) and power cable. Then, we focused the cross-hair beam at the desired range (upon the camera screen) by twisting the lens cap of the laser diode. Once focused, we used electrical tape to fix the position of the lens cap. Then, we inserted the diode into the aluminum laser mount, as seen in Figure 3.37, and used a hex key to tighten the grip of the laser mount on the diode.

Figure 3.37: Custom adjustable laser fixture.

Referring to Figure 3.37, we then adjusted the horizontal position of the beam by releasing

the wing nuts and adjusting the custom-designed laser fixture. We then tightened the wing nuts and adjusted the vertical position of the beam by adjusting the laser fixture after loosening the nuts that attach the fixture to the metal box. After tightening those nuts and reconnecting the laser diode to the V44 battery in the laser module, we sealed off any electrical joints with tape and confirmed that the Arduino in the laser module was receiving power from the V44 battery. This can be done by ensuring the power delivery board in the laser module is connected to the V44 battery, and that the battery is charged and outputting power; see Figure 3.38. The HM-10 BLE unit should be flashing red, and the Arduino's green power LED should be on. Finally, we locked the laser module with a padlock. At this point, the Arduino in the laser module was in idle, with the HM-10 BLE transmitter advertising. Next, we proceeded to initialize the camera module.



(a) Camera unit (b) Laser unit

Figure 3.38: Internal view of the hardware.

For the camera unit, we only needed to ensure that the appropriate units were receiving power. To do so, we checked that the V44 battery was charged and outputting power (Figure 3.38) and connected to the power delivery board. The Raspberry Pi's red power LED should be off. Regarding the Adafruit FONA, the blue LED should be solid and the red LED should be flashing slowly on the FONA's board, and its battery should be charging, i.e., the yellow charging LED indicator should be on. The green power LED on the Arduino should be on. When the camera unit receives a wake-up signal, the FONA charging LED will switch off, and the Pi will turn on. At this point, we can close this box and lock it.

Deployment Challenges

This stage of the project revealed various considerations that were not obvious in laboratory tests. The two most significant challenges were the limitations of 2G coverage and the sensitivity of the monitoring units to vibrations.

Recall that the monitoring system ultimately relies on three communication pathways: (1) a 2G call connection for the wake-up signal; (2) a BLE connection between the camera and laser modules for measurement commands; and (3) a 2G data connection to the Internet for data transmission. All three of these components were tested independently during development. For instance, to test connection (2), we placed the two modules at varying lengths in a clear outdoor space and found that the selected hardware allowed us to obtain connection at ranges in excess of

60 ft. Note: this remarkable range required that no obstructions block the field of transmission between the modules.

The 2G coverage was tested by sending SMS messages to the FONA in both a clear outdoor setting as well as in a room inside a reinforced concrete building. With this in mind, it was expected that the 2G coverage underneath a bridge located in the middle of UCB campus would have been more than sufficient. Unfortunately, this was not the case. It was found that when mounting the unit onto the bottom flange of a thick I beam, which was merely a few feet from the edge of the bridge, neither SMS messages nor calls could reach the FONA. Refer to Figure 3.34 for the underside view of this bridge.

To address this problem, two solutions were investigated: (1) extending the antenna to reach the edge of the bridge; and (2) physically moving the unit to the edge of the bridge. Solution (1) showed promise when implemented in a clear setting as the signal strength is high, and any signal losses experienced through a long antenna (approximately 70 in.) were not significant compared to the strength of the received signal. When tried underneath the bridge *in situ*, the FONA could not receive calls. We postulate that the signal strength, even at the edge of the bridge, was too weak to overcome the losses in the lead wire.

To investigate why the signal strength may be so weak even at the edge of the bridge required undertaking efforts not possible in our time constraints, so we therefore attempted Solution (2), which worked. By mounting the camera module onto the perimeter beam (by drilling an angle bracket into the wooden beam), the FONA with an un-extended antenna was able to pick up signals the majority of the time.

The second issue concerns the sensitivity of the measurements to vibrations. Recall that the monitoring system is designed to take static measurements. These images require long exposure times (in excess of 3 sec) as the camera takes pictures in a very dark enclosure and is designed for night-time measurements in order to eliminate as much light pollution as possible per the computer vision algorithm. During those 3 sec of exposure time, the image sensor is capturing light. In laboratory tests, the camera module was still during those 3 sec, but this bridge experiences significant pedestrian and vehicle traffic. It was observed that the bridge–despite its large I-beam substructure–vibrates substantially. This vibration is substantial enough that the laser beam is noticeably not still on the camera screen while the image is captured. This would of course introduces error in our measurement values as discussed later.

Deployment Results

We deployed the system and operated it for several days. Despite removing the camera module twice at the onset in order to fix and debug the unit, all of the data transmitted successfully was saved on the server. This data was isolated from monitoring service interruptions as the trajectory of the laser beam and the position of the camera module were not changed. For instance, if a technician is sent to open either of the modules to inspect the hardware, the system can be merely rebooted (power-cycled) after the inspection is completed; any subsequent data transmitted to the server is completely valid and merely appended to any previous data already collected. In the worst case scenario, all data files (captured images and data log files) are saved onto the Raspberry Pi's SD card. This card may be removed by hand, and its information accessed separately. Figure 3.39

presents the displacement measurement data collected over the course of a week. Note that these plotted changes in position were calculated by designating a single datum as a reference (the first point in a sequence of captured images) and subtracting that datum from each subsequent value. In this case, the reference point is the first data point collected by the system after installation, and every subsequent point should closely match that reference point if no significant movement occurred.

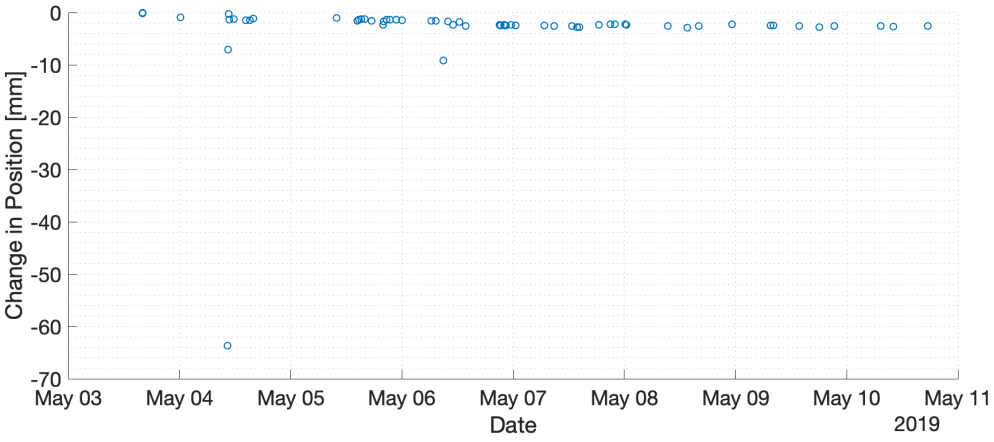


Figure 3.39: Complete one week displacement dataset.

Despite our laboratory test findings that demonstrated that our computer vision algorithm is most accurate when fed images taken at night, convergent results were taken by our monitoring system during the day. In this context, "convergent" means that the day-time measurements match the night-time measurements. This may be due to a variety of factors, which are not limited to: (1) the lack of light even in daytime underneath the bridge to begin with; and (2) cloudy weather conditions. To bolster these two possibilities, we note that on the mornings of May 4th and May 6th, there are three data points that digressed significantly from previous readings. This can be attributed to the morning hour sun conditions at the bridge, in which there was direct sunlight shining on the screen. If we remove these points, we are left with the data in Figure 3.40. We also present the tilt values measured using the digital accelerometers in the camera and laser modules over the course of the same week in Figure 3.41 that matched the "cleaned" dataset presented in Figure 3.40.

In Figures 3.40 and 3.41, we connected the data points to better visualize the point-to-point changes in the measurement values, choosing the first reading as the reference point that was gathered on May 3rd. We note that there seems to be two significant shifts in position of the laser beam centroid: first on May 4th, and then on May 6th. Initially, it was thought that this was attributed to the use of C-clamps to install the prototype under the bridge. The bridge shakes vigorously even in response to pedestrian crossings, so it would not be unusual if the clamps may have slipped once or twice. To confirm this assumption we studied the tilt readings; the camera module did not experience any significant tilt on May 4th or May 6th. If shaking of the bridge had caused the module to slip, we would expect some change in the tilt reading.

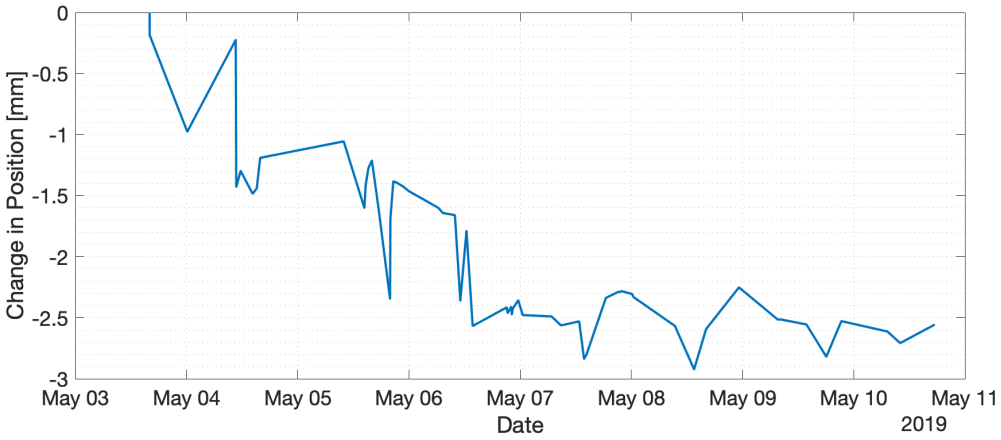


Figure 3.40: "Cleaned" one week displacement dataset.

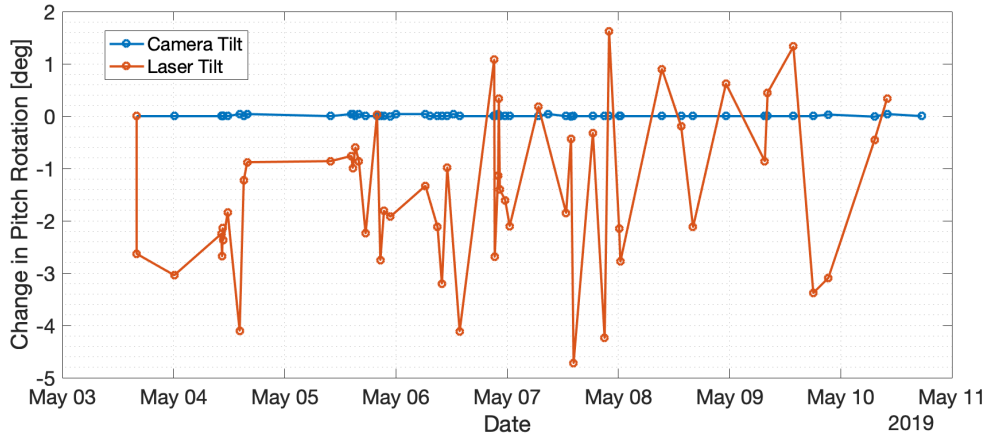


Figure 3.41: Complete one week tilt (pitch) dataset.

We acknowledge variability in the laser module tilt readings. These large readings are not real, but instead are due to jitter and fluctuations in the digital accelerometer mounted in the laser module. We can confirm this because 5^o of tilt of the laser module (the largest deviation in laser module rotation, as seen in Figure 3.41) would result in translations of over 3 cm on the camera module, which is not seen in Figure 3.40. To account for this electronic fluctuation in the camera module, the first few samples of the accelerations are sampled and then averaged. In the laser module, this was not done. Ultimately, more long-term testing and evaluation is required to fully validate this prototype for field deployment. Note, from mid-day May 6th and onward, the camera module measured very consistent readings by tracking an approximately -2.5 mm position with sub-millimeter precision.

This system was operational until May 15th, nearly two weeks after installation. At that time, it was seen that the camera module was still fully powered, but the laser module had nearly been drained its battery supply. This was due to a combination of factors: (1) the immensely cloudy and rainy weather that the test bridge experienced during the latter half of the deployment period; and (2) the laser module's solar panel was mounted in a very shaded region and posed a significant drain on the battery. As was observed post-installation, this panel saw direct sunlight for only a very limited period of time during the day. Worth noting is that the laser module was able to sustain itself for this long with sub-optimal weather conditions and the less than optimal location of the solar panel mounting lends confidence to the power consumption estimates and calculations performed in Section 3.2.5. This, combined with reasonable displacement and rotation measurements, demonstrates the viability of this prototype, and warrants building on the limitations noted herein as a jumping off point for Phase III.

4 Summary, Conclusions, and Future Directions

Below is a discussion of the performance of this system in terms of the projected laser target method, the embedded programming that implements the measurement method for monitoring, and the monitoring system in general.

4.1 DISCUSSION OF PROJECTED LASER TARGET METHOD

This method was proposed in order to improve upon the issues encountered by the many previous research teams who used vision-based methods. These issues included the necessity of special camera equipment to zoom and capture a far-away target and eliminate artifacts from ambient light conditions. By deconstructing the painted target, we proposed to bring what was on the canvas much closer to the camera in the form of a translucent acrylic screen inserted into a fixed camera chamber that projects the pattern via a laser beam from far away. This would absolve the need for special camera equipment and reduce the effects of ambient light by enclosing the camera's field of view into a chamber.

Ultimately, laboratory tests proved the projected laser target method was successful in meeting its goals, albeit under certain conditions; the largest limitation was the strength and focus of the laser beam. Eliminating the need for special camera equipment created the need for a higher-quality laser emitter. The selected 100 mW laser that we chose for this project had enough wattage to project a strong enough beam at a distance in excess of 60 ft, but unfortunately the lens that forms the cross-hair shape (see Figure 4.1) is not of high enough quality to produce a focused beam at that distance.

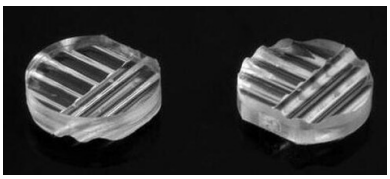


Figure 4.1: Typical cross-hair laser lens.

To this end, there are two straight-forward options to increase the effective range of this system:

1. Use a higher quality laser with a focus on the abilities of the laser beam at high range, respective to the desired range
2. Impose certain camera settings in order to account for the lens distortions; and
3. Explore the use of a non-cross-hair laser beam.

The first option requires higher monetary investment compared to the setup developed herein. We explored the second option and found it produced sub-millimeter results at 60 ft; see Section 3.3.3. Manipulating the camera settings to resolve this hardware limitation is a clumsy solution at best. We recommend using a higher quality, more expensive laser.

The third option would be a promising route for future investigation. The motivation of using a cross-hair beam came from many sources, one of which was the study by Olaszek (1999), which implemented a cross-hair patterned target. This in turn led to the development of the edge-detection computer vision method, as implemented for the purpose of this vision-based method.

Another tack might be to use a mere dot, which would resolve many of the issues resulting from focus and strength limitations of the laser. The laser beam, as produced in the laser diode, is inherently circular; the lens used to create the cross-hair beam is expected to always have shorter effective range than a lens that focuses the already circular beam into a finer beam. This would necessitate the creating a different computer vision algorithm to calculate the centroid of the laser point.

That being said, a similar image subtraction approach would probably be just as adequate in removing light artifacts. In addition, one could explore the use of many laser points instead of just one, and then approach this projected laser target method with more of a structured light motivation, following the work done by Myung et al. (2011) and Jeon et al. (2011). These methods also have the advantage of utilizing affine transforms to produce measurements of more DOFs other than merely one vertical displacement, including rotations. This would eliminate our reliance on accelerometers to measure the tilt of the modules.

In addition, the projected target method throws significant challenges when implemented in non-dark conditions. Regardless of how focused or strong a laser beam, be it a cross-hair or simple dot, if the acrylic screen is "saturated" with light and the laser beam can impart no significant increase in brightness on the screen from baseline conditions, no computer vision algorithm can realistically discern the position of that beam. This is a hard limitation in the projected laser target method. One can step around this limitation is by taking measurements in the shade or at night (and reworking the camera settings as needed), as we ultimately did in our field deployment.

To create a more light-robust method may require more information concerning the light conditions in the camera chamber as well as the outside world. With some measurement of the internal and external light conditions, some automatic modification could be made of the camera settings. A brief investigation into this during the course of this study was done by installing a photo-resistor (such as the one in Figure 4.2) in the 3D printed camera chamber and attempting to correlate the amount of light in the chamber with the convergence of the computer vision algorithm. "Convergence" in this case means if the computer vision algorithm was able to discern a reasonable centroid or not.

This photo-resistor produced inconsistent values and was very sensitive to its mounting orientation within the camera chamber. Even if light conditions were accurately read, extensive work would have to be made to develop and implement the feedback control of the camera settings to match certain light settings.

Figure 4.2: Analog photo-resistor.

4.2 DISCUSSION OF EMBEDDED PROGRAMMING FOR MONITORING

The embedded programming of the computational unit for this monitoring system was developed to implement the tasks necessary to perform the projected laser target method. The embedded system was designed according to the framework posed by Wang et al., a computational unit interfacing with communications and a sensor interface [Wang et al. (2007)]. To this end, we developed two sets of embedded programming for this project; one for each phase (I for laboratory deployment and II for field deployment). A software suite that implemented synchronous communications as well as time synchronization, adept for lab-space deployment using TCP/IP across a wireless router, was developed in Phase I but was not extensively tested nor evaluated.

A second software suite was developed for Phase II that implemented asynchronous communication using BLE technology to allow for communication between modules in the monitoring system and synchronous 2G GSM communication to transmit data to a server. This set of software was designed to be robust, by handling errors to prevent fatal program executions, and resilient, in that any data (including error codes) not transmitted during one program execution would be saved for future transmission. This software was tested and debugged in a laboratory setting, and its error-handling capabilities were evaluated. It can be expected that the same behavior will follow suit when the system is deployed in an outdoor setting as the hardware has not been changed, nor has the software; it was placed into a metal box and located outdoors.

This was validated as it was confirmed that when data was not transmitted to the server from the deployed system, be it due to transmission issues or another safely-handled error, the data (be it complete or incomplete) would always be transmitted the next time a successful data connection was made with the server from the monitoring system. In no case did the system ever become unresponsive due to a fatal error encountered by the embedded program in either the laser or camera module during its run time. “Unresponsive” in this case means that the system does not respond to a wake-up signal. Recall that the camera module is designed to always return to an idle state in which the Pi is powered off. When a wake-up signal is received, the Pi is power-cycled, effectively re-booting the Pi. Much like when one may “turn their computer on-and-off again” to correct for any hardware or software failures that may have occurred, this corrects for many of the common problems that may be encountered by embedded systems, such as segmentation faults.

It is noteworthy to consider that this embedded system could be extended to other monitor-

ing applications. The BLE implementation could be adapted for communication between computational units performing tasks not related to this projected laser target method. The sensing interface could include other sensors. Being asynchronous or synchronous in nature, the communications could be implemented using other technology. Ultimately, the flexibility of this embedded system lies in its state machine design. If a proposed system can be broken down into a finite number of operational states, one could always piece those states together using transitions that rely on input signals (internal or external in source) that may be delivered through wired or wireless means, and then output actions and/or information.

4.3 DISCUSSION OF MONITORING SYSTEM AND RECOMMENDATIONS

Below is a discussion of the monitoring system that first details the limitations of the current system and what immediate improvements can be made to the current system. The second discussion focuses on future extensions of the project.

4.3.1 Limitations of the Current System and Immediate Recommendations

The monitoring system developed herein works in the context of the specific setting chosen. We deployed it under a bridge on the UCB campus. We requested and gathered measurement data points for several days that were convergent under the correct lighting conditions, and were able to track a "baseline" zero-displacement reference with sub-millimeter precision. "Convergent" in this context, merely means that the computer vision algorithm output a reasonable pixel value for the centroid of the laser beam. That being said, certain limitations have been observed. They are:

1. High power draw of the cellular module battery;
2. Limitations of the 2G coverage;
3. Limitations of the BLE connectivity; and
4. Remote accessibility of data and system settings.

High Power Draw of the FONA

The embedded system developed herein accommodated the specific use case of monitoring the gradual settlement of bridge foundations through static measurements taken at long periods. However, recall that due to limitations as discussed in the power consumption portion of Section 3.2.5 specifically concerning the cellular module's separate battery, the interval of time between measurements is limited to six hours. This is done to ensure the cellular module's battery is always sufficiently charged; unfortunately, it increases the overall power consumption of the camera module. To compensate for this, we used two solar panels in parallel instead of one. This is not a long-term solution. If this problem had been caught earlier in the project development, we would have endeavored to (1) use a less power consuming cellular module; or (2) design a more elegant way of monitoring the battery state.

Concerning Approach (1), a prerequisite for the device is that it must be operating at all times: it must be able to receive incoming messages/calls at all times. The SIM800 series of GSM

chips, as is included in the Adafruit FONA used herein, is a very common chip for implementing GSM communications in embedded systems. It may be the case that there are other GSM boards that employ it in a much less power consuming way compared to the Adafruit.

Concerning Approach (2), if we were to keep the Adafruit FONA, there are certainly other ways of ensuring that its separate power supply is always charged sufficiently. Instead of appending the task of checking the battery state of the FONA to the list of tasks that the Raspberry Pi must perform as described herein, this could be delegated to the Arduino instead. Because turning on the Pi automatically takes a measurement, we are bound by our current method to take a measurement every time we want to check the battery state. We only chose this approach is because the TX/RX serial lines of the FONA are wired to the Pi only in the current prototype. Adafruit programmed a unique AT command that directly returns the charge of the battery when the command is issued to the FONA, making this a very convenient way to employ to determine the charge of the battery.

Changing the wiring of the data lines is possible but with the caveat that it is important to ensure that the Pi has exclusive access to the data lines when it is on, and the Arduino has exclusive access to the data lines when the Pi is off with no overlap. To implement this, one could use a relay. The ones we already use in our power delivery board are DPDT (double pole double throw), meaning that there are two output ports that alternate in their states: when one is on, the other one is off, and vice versa. One could merely use one of these relays to share access to the FONA data lines between the two computational boards. In addition, the state machine deployed on the Arduino must be changed, i.e., whenever the Arduino regains access of the FONA, a serial connection to the FONA must be established in order to communicate with the FONA.

BLE Communication Limitations

This issue did not have any serious ramifications in our field tests. In the few cases where the camera module failed to connect to the laser module, the camera module handled that error, saved its code into memory, and then we wait for the next measurement. That being said, in our test bridge, the distance between modules is approximately 12 ft, and nowhere near the approximate 70 ft tested earlier with no issues. We acknowledge that this BLE range is a limitation on the range of our monitoring system, coupled with the strength of the laser, as mentioned in Section 4.1. If one were to mount this device farther than 70 ft, a longer range communication technology must be employed.

2G Communication Limitations

It was already mentioned that the 2G coverage underneath the test bridge was unexpectedly sub-par. We remedied this by mounting our module closer to the edge of the bridge and tried using a longer lead-wire for the antenna of the FONA. This is not a comprehensive solution to this problem. More permanent solutions exist: (1) use a stronger antenna or perhaps a 2G signal booster; and/or (2) migrate this communication pathway to a newer infrastructure, such as 3G or 4G. Concerning Solution (1), the stronger antenna is expected to be passive (see Figure 4.3), meaning that it should not consume significantly more power than the antenna currently employed in the prototype. Using a signal booster will be a non-trivial increase in the power consumption.

The motivation for Solution (2) is two-fold; newer cellular technology will not only carry

Figure 4.3: Auxiliary antenna.

more data faster (thus allowing the system to transmit more data to the server than merely a few bytes of data), but it is also expected to out-last 2G in terms of carrier adoption as 2G technology has existed In the United States since the 1990s. Many carriers already have phased out of 2G [30]; T-Mobile is the last carrier providing 2G GSM coverage. To this end, it may be advantageous to upgrade the cellular module of this prototype to employ 3G technology if not newer. In addition, with newer cellular technologies, we can expect to benefit from improved coverage and range.

Remote Accessibility of Data and System Settings

Another immediate limitation of our system is due to our very limited use case: the technician is only meant to install the unit, boot it up, and then leave it. The data is then expected to merely appear at the server after a measurement is requested. More access to the system would be helpful, e.g., a feature that could re-boot the entire camera module or laser module remotely in the case of an unanticipated hardware issue. The fact that both of the modules return to their respective idle states is our means of ensuring that a forced re-boot is not required. Instead of transmitting only the final measurement, perhaps the two photos used to generate those measurements could also be transmitted to the server. If some measurements stray from the baseline, an engineer would find it useful to see the raw photos to determine if the aberrant measurement was due to light pollution or actual settlement.

Finally, there are certain settings that are hard-coded into the embedded program on both of the modules, including the amount of query time allowed for connections to wait before returning a failed connection error and camera settings. If the engineer wanted to alter any of these settings, a technician would have to access the devices on site. A useful modification would be the ability to change these settings remotely. This could be implemented by sending the information through SMS messages to the camera module as the wake-up signal. Currently, the system is blind to the actual contents of the wake-up signal, but it could potentially read the saved SMS message and act on it. If the SMS message contains changes for the laser module, this information could be passed to the laser module via BLE.

4.3.2 Long-Term Directions

Modifications to the Projected Laser Target Method

It has already been mentioned previously that an extension could be made to this project by modifying the projected laser target method, itself. Instead of projecting a cross-hair laser beam, the

method could instead adopt more heavily from the structured light methods used by Myung et al.; see Myung et al. (2011) and Jeon et al. (2011). This study projected multiple laser dots, measuring the six DOFs that describe the camera module's position in space. Although this approach may be too complex for this project, we could use a single laser dot instead of a cross-hair beam. This would have the added benefit of simplifying the computer vision algorithm from an edge-detection method to something perhaps less complex and more robust.

If one were to use four dots, more DOFs could be explained, allowing the project to eliminate the need of the digital accelerometers to measure tilt. Investigate the accuracy of the competing methods would be prudent. Our current prototype and a modified prototype that employs a system of kinematic equations fed by measurements extracted from images of several laser dots to measure more degrees-of-freedom is worth investigation.

Implementation using Other Hardware

A critical limitation is the scalability of our prototype. We addressed the goal of this project by utilizing affordable and readily available hardware to implement our monitoring system. Meeting this deliverable was at a cost. It takes a non-trivial amount of time to duplicate our system, ensure that the hardware is integrated and then assemble it in a robust manner. The prototype deployed in our field tests is not easily replicated to produce more than a few units at a time. The mounting fixtures for the hardware securely had to be custom laser cut from sheets of acrylic. The hardware had to be assembled onto the mountings and even when wired together must be validated to ensure that not one of the many connections is done incorrectly or poorly. To conduct this validation, the software for the various computational units must be installed and compiled onto the boards. Finally, the camera modules themselves must be calibrated to determine the scale factor between pixels and engineering length units.

A large step to streamlining this process, and perhaps even improving the robustness of the system, is to design our own custom integrated circuit (IC) board that contains only the components that are absolutely necessary from all of the various hardware we used. For instance, for the Arduino, many of the LEDs and GPIO pins are unused, as well as the ADC converter. To this end, many hobbyists have created their own "Arduinos" by buying the micro-processor chip itself and building their own IC boards around it. If we extended this to both the Pi and Arduino, and perhaps the cellular board, as well, we would eliminate many of the hardware issues that arose due to broken components and faulty connections. Designing our own boards would vastly improve the quality control of the product and the organization of the many wired connections that currently exist in our prototype.

If a custom IC board is not used, one could also use other boards that are designed for the deployment of embedded systems, such as the Berkeley Mote. These custom devices may be compact and designed specifically for these applications. Note: these custom devices require a higher skill level and learning curve compared to the system developed herein. Along with budgetary restraints, these reasons contributed to us not adopting these other boards. Many groups do indeed use these devices to great success, as summarized by Lynch and Loh (2006).

5 List of URLs

[1] Fisher, C., “Using an accelerometer for inclination sensing,” https://www.analog.com/media/en/technical-documentation/application-notes/AN-1057.pdf, Analogue Devices, (2010).

[2] Raspberry Pi Foundation, “Camera module,” https://www.raspberrypi.org/documentation hardware/camera/,” (2018), accessed 11/2018.

[3] Vis, P., “Raspberry Pi CSI camera interface,” https://www.petervis.com/Raspberry_PI/Raspberry_Pi_ SI/Raspberry_Pi_CSI_ Camera_Interface.html, accessed 3/2019.

[4] Mitchell, B., “The range of a typical WiFi network,” https://www.lifewire.com/ range-of-typical-wifi-network-816564, (2019).

[5] Jennings, N., “Socket programming in Python,” https://realpython.com/python-sockets/#socket-api-overview, accessed 2018.

[6] “BCM2835 ARM peripherals,” https://www.raspberrypi.org/app/uploads/2012/02/ BCM2835-ARM-Peripherals.pdf”, (2012), accessed 2/2019.

[7] “Pull up resistor / Pull down resistor,” http://www.resistorguide.com/pull-up-resistor_pull-down-resistor,” accessed 4/2019.

[8] “Low Signal Relay, G6A,” https://www.mouser.com/datasheet/2/307/70175635-1189505.pdf, accessed 3/2019.

[9] Progeny.co.uk, “Back EMF suppression,” https://progeny.co.uk/back-emf-suppression/, accessed 11/2018.

[10] Sparkfun.com, “Crystal 16 MHz,” https://www.sparkfun.com/products/536.

[11] Microchip Technology Inc., “megaAVR data sheet,” https://store.arduino.cc/usa/arduino-uno-rev3.

[12] Hassaei, A., “Arduino timer interrupts,” https://www.instructables.com/id/Arduino-Timer-Interrupts/, accessed 3/2019.

[13] Ting Inc., https://en.m.wikipedia.org/wiki/Ting_Inc., accessed 4/2019.

[14] SimCom, “SIM800H/L Hardware Design V2.01,” http://www.sabreadv.com/wp-content/uploads/SIM800HL_Hardware_Design_V2.01.pdf, accessed 2/2019.

[15] SimCom, “SIM800 Series AT Command Manual V1.09,” https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf, accessed 2/2019.

[16] “Bluetooth Low Energy,” https://en.wikipedia.org/wiki/Bluetooth_Low_Energy, accessed 3/2019.

[17] “Bluetooth Low Energy,” http://bluetoothinsight.blogspot.com/2008/01/bluetooth-power-classes.html, (2008).

[18] “CY3W43438 Single-Chip IEEE 802.11ac b/g/n MAC/Baseband/Radio with Integrated Bluetooth 4.1 and FM Receiver,” https://www.cypress.com/documentation/datasheets/cyw43438-single-chip-ieee-80211ac-bgn-macbasebandradio-integrated-bluetooth,” accessed 3/2019.

[19] JNHuaMao Technology Company, “Bluetooth 4.0 BLE module Datasheet,” http://fab.cba.mit.edu/classes/863.15/doc/tutorials/ programming/bluetooth/bluetooth40_en.pdf, accessed 2019.

[20] Murdoch M., “Try/catch block in Arduino,” https://stackoverflow.com/questions/10228562/try-catch-block-in-arduino, (2012), accessed 11/2018.

[21] Sparkfun.com, “Reducing Arduino power consumption,” https://learn.sparkfun.com/tutorials/reducing-arduino-power-consumption/all.

[22] Adafruit, “Adafruit FONA?,” https://forums.adafruit.com/viewtopic.php?f=22&t=56561 (2014), accessed 1/2019.

[23] Adafruit “FONA on power supply?,” https://forums.adafruit.com/viewtopic.php?f=22&t=56561 (2014), accessed 2/2019.

[24] Solar Power Authority, “How to calculate your peak sun-hours,” https://www.solarpowerauthority.com/how-to-calculate-your-peak-sun-hours/, (2012), accessed 4/2019.

[25] “The Raspberry Pi UARTs,” https://www.raspberrypi.org/documentation/configuration/uart.md, accessed 2/2019.

[26] Harvey, I., “Failed to execute mgmt cmd ’scanend,” https://github.com/IanHarvey/bluepy/issues/150, accessed 3/2019.

[27] Jones, D., “Camera Hardware,” https://picamera.readthedocs.io/en/release-1.13/fov.html, accessed 11/2018.

[28] Jaimemarijke, “Camera.close() with long exposure hangs, then kernel panics #528,” https://github.com/waveform80/picamera/issues/528, accessed 2/2019.

[29] Teng, H. and Binder, W. “Assembly tutorial,” https://youtu.be/xDdX3aY9kFg, (2019).

[30] Kovaleva, M., “Global 2G phase out: What do we know so far?,”

https://www.emnify.com/blog/global-2g-phase-out, accessed 4/2019.

[31] “System on a chip”, https://en.wikipedia.org/wiki/System_on_a_chip,” accessed 4/2019.

[32] shabaz, “ Raspberry Pi 3 block diagram, https://www.element14.com/community/community/
-pi/blog/2017/01/16/raspberry-pi-3-block-diagram”, accessed 3/2019.

[33] Geerling, J., “Raspberry pi dramble: power consumption benchmarks,”
https://www.pidramble.com/wiki/benchmarks/power-consumption,
accessed 1/2019.

[34] “Sobel Operator,” https://en.wikipedia.org/wiki/Sobel_operator, accessed 4/2019.

REFERENCES

Abdelbarr, M., Chen, Y., Jahanshahi, M., S.Masri, Shen, W.-M., and Qidwai, U. (2017). "3d dynamic displacement-field measurement for structural health monitoring using inexpensive rgb-d based sensor." *Smart Mater. Struct.*, 26(12), 125016.

Anderson, D. and Thorarinsson, A. (2009). *Long term bridge monitoring in Seattle, WA*. Seattle Department of Transportation, Seattle, WA.

Arias-Lara, D. and la Colina, J. D. (2018). "Assessment of methodologies to estimate displacements from measured acceleration records." *Measurement*, 114, 261—273.

Caltrans (2015a). "Bridge design practice." California Department of Transportation, Sacramento, CA.

Caltrans (2015b). "Five-year maintenance plan." California Department of Transportation, Sacramento, CA.

Feng, D. and Feng, M. (2015). "Vision-based multipoint displacement measurement for structural health monitoring." *Struct. Control Hlth.*, 23, 876—890.

Feng, D. and Feng, M. (2016). "Experimental validation of cost-effective vision-based structural health monitoring." *Mech. Syst. Signal Pr.*, 88, 199–211.

Fukuda, Y., Feng, M., and Shinozuka, M. (2010). "Cost-effective vision-based system for monitoring dynamic response of civil engineering structures." *Struct. Control Hlth.*, 17, 918—936.

Im, S.-B., Hurlebaus, S., and Kang, Y.-J. (2013). "Summary review of gps technology for structural health monitoring." *J. Struct. Eng.*, 139(10), 1653—1664.

Jauregui, D., White, K., Woodward, C., and Leitch, K. (2003). "Noncontact photogrammetric measurement of vertical bridge deflection." *J. Bridge Eng.*, 8(4), 212—222.

Jeon, H., Bang, Y., and Myung, H. (2011). "A paired visual servoing system for 6-dof displacement measurement of structures." *Smart Mater. Struct.*, 20(4), 04519.

Khuc, T. and Catbas, F. (2017). "Computer vision-based displacement and vibration monitoring without using physical targets on structures." *Struct. Ins. Eng.*, 13(4), 505–516.

Lee, E. and Seshia, S. (2017). *Introduction to Embedded Systems*. MIT Press, Cambridge, MA.

Lee, J.-J. and Shinozuka, M. (2006). "A vision-based system for remote sensing of bridge displacement." *NDT&E International*, 39, 425—431.

Lienhart, W. and Brunner, F. (2003). "Monitoring of bridge deformations using embedded fiber optical sensors." *Proceedings, 11th FIG Symposium on Deformation Measurements,* Santorini, Greece.

Lynch, J. and Loh, K. (2006). "A summary review of wireless sensors and sensor networks for structural health monitoring." *Shock Vib*, 38(2), 91—128.

McCallen, D., Petrone, F., Coates, J., and Repanich, N. (2017). "A laser-based optical sensor for broad-band measurements of building earthquake drift." *Earthq. Spectra*, 33(4), 1573—1598.

Moschas, F. and Stiros, S. (2011). "Measurement of the dynamic displacements and of the modal frequencies of a short-span pedestrian bridge using gps and an accelerometer." *Eng. Struct.*, 33, 10—17.

Myung, H., Lee, S., and Lee, B. (2011). "Paired structural light for structural health monitoring robot system." *Struct. Control Hlth*, 10(1), 49—64.

O'Connor, S. M. (2015). *Wireless Monitoring Systems for Long-Term Reliability Assessment of Bridge Structures based on Compressed Sensing and Data-Driven Data Interrogation Methods*. University of Michigan, Ann Arbor, MI.

Olaszek, P. (1999). "Investigation of the dynamic characteristic of bridge structures using a computer visio method." *Measurement*, 25, 227—236.

Ozer, E., Feng, D., and Feng, M. (2017). "Hybrid motion sensing and experimental modal analysis using collocated smartphone camera and accelerometers." *Meas. Sci. and Technol.*, 28(10), 105903.

Park, H.-S. and Lee, H.-M. (2007). "A new approach for health monitoring of structures: Terrestrial laser scanning." *Comp.-Aided Civ. and Inf.*, 22, 19—30.

Park, J.-W., Sim, S.-H., Jung, H.-J., and Jr., B. S. (2013). "Development of a wireless displacement measurement system using acceleration responses." *Sensors*, 13, 8377—8392.

Park, K.-T., Kim, S.-H., Park, H.-S., and Lee, K.-W. (2005). "The determination of bridge displacement using measured acceleration." *Eng. Struct.*, 27, 371—378.

Park, S.-W., Park, H.-S., Kim, J.-H., and Adeli, H. (2015). "3d displacement measurement model for health monitoring of structures using a motion capture system." *Measurement*, 59, 352—362.

Wahbeh, A. M. (2003). "A vision-based approach for the direct measurement of displacements in vibrating systems." *Smart Mater. Struct.*, 12, 785—794.

Wang, X., Wu, L., Zhou, Y., and Wang, Y. (2015). "The long-term settlement deformation automatic monitoring system for the chinese high-speed railway." *Shock Vib.*, 2015(2), 1—12.

Wang, Y., Lynch, J., and Law, K. (2007). "A wireless structural health monitoring system with multithreaded sensing devices: design and validation." *Struct. Inf. Eng.*, 3(2), 103–120.

Washer, G. (2010). "Long-term remote sensing system for bridge piers and abutments." *Transportation Research Board,* Washington, D.C.

Washer, G. (2011). *TechBrief: Reliability of visual inspection for highway bridges, FHWA-RD-01-020*. Federal Highway Administration, Washington, D.C.

Yunus, N., Ibrahim, N., and Ahmad, F. (2018). "A review on bridge dynamic displacement monitoring using global positioning system and accelerometer." *Proceedings, IntCET conference,* Patrajaya, Malaysia, 1930(1).

Appendix A: Discussion of Tilt Error Contribution

Jeon et al. (2011) realized in their development of a servo-ing system for their structured light approach, the limitation of projecting lasers a far distance onto a target is directly related to the target size. When one projects a laser beam far away from a target, small angle changes of the laser emitter will result in a large translation on the receiving end. Our projected laser method measures one-dimensional displacement by taking pictures of the interior face of the acrylic screen, making it blind to the tilt of both the laser emitter and camera modules. The effects of laser tilt and camera tilt are illustrated in Figures A.1 and A.2, respectively.

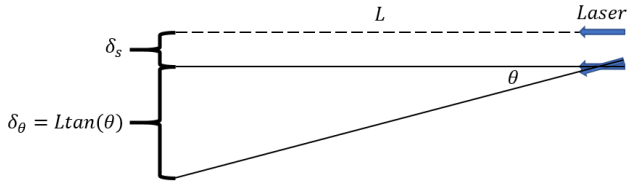


Figure A.1: Contribution of laser tilt to perceived translation.

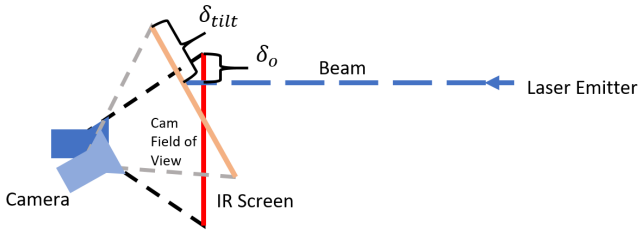


Figure A.2: Contribution of camera tilt to perceived translation.

In Figure A.1, we split the total measured displacement into two parts: "true" settlement δ_s and the contribution from an angular perturbation θ of the laser emitter, $\delta_\theta = Ltan(\theta)$. This means that a screen of vertical dimension h, which can capture translational displacement of $[-\frac{h}{2}, \frac{h}{2}]$, would only be able to capture an angular displacement of $[-arctan(\frac{h}{L}, arctan(\frac{h}{L})]$. Thus, if we had a screen 100 mm tall, we would only capture $\pm 0.286^o$ of laser tilt projected across 20 m. The issue of camera tilt is of lesser concern, as the rotation arm of the camera screen is much smaller than the rotation arm of the laser beam. The perceived settlement, as defined in A.2 as δ_{tilt} versus

the true settlement of δ_o, is certainly nontrivial if the camera is tilted enough. How much may be considered "enough" depends on the rotation arm, but due to the extreme difference in rotation arms, our team decided to manage these two error contributions as discussed below.

We used accelerometers to measure the angular position of both the camera and laser modules; there is no other way to obtain a tilt reading beyond a sensor measurement as the projected laser target method is blind to rotational influences. However, the resolution of typical digital accelerometers is not fine enough to measure the tilt of the modules to the precision required to correct any errors in the perceived settlement due to module tilt. For instance, at 20 m, an accelerometer on the laser emitter must be able to measure 0.00286^o to account for 1 mm of translational displacement on the camera screen.

This gap in information must be accounted for. We assumed that both the laser module and camera module will not rotate appreciably after installation. They will be fixed firmly; for highway overpasses this is a reasonable assumption to make. Washer's study (2010) used tilt meters to measure bridge column rotations and found that the major contribution to bridge piers was temperature fluctuation, with daily temperature cycles inducing a very trivial amount of tilt ¡¡l^o, all centered about zero. If static settlement measurements are taken daily at about the same time every day, these influences should not be of concern. That being said, the tilt measurements from both the camera and laser modules should not be ignored. They will be reported to the engineer in addition to the settlement estimate extracted from the projected laser image. We will merely not use the measured tilt to correct for any induced perceived settlement in the settlement estimate. The engineer may use their own intuition and judgement to determine if the reported settlement estimate and tilt measurement may warrant an on-site inspection.

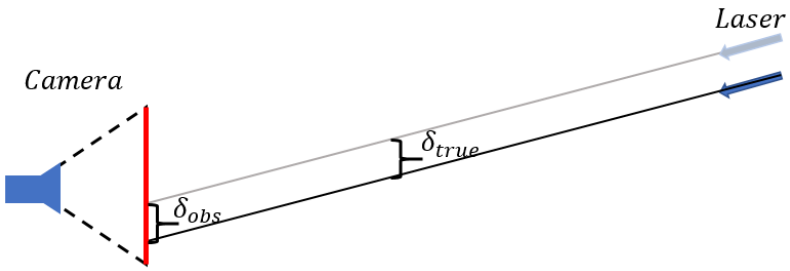


Figure A.3: Measured relative displacement.

Installation of the laser and camera modules is important. To reduce the perceived settlement errors induced by camera tilt, the camera module must be installed as close to perpendicular to the direction of settlement as possible. This ensures that any translational displacement caused by motion of the camera is due to the vertical settlement of that module. The laser module need not be installed such that the beam is perpendicular to the plane of the camera screen; the beam must simply not rotate following installation. So long as the laser beam imparts a pattern onto the screen when the initial reference frame is captured, the translational motion of either the laser or camera module will be correctly tracked by the projected laser method. This is illustrated in Figure A.3. Note that in this figure, the tracked displacement δ_{obs} is due to the vertical motion of the laser δ_{true}. Similar tracking can be made if the laser is still and the camera module moved down.

If both modules were to shift vertically, the effects on the measured displacement could be additive or subtractive. This is illustrated in Figure A.4, where both the camera and laser have shifted downwards, but the vertical motion of the laser, δ_{laser} is not equal to δ_{obs}, the measured displacement. Instead, the displacement of the camera module δ_{cam} has affected δ_{obs} in the following manner:

$$\delta_{obs} = \delta_{cam} - \delta_{laser} \tag{A.1}$$

This emphasizes that we are, in fact, measuring a relative settlement. Indeed, there is a reference point for the sequence of motion tracked according to the motion-tracking method adopted, seen in Figure 1.13, but there is not a reference point in physical space. The motions of each module are coupled by Equation (A.1). With the chosen projected laser target method, there is no other means for measuring either δ_{laser} or δ_{cam} to solve for the other. Therefore, this method can only track the relative settlement of one element to another, i.e., those elements being the mounting locations of the camera and laser modules.

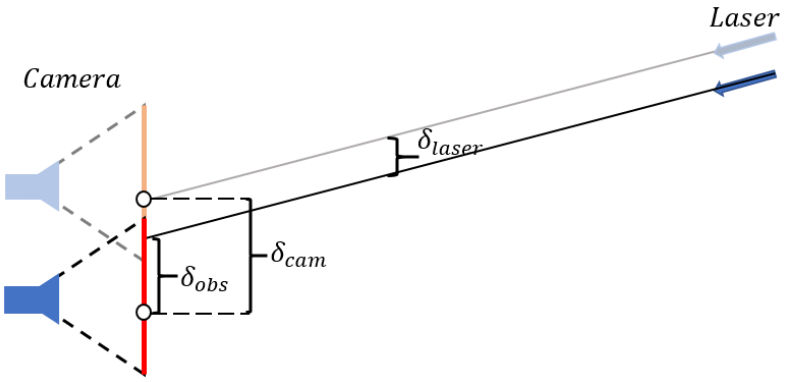


Figure A.4: Coupled measured relative displacement.

Appendix B: Microcontroller Selection: Raspberry Pi

A Raspberry Pi is a system on chip (SoC), which is a chip that integrates all components of a computer, such as the central processing unit (CPU), memory, input/output ports, and secondary storage [31]. In other words, a Raspberry Pi is a small computer. The latest model, the Raspberry Pi 3B+, Figure B.1, employs a Broadcom BCM2837 chip, which features a quad-core Cortex-A53 64-bit processor, with a maximum clockspeed of 1.4 GHz, with 1 GB of SDRAM memory. This device has more than enough computational ability to perform the required computational tasks. Wang et al. were able to implement Fast Fourier Transforms, wavelet transforms, and various other algorithms on a much simpler computing unit, the 8-bit ATmega128 micro-controller, which only has a clock of 8 MHz and 128 kB on in-system flash memory [Wang et al. (2007)].

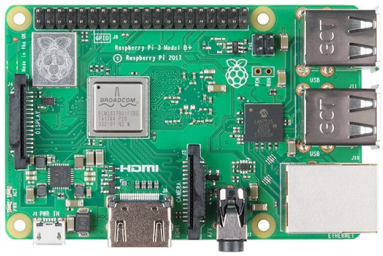


Figure B.1: Raspberry Pi 3B+, Google Images.

The input/output abilities of the Raspberry Pi must be considered. Per its specifications, the BCM2837 SoC contains the following peripherals "which may safely be accessed by the ARM [processor]": timers, interrupt controller, GPIO, USB, I2S, I2C master, I2C/SPI slave, SPI0, SP11, SPI2, PWM, and two UART busses [6]. A useful diagram that illustrates some of these features is included in Figure B.2 [32].

The timers and interrupt controller are adequate to ensure that whatever measurement tasks we intend to perform will be implemented correctly and on time. The 40 general input/output (GPIO) pins are a crucial component of the Raspberry Pi design that allows us to interface with a slew of sensors and communication devices. A GPIO pin is, by definition, a pin that does not have a specific function. The function of a GPIO pin is customizable and can be controlled by software. In the case of the Raspberry Pi, the BCM2837 architecture employs a subset of the GPIO pins to implement various data and communications protocols. For instance, some of the GPIO pins

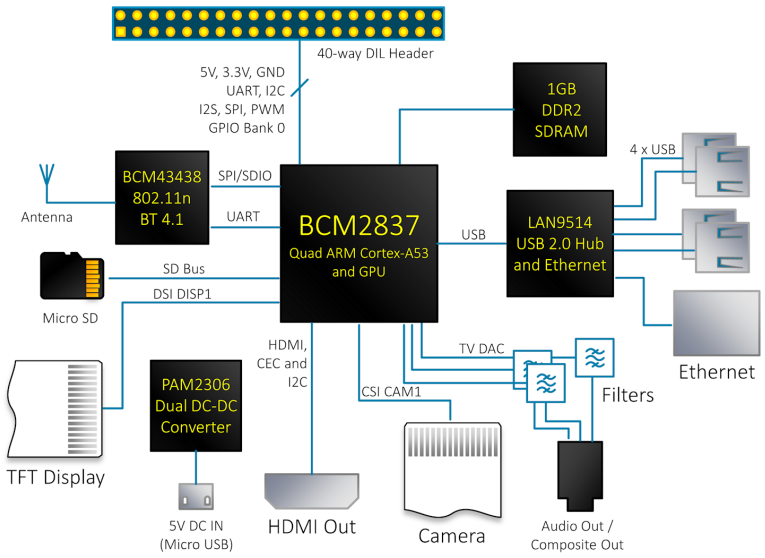


Figure B.2: Raspberry Pi 3B+ peripherals block diagram [32].

are dedicated to I2C/SPI functionality. This communication protocol is particularly useful for interfacing with many digital sensors and peripherals, like accelerometers. Some GPIO pins are also dedicated to a UART bus (there exist two on the Raspberry Pi 3B+), which is a physical circuit on the Raspberry Pi board that transmits and receives serial data, thus allowing the Raspberry Pi to interface with various serial communication devices, such as Bluetooth and Wi-Fi. Some of these various GPIO functions can be seen plainly in Figure B.3. For instance, Pin 2 outputs DC power at 5V, and pins 8 and 10 implement a pair of serial transmitting/receiving, which is used by one of the two UART busses.

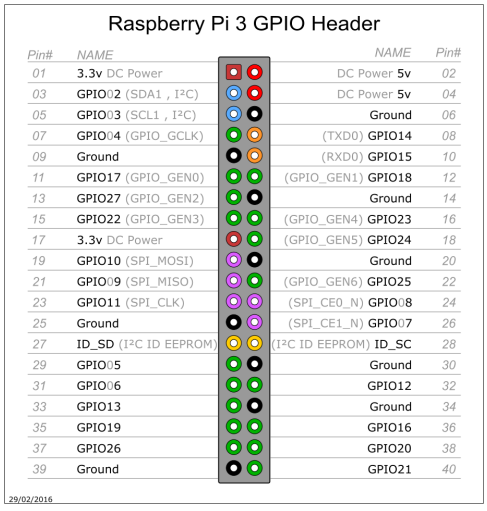


Figure B.3: Raspberry Pi 3B+ GPIO diagram.

Note: the Raspberry Pi 3B+ has "built-in" Bluetooth Low Energy and Wi-Fi capabilities. These features are actually implemented not by the BCM SoC but through a separate communications chip, the Cypress CYW43455 802.11ac and Bluetooth/Bluetooth Low Energy combo chip. This separate SoC is connected to the BCM SoC via UART data lines, as evidenced by the functional block diagram found in the Cypress CYW43455 documentation [19] and reproduced in Figure B.4.

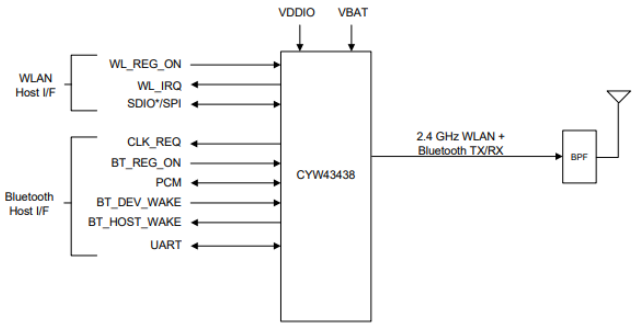


Figure B.4: Cypress communications chip block diagram [19].

The Raspberry Pi 3B+ contains its own Linux distribution as its operating system. Because of this Linux OS, specifically called "Raspbian," users can easily program on the Raspberry Pi using high-level programming languages such as Python. This is perhaps one of the most appealing features of the Raspberry Pi, as all of the useful scientific and computational code packages that are supported through open-source Python ultimately allow a user to make the most out of the various resources a Raspberry Pi offers in a reasonable amount of time without extensive training. For instance, we are able to employ the computer vision Python module, OpenCV.

The last crucial detail that must be considered is the expected power consumption of the Raspberry Pi. It would be impractical to choose a computing module that would not be deployable due to unreasonable power consumption. To investigate this, we initially examined online resources on the empirical power consumption of various Raspberry Pi models running under different conditions. Nominal power consumption is tabulated in Table B.1.

Table B.1: Nominal Raspberry Pi baseline power consumptions [33].

Raspberry Pi Model	Power Consumption	
	Idle	400% CPU Load
3B+	350 mA (1.9 W)	950 mA (5.0 W)
3B	260 mA (1.4 W)	730 mA (3.7 W)
2B	220 mA (1.1 W)	400 mA (2.1W)

Choosing the Raspberry Pi 3B+ versus an earlier model resulted in higher baseline power consumption, where the baseline is defined as the Raspberry Pi board running without any attached peripherals such as a keyboard or mouse [33]. The key advantage of the Raspberry Pi 3B and 3B+

is that they both have built-in Bluetooth and WiFi capabilities. For either a laboratory or outdoor deployment, these wireless communication options are key. We opted for the 3B+ as it is the latest and most updated board. We disregarded its increased clock speed, which may be useful for higher sampling rates. We acknowledge that while the Raspberry Pi has immense online support and documentation, the quality and reliability of that documentation and support may be sub-par; therefore, we selected the most updated board to take advantage of any revisions made by the developers at the official Raspberry Pi Foundation that may not be explicitly documented in both official and unofficial sources.

Appendix C: Micro-Controller Selection: Arduino

As opposed to the Raspberry Pi, which is essentially a computer, the Arduino is a micro-controller. A system-on-chip like the Raspberry Pi may actually contain a micro-controller as one of its components. The micro-controller must have a processor that has access to memory and various input/output peripherals. It does not have an operating system and does not have the large amount of memory and storage, as does the Raspberry Pi. What the Arduino makes up for in lack of features and built-in hardware is its low power consumption and accessibility to lower-level programming of its hardware. A typical Arduino (model: UNO) consumes around 15 mA of current at about 5 VDC. This is significantly less than the Raspberry Pi 3B+. The Arduino UNO Rev3, see Figure C.1, features the ATmega328P processor, an 8-bit processor that includes timers and interrupt capabilities, 32 kB of programmable flash memory, a real-time clock with separate oscillator, various input/output pins, and a 10-bit ADC. It also supports master/slave SPI serial communication, I2C serial communication, and UART functionality.

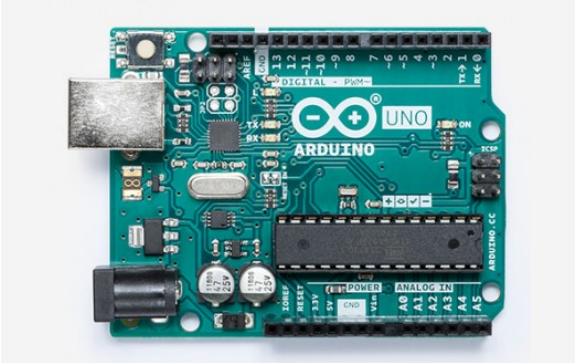


Figure C.1: Arduino UNO Rev3.

The key operational difference between the two computational units is that the Raspberry Pi has many layers of abstraction between the user and the hardware as provided by the operating system. On the Arduino, the user can code in a variant of C and C++, and directly compile the code onto the micro-controller itself. These low-level languages allow the user to directly control tasks

such as memory allocation and task scheduling. Even at this level, Wang et al. (2007) were able to perform many complex mathematical procedures on the very same micro-controller that the Arduino uses. This proved to be a disadvantage in this project. Again, we emphasize that this project was not aimed at pushing the limits of various hardware options but developing an application of embedded systems designed to deploy an existing measurement method for monitoring purposes. The Raspberry Pi, with its many layers of abstraction through the helpful operating system and support for Python programming, is very appealing, compared to the low-level functionalities of the Arduino.

Note: the Raspberry Pi nominally consumes over 20 times the amount of current that the Arduino consumes. For long-term monitoring purposes, adequate power delivery is a significant challenge. Moreover, the Raspberry Pi, does not completely "shutdown" unlike an actual computer. Even when turned off, the board still consumes power, driving various LEDs and low-level components. The only way to ensure the Raspberry Pi consumes no power is to cut its power supply. We therefore employed both computing units for this project, as illustrated in a subset of Figure 2.1 and repeated in Figure 2.3 for clarity. The Arduino, due to its ability to interface with various peripherals and perform basic processing based on inputs from those peripherals, was designated as controlling the power supply to the Raspberry Pi. The Arduino delivered power to the Raspberry Pi only when the Raspberry Pi was taking measurements; for most of the system's operation, the Raspberry Pi is barely on. More details on power consumption are included in Section 2.3.4.

Appendix D: Phase II: Enclosure Design

The specifications of the exterior enclosure was designed to meet the conditions needed in field applications of the system. After prototyping with 3D printed enclosures, we settled on an electrical box from a local supplier as it is designed for outdoor use and has a 3R NEMA rating. This box provided a degree of protection for the electronics enclosed, and the knockouts as well as padlockable latch offered security for the components. The enclosure has outer dimensions of $8\times8\times6$ in.; we used identical boxes for the camera module and laser module.

In regards to the interior of the enclosure, we designed a custom mounting fixture, dubbed affectionately the "cradle," to affix the hardware to, which was then dropped into the metal box. The custom, quarter-inch-thick acrylic plate was designed to accommodate the mounting of all the parts and replicate the camera chamber within the camera box by leaving a clear field of view from the camera to the acrylic screen. The location of the screen on the lid of the camera box was aligned with the camera, and the camera was installed a prescribed distance from the screen to best replicate the camera chamber used in prototyping in the camera box. On the laser side, a two-axis laser mount was custom designed to allow a field technician to aim the laser at the screen. This design was found to be more secure and durable than off-the-shelf mounts intended for indoor use. Additionally, all of the wiring was secured and bare connections were protected using heat-shrink tubing. Any exterior wiring was protected using conduit tubing.

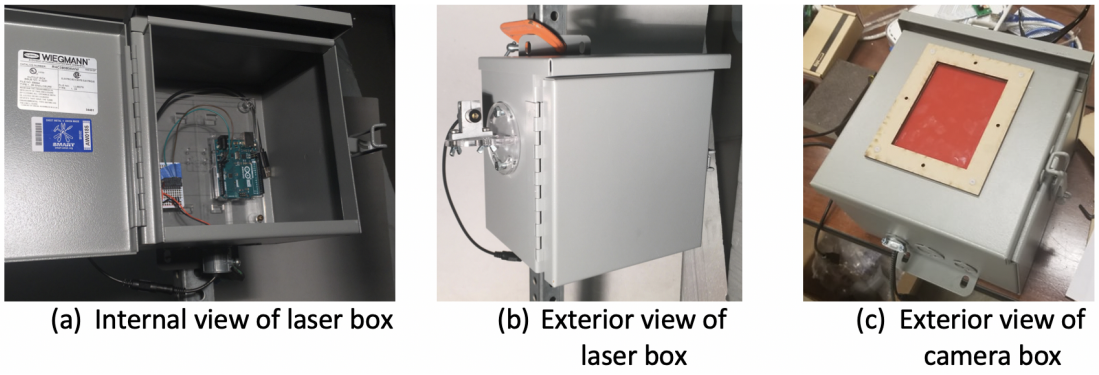


Figure D.1: Phase II: internal and external views of camera and laser enclosures.

Figure D.1 shows the custom cut acrylic laser mount, which can swivel in order to allow the technician to aim the laser beam. On top of this fixture, we attached the purchased laser mount, whose hole was enlarged to accommodate the laser diode. Also note the mounting holes on the top of bottom flaps of the enclosures.

The enclosure was designed for ease of installation, and the cradle was setup to be dropped into the metal box with minimal adjustments necessary. In the field, the technician is expected to have the means for mounting the two enclosures to the bridge piers with concrete screws passing through the top and bottom eye-holes of the enclosures, which are expected to be aligned such that the laser beam has a clear line of sight to the camera screen. After securing both the enclosures to the bridge piers, the field technician powers up the laser independently from the power delivery system of the laser module and then adjusts the line of sight of the laser to align the cross-hairs on the screen of the camera module. Any final adjustments should be made to secure the solar panel, as well as any peripheral antennas extending from the box. After ensuring that the lid is free of wires and is not obstructed by any parts, the technician closes the latch and affixes an appropriate padlock to prevent any tampering with the internal system.

Appendix E: Computer Vision Algorithm and Server/Website

Although this report was not concerned with the development of the computer vision algorithm, it is discussed briefly in Section 2.3.5; some of its limitations when deployed using our monitoring system are discussed in Section 3.3. In addition, the server and website were also not discussed in detail herein beyond general functionality. Below is a brief discussion of how the computer vision algorithm's outputs interacted with the server and website, and how to navigate and use the website developed for this project. The website serves as a user interface for users to configure, retrieve data from, and transmit signals using the server.

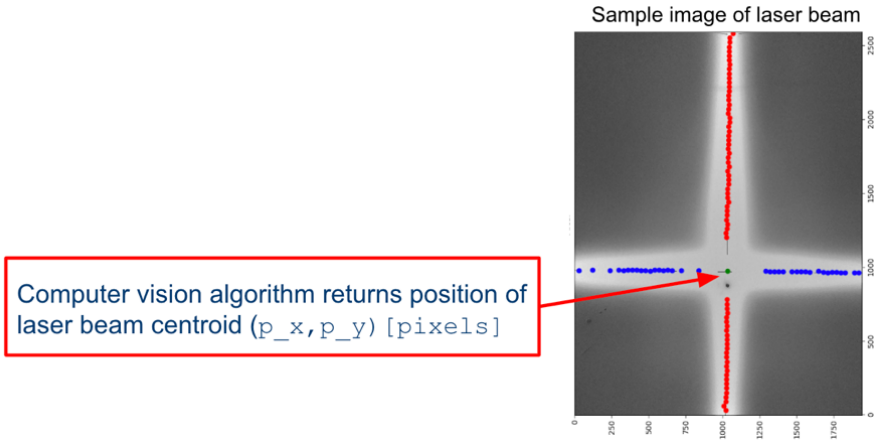


Figure E.1: Computer vision algorithm outputs.

The computer vision algorithm uses image subtraction to correct for light artifacts. After de-noising the image and applying a Sobel operator [34] to the pixel values of the image, it fits two regression curves to selected regression points in the image domain. The intersection of these regression curves is the desired pixel coordinates of the centroid of the laser beam, $[p_x, p_y]$; see Figure E.1. These coordinates are then passed to the camera module program as data and transmitted to the server along with other information. This server was written using the Django framework and information is pushed to it from the camera module using HTTP put requests. At all times, the camera module has hard-coded settings, which include information such as a "name" for that particular sensor pair, the URL address for the server, and the password for access into the server. For instance, for the system that was deployed in Phase II of this project, the settings relevant to

the transmission of data to the server are:

```
BRIDGE_NAME = 'bridge-1'
SERVER = "http://apps2.peer.berkeley.edu/caltrans/sensors/...
...+BRIDGE_NAME+"/update/"
PASSWORD = "djioewfj34jod2jdoi3jr0jl983jsa"
```

We can access the website using a web browser through the URL address: `http://apps2.peer.berkeley.edu/caltrans`, which directs us to the login page, as seen in Figure E.2. To access the main page of the website requires using a particular username and a corresponding password to login as an administrator; see Figure E.3. To illustrate how to configure and use a new monitoring system, click on the "Bridges" button, which leads to a menu page; see Figure E.4. On this page, a new system can be added by clicking on the button on the top right corner. Once a new system has been initialized, we can configure its settings from the menu page by clicking on its name; see Figure E.5. On this page, we can set the phone number that corresponds with the SIM card on that system's FONA 2G cellular board. We can also set the unique calibration value for that camera module. Once this is initialized, we can return to the menu page and interact with that system by selecting it from the menu and using the drop-down menu to access certain actions; see Figure E.6. The two actions that are most important are as follows: (1) taking a measurement, which sends out an SMS message to the FONA; and (2) retrieving all of the data received from that system. Action (2) returns a .csv file that can be saved locally.

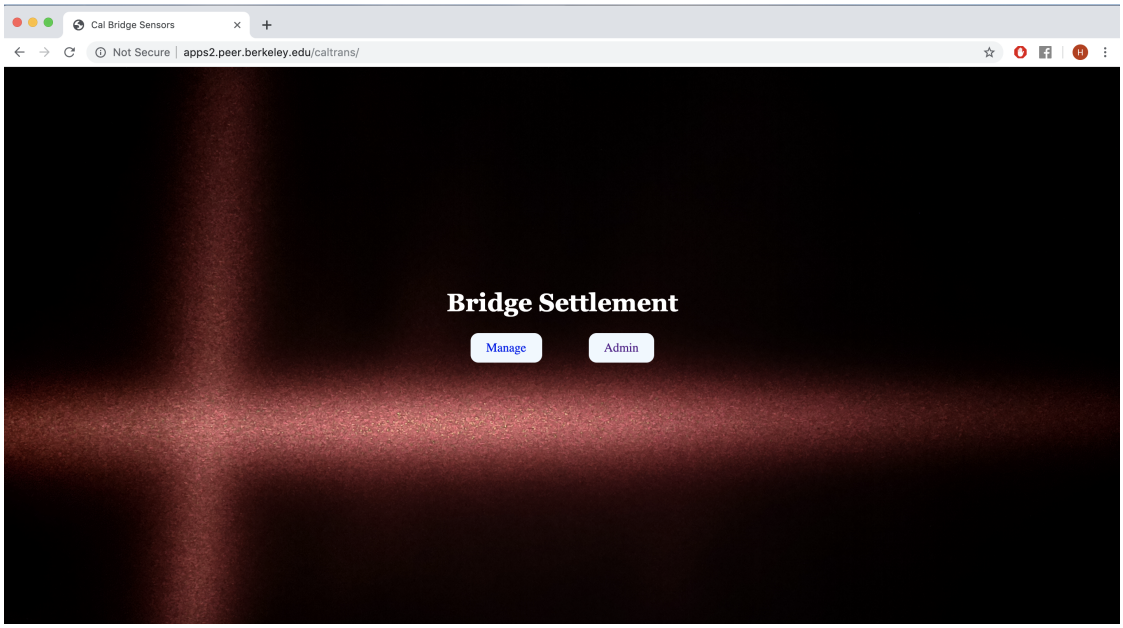


Figure E.2: Login page of website.

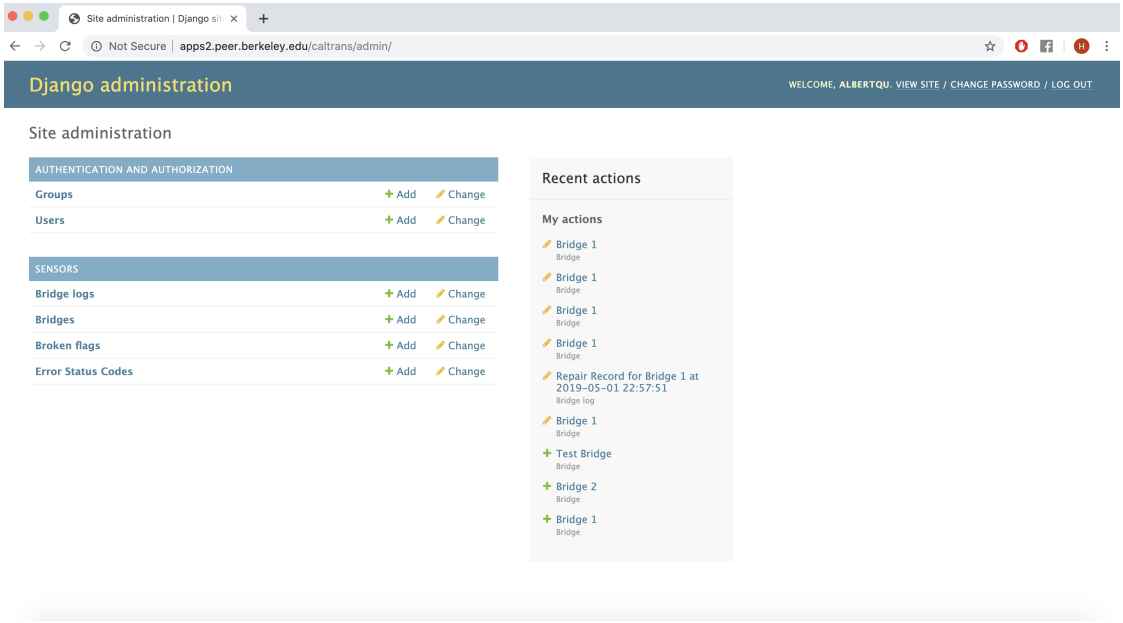


Figure E.3: Main page of website.

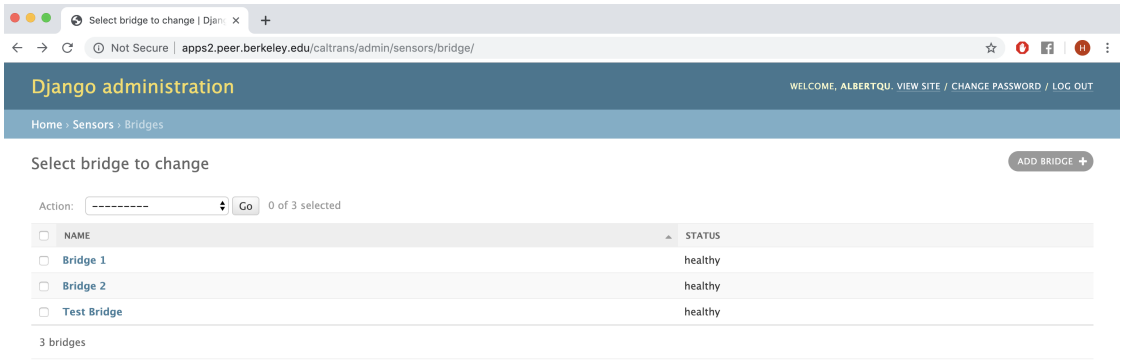


Figure E.4: "Bridges" menu page of website.

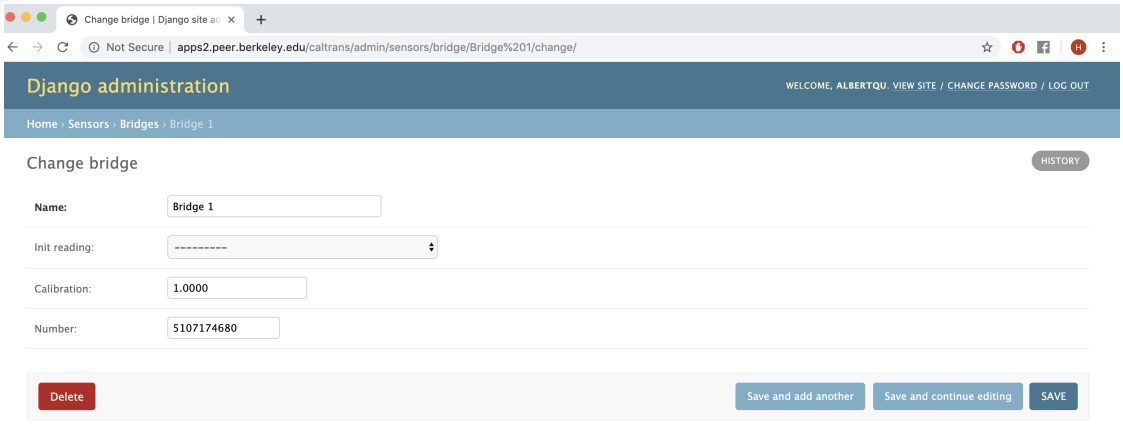


Figure E.5: System configuration page of website.

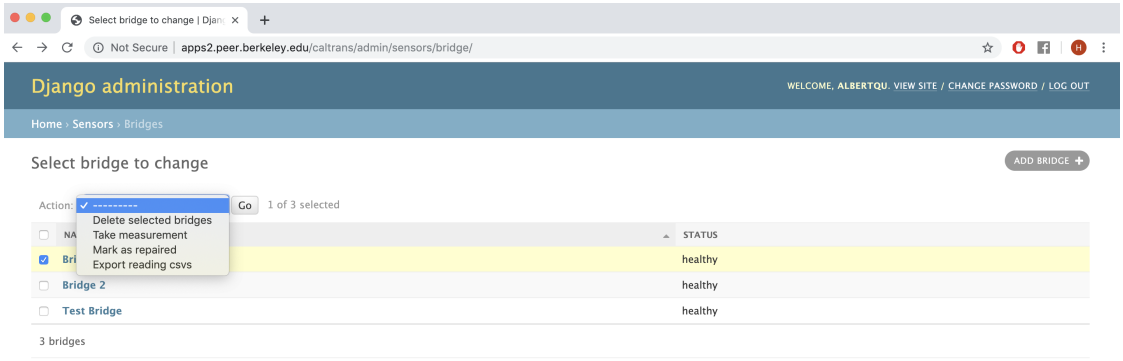


Figure E.6: Drop-down menu actions on website.

Appendix F: Phase II: Bill of Materials

Table F.1: Phase II: Bill of Materials.

Item	Description	Component	Unit Price	x	Price
Raspberry Pi 3B+		Camera module	35	1	35
32 GB SD Card	for Raspberry Pi	Camera module	10	1	10
Arduino UNO Rev3		Cam. & Laser module	20	2	40
100 mW Laser Diode		Laser module	50	1	50
Raspberry Pi Camera v1		Camera module	15	1	15
5VDC Latching DPDT Relay		Both modules	5	3	15
HM-10 BLE		Laser module	10	1	10
2G GSM Adafruit FONA		Camera module	50	1	50
2G Ting SIM card	for FONA	Camera module	5	1	5
1200 mAh 3.7V Li-Poly battery	for FONA	Camera module	15	1	15
GSM Quad Band Antenna	3 dBi uFL, for FONA	Camera module	5	1	5
uFL extension	to extend antenna	Camera module	5	1	5
Tripp Lite Class 1 BLE dongle		Camera module	15	1	15
Voltaic Systems 10W Solar Panel		Power supply, both	50	3	150
V44 12000 mAh battery pack		Power supply, both	70	2	140
Assorted USB cables & jumper cables		Both modules	20	1	20
Breadboard		Both modules	5	2	10
Metal utility box	Weighmann 8x8x6 with Latch	Enclosures, both	115	2	230
Hardware Mounting Frame (“Cradle”)	Laser Cut Acrylic	Enclosures, both	10	2	20
Acrylic Laser Mount	Laser Cut Acrylic	Las. module enclosure	5	1	5
Acrylic Screen	Opaque Red Acrylic	Cam. module enclosure	5	1	5
Farhop 12mm Laser Holder	Bored out to 13mm	Las. module enclosure	15	1	15
Assorted fasteners and hardware		Enclosures, both	20	1	20
Total price					**885**
Total for enclosures					**295**
Total for power supply					**290**
Total for embedded system					**300**

Appendix G: Additional Figures and Tables

Table G.1: Phase I deployment power delivery system.

Component	Description	Details
5V Latching DPDT Relay	A switch, controlled by Arduino, for Pi power delivery	Located on breadboard: G6AK-234P-ST-US-DC5 part
1N457 Small Signal Diode	Stops back EMF from Relay coils	Located on breadboard, left of Relay
Relay Coil Line 1	Wired from Arduino pin 13 to Relay coil 1	Can be driven ”high” or ”low” by Arduino. Located on breadboard: Brown Wire
Relay Coil Line 2	Wired from Arduino pin 12 to Relay coil 2	Can be driven ”high” or ”low” by Arduino Located on breadboard: Green wire
pin_{sync} Line	Wired from Pi GPIO18 pin to Arduino pin 2	Can be driven ”high” or ”low” by Pi Located on breadboard: White wire
pin_{power} Line	Wired from Pi GPIO23 pin to Arduino pin 3	Can be driven ”high” or ”low” by Pi Located on breadboard: Orange Wire
Two Pull Down Resistors	Pull floating voltages to GND	Located on breadboard: Orange and White wires junction
System power source	See: barrel plug feeding into top power rails	Provide power to the entire system Can source power from battery or other
Raspberry Pi power source	See: barrel plug connected to PWR from relay	PWR controlled by relay
Arduino power source	See: barrel plug connected to bottom rail	Always on.

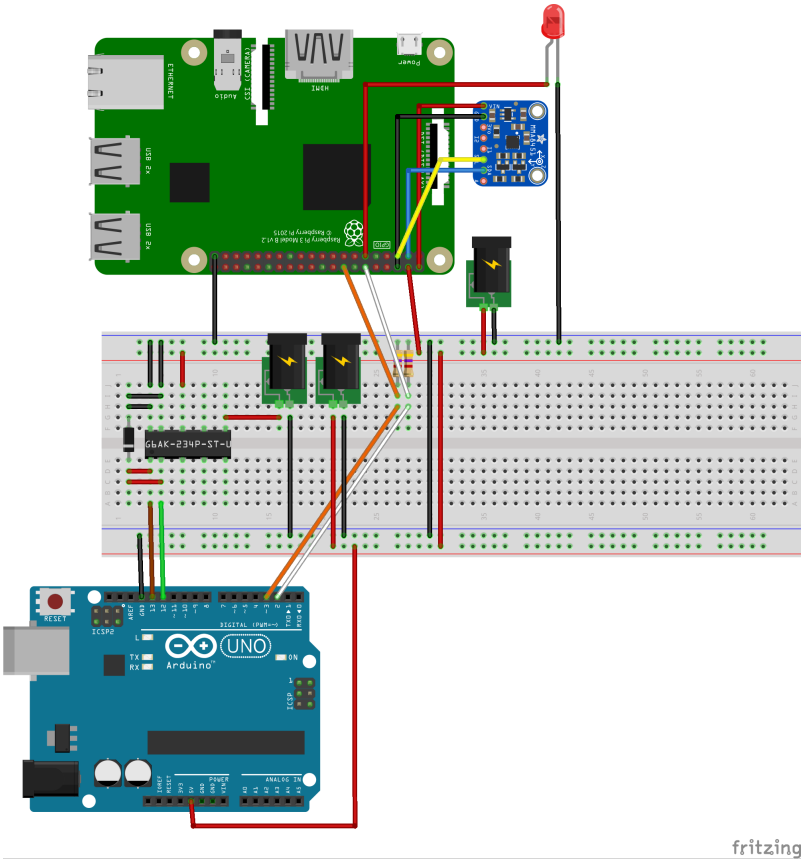


Figure G.1: Phase I power delivery system.

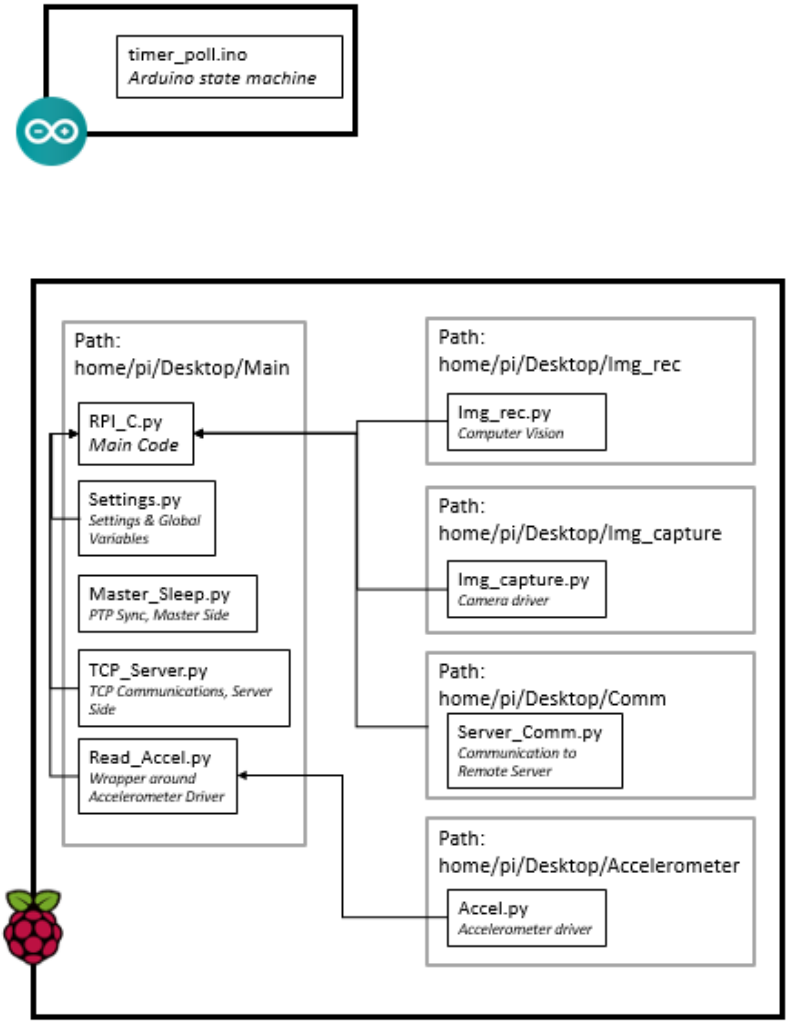


Figure G.2: Phase I file organization and dependencies, camera module

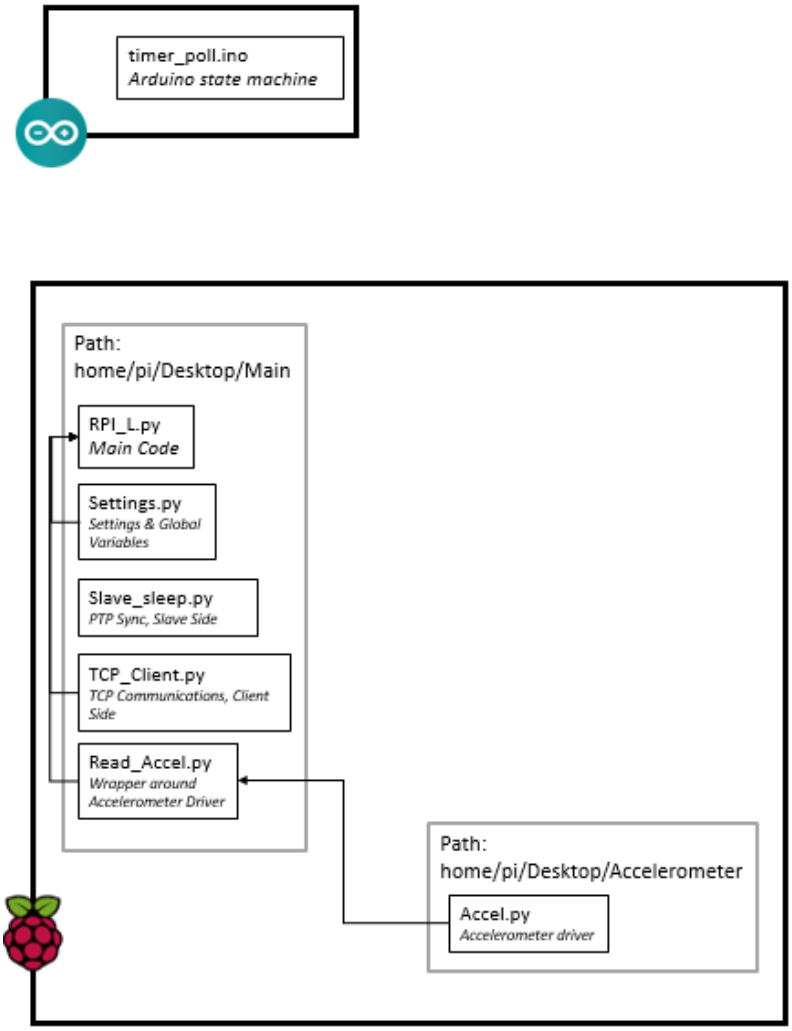
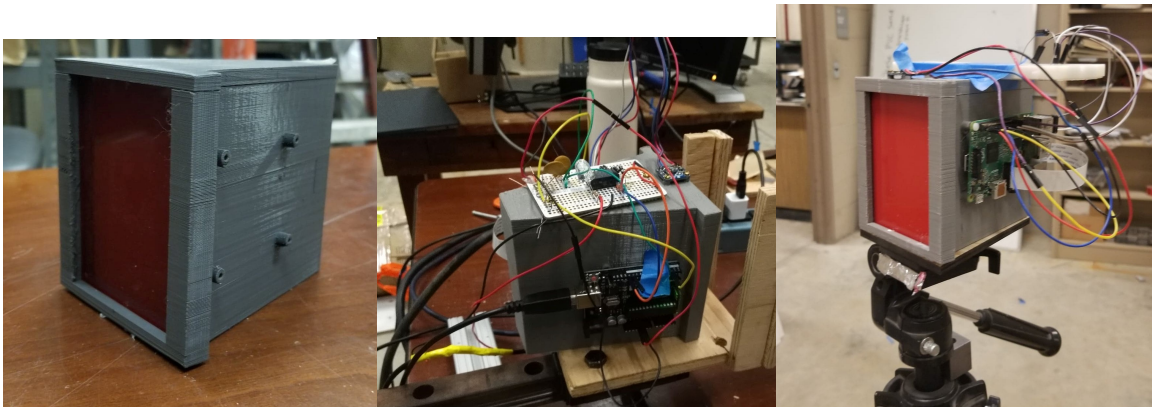
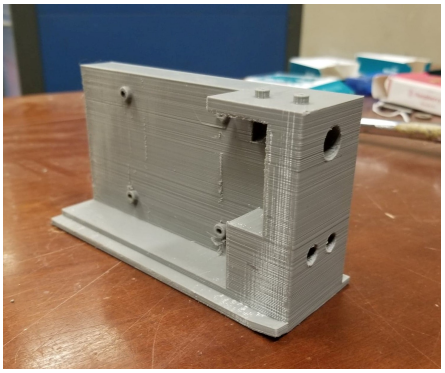


Figure G.3: Phase I file organization and dependencies, laser module

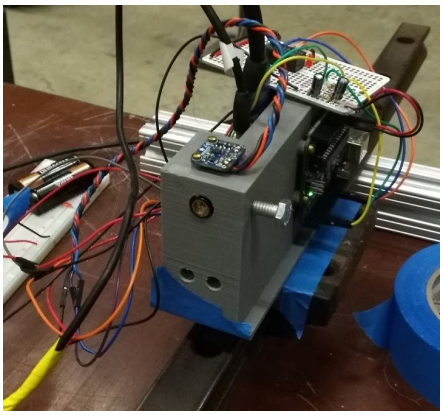


(a) Camera chamber (b) Camera chamber & hardware (c) Camera chamber on tripod

Figure G.4: Version 1 camera module.



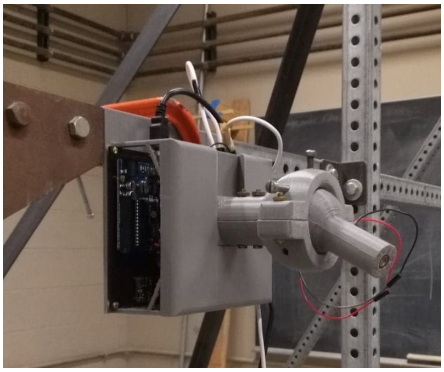
(a) Laser mounting



(b) Laser mounting, with hardware

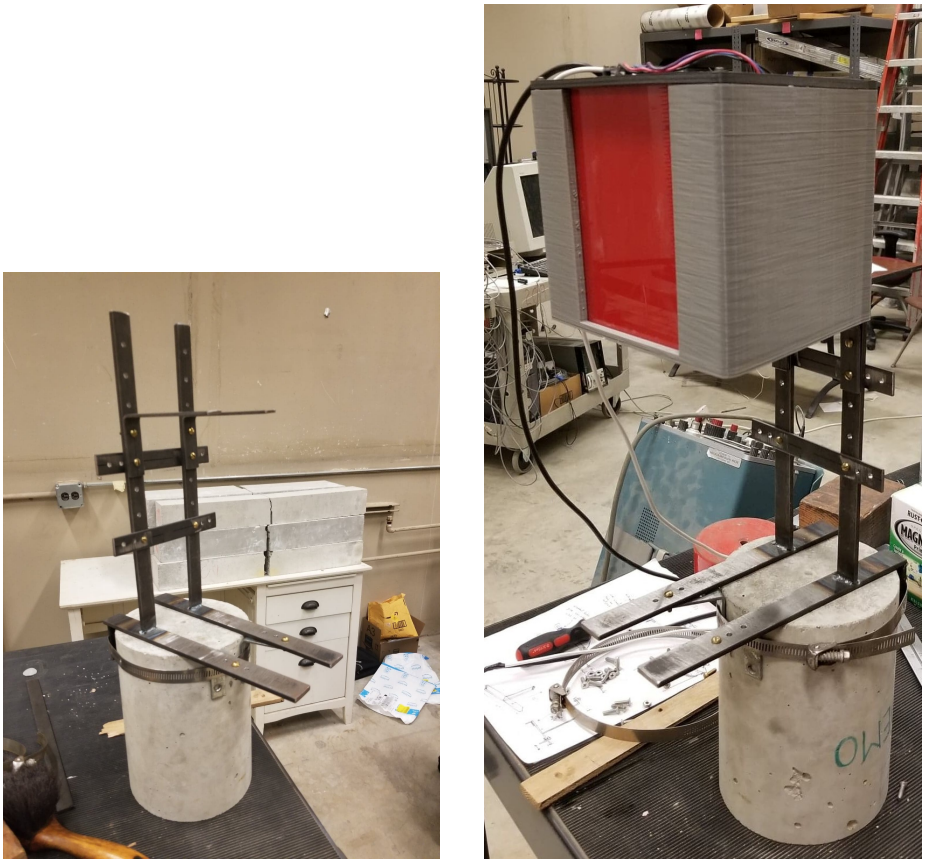
Figure G.5: Version 1 laser module.

(a) Camera chamber



(b) Laser mounting

Figure G.6: Version 2 modules



(a) Specimen mount for camera

(b) Camera mounted on specimen

Figure G.7: Possible specimen mounting configuration.

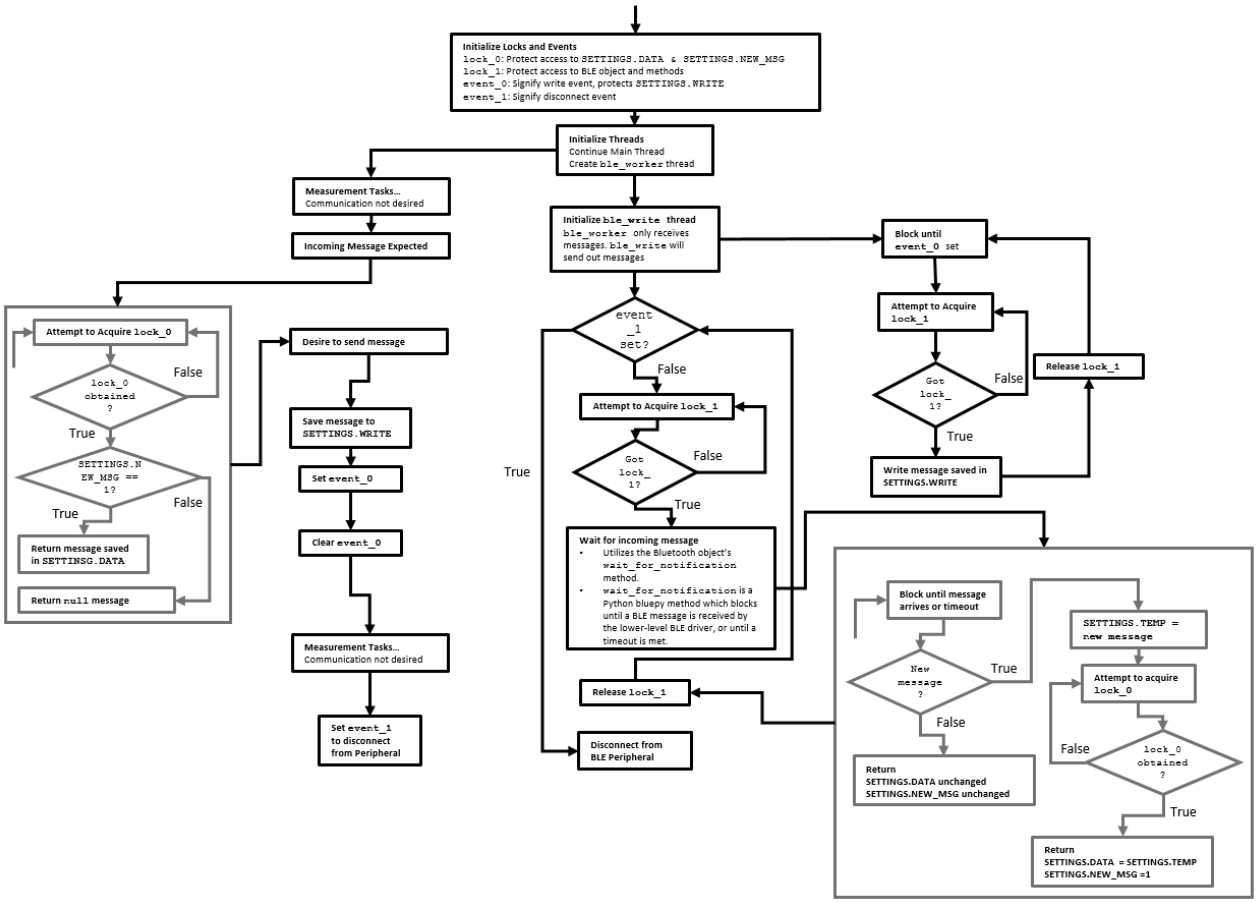


Figure G.8: Asynchronous BLE communications.

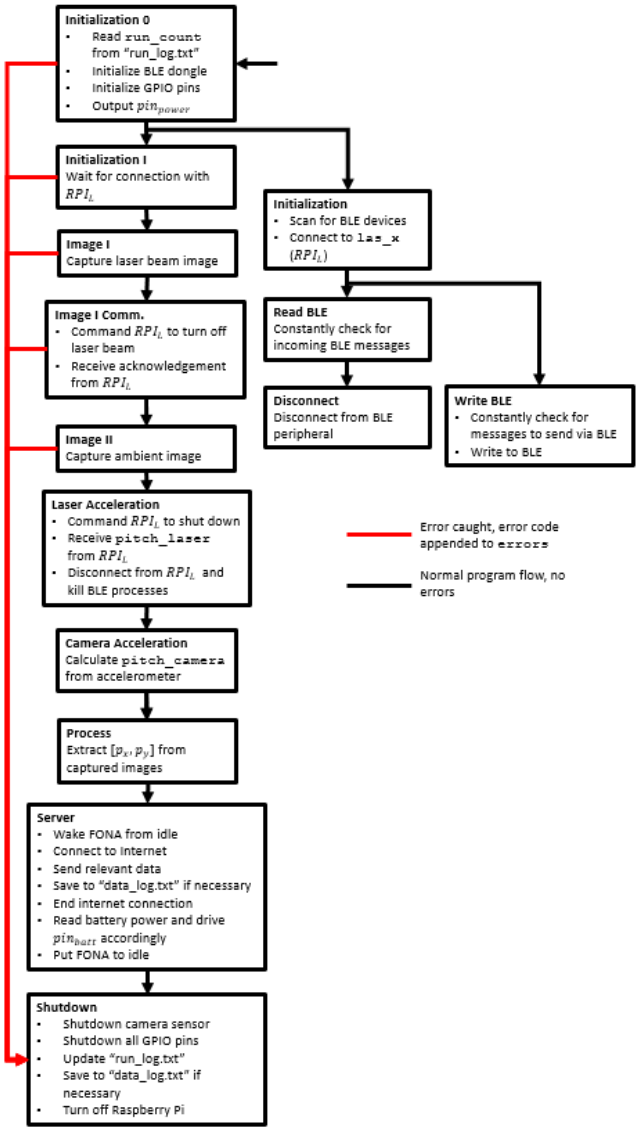


Figure G.9: Phase II deployment: Raspberry Pi camera module transition diagram.

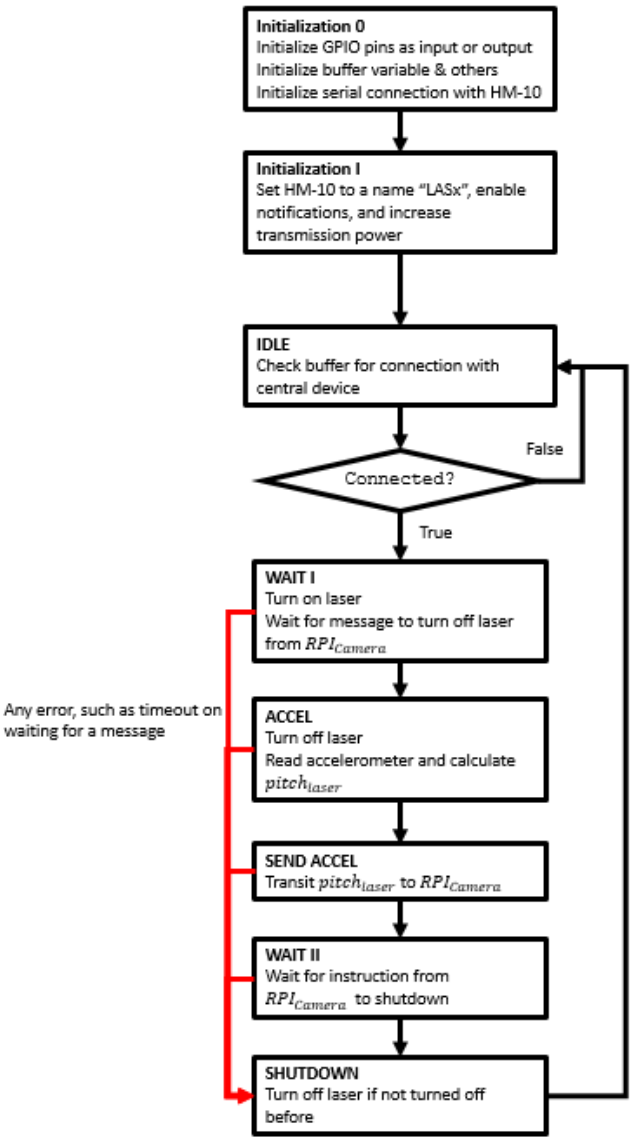


Figure G.10: Phase II deployment: Arduino laser module state diagram.

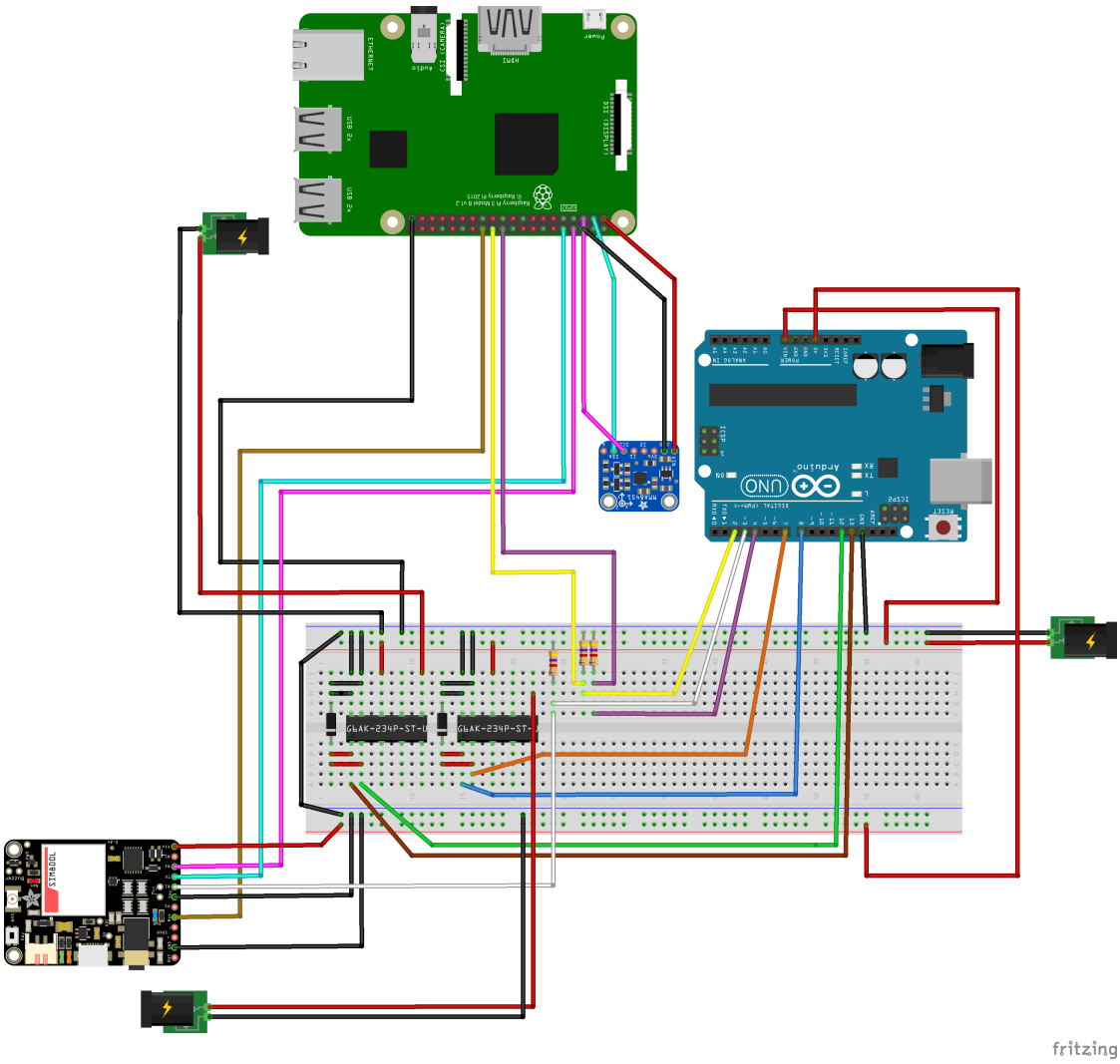


Figure G.11: Phase II deployment: power delivery system for camera module.

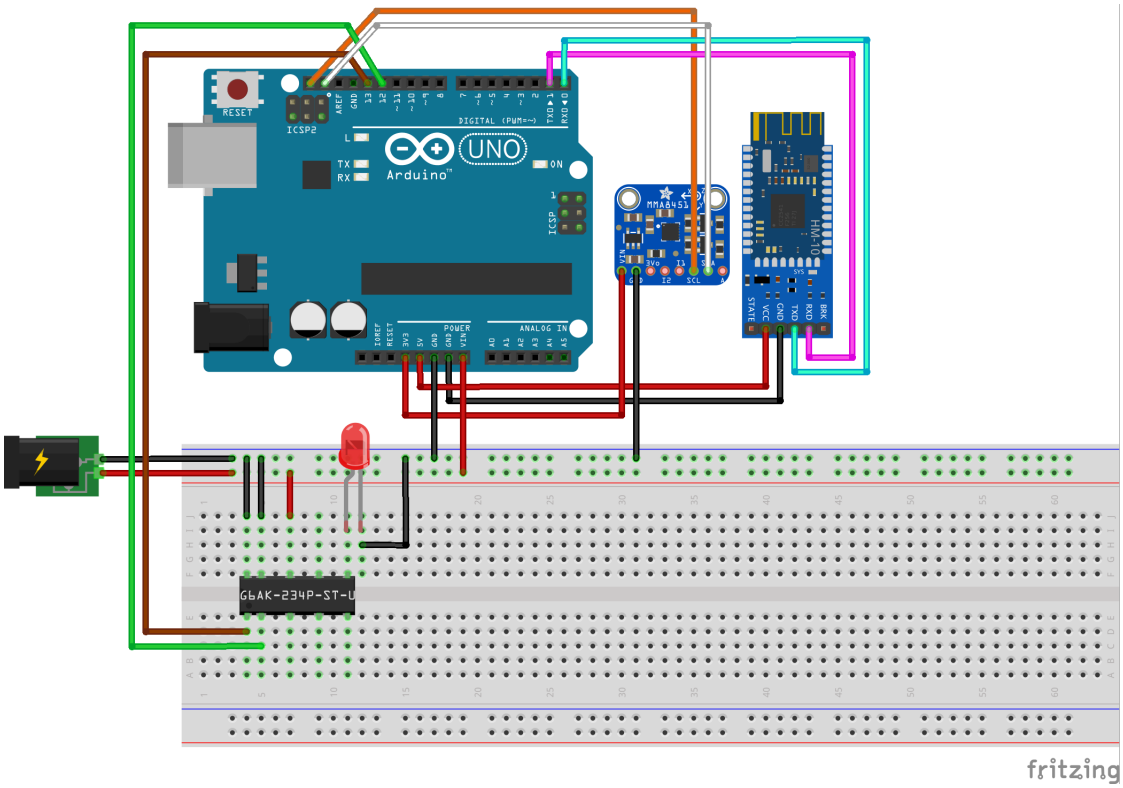


Figure G.12: Phase II deployment: power delivery system for laser module.

Table G.2: Phase II enumerated system error codes (transmitted by camera module).

1-10: Sensor/GPIO related 11-25: Communication related 26-30: Image algorithm related 31+: Other		
Error Code	**Origin State**	**Description**
1	Pre-INIT_0	GPIO Pin Error
2	IMAGE_I	Camera Malfunction
3	IMAGE_II	Camera Malfunction
4	C_ACCEL	Accelerometer Malfunction
5	SERVER	Battery pin reading error
6	SERVER	Battery pin GPIO error
11	INIT_I	No connection to RPI_L
12	IMAGE_I_COMM	Wrong Ack. from RPI_L
13	IMAGE_I_COMM	No msg received from RPI_L
14	L_ACCEL	No msg received from RPI_L
15	L_ACCEL	pitch_L data type error suspected
16	SERVER	Runtime Warning: Value not pushed to server
17	SERVER	Value not pushed to server
18	SERVER	No active connection FONA not put to sleep
19	SERVER	No active connection FONA put to sleep
20	INIT_0	Could not initialize BLE dongle
21	SERVER	SMS messages not deleted
26	PROCESS	Image Processing Error
31	Pre-INIT_0	Run File Error
32	INIT_1	General error
33	IMAGE_COMM_COMM	General error
34	L_ACCEL	General error
35	L_ACCEL	pitch_L received not numeric
36	SERVER	Active connection made FONA not put to sleep
37	L_ACCEL	Error terminating BLE processes

Table G.3: Example of error catching with actions and corresponding data log.

Actions performed	Corresponding data log
Recall `data = [run_count, p_x, p_y, pitch_C, pitch_L, errors]` In this test, hardcoded: `[p_x,p_y] = [1,2]`	
unplugged accelerometer unplugged FONA only RPI_{laser} unplugged RPI_{laser} unplugged during first photo RPI_{laser} unplugged during second photo RPI_{laser} has unplugged accelerometer unplugged FONA only *Note: all above actions also include unplugging FONA*	0,1,2,None,17.2557430000,4,18 1,1,2,0.327253738789,17.1295340000,18 2,None,None,None,None,11 3,None,None,None,None,13 4,1,2,0.326877493335,11.0077240000,18 5,1,2,0.323774445807,None,14,18 6,1,2,0.330329407426,None,14,18 7,1,2,0.332966050035,11.0066180000,18
Plugged in FONA	data log cleared
Unplugged FONA and moved the RPI_{laser} far away	9,None,None,None,None,11 10,None,None,None,None,11 11,1,2,0.338712969875,11.0003760000,18 *Note: runs 9 and 10 could not connect*
Plugged in FONA	data log cleared

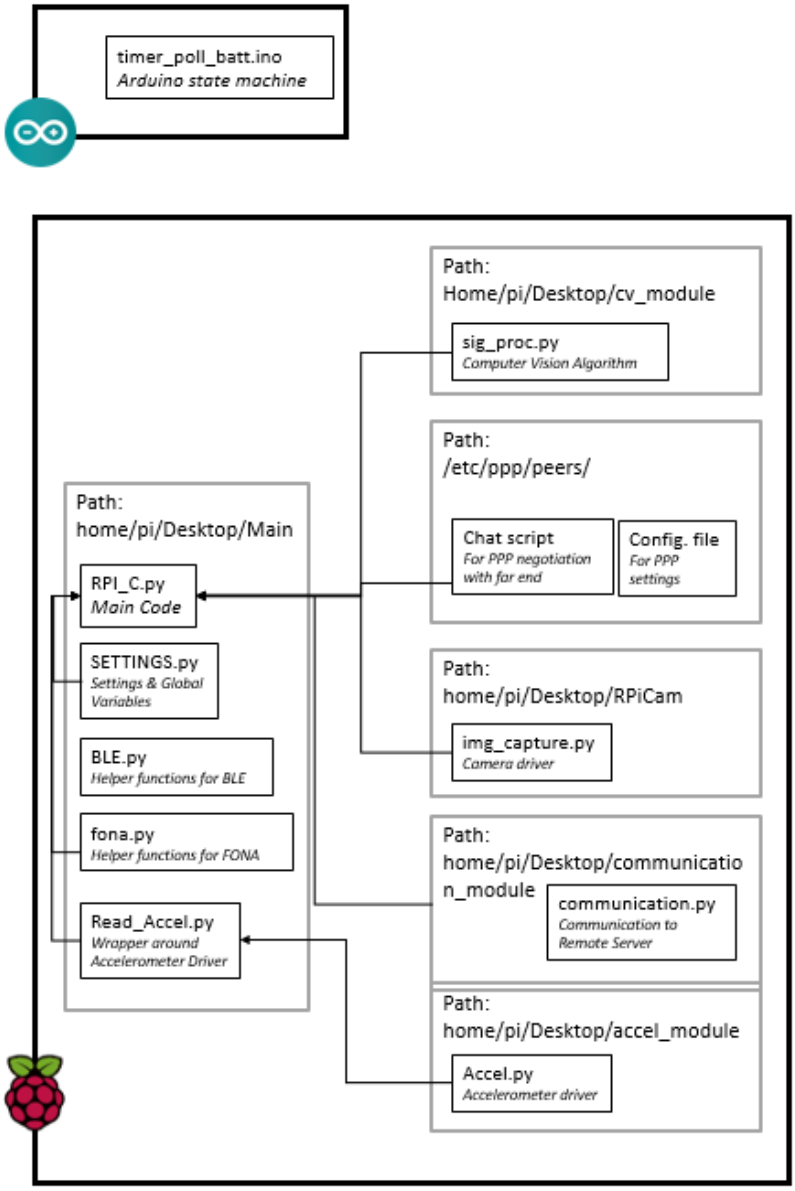


Figure G.13: Phase II: file organization on the Raspberry Pi in the camera module.

Appendix H: Codes

All Phase I and Phase II codes can be accessed through Github:
`https://github.com/hteng1995/peer_caltrans_bridge_settlement`

PEER REPORTS

PEER reports are available as a free PDF download from https://peer.berkeley.edu/peer-reports. In addition, printed hard copies of PEER reports can be ordered directly from our printer by following the instructions at https://peer.berkeley.edu/peer-reports. For other related questions about the PEER Report Series, contact the Pacific Earthquake Engineering Research Center, 325 Davis Hall, Mail Code 1792, Berkeley, CA 94720. Tel.: (510) 642-3437; and Email: peer_center@berkeley.edu.

PEER 2020/25 *Regionalized Ground-Motion Models for Subduction Earthquakes based on the NGA-SUB Database*. Norman Abrahamson and Zeynep Güerce, December 2020.

PEER 2020/24 *Seismic Performance of Single-Family Wood-Frame Houses: Comparing Analytical and Industry Catastrophe Models* (PEER-CEA Project). Evan Reis. December 2020.

PEER 2020/23 *Earthquake Damage Workshop* (PEER-CEA Project). Kylin Vail, Bret Lizundia, David P. Welch, and Evan Reis. December 2020.

PEER 2020/22 *Technical Background Report for Structural Analysis and Performance Assessment* (PEER-CEA Project). David P. Welch and Gregory G. Deierlein. December 2020.

PEER 2020/21 *Comparison of the Response of Small- and Large-Component Cripple Wall Specimens Tested under Simulated Seismic Loading* (PEER-CEA Project). Brandon Schiller, Tara Hutchinson, and Kelly Cobeen. December 2020.

PEER 2020/20 *Large-Component Seismic Testing for Existing and Retrofitted Single-Family Wood-Frame Dwellings* (PEER-CEA Project). Kelly Cobeen, Vahid Mahdavifar, Tara Hutchinson, Brandon Schiller, David P. Welch, Grace S. Kang, and Yousef Bozorgnia. December 2020.

PEER 2020/19 *Cripple Wall Small-Component Test Program: Comparisons* (PEER-CEA Project). Brandon Schiller, Tara Hutchinson, and Kelly Cobeen. December 2020.

PEER 2020/18 *Cripple Wall Small-Component Test Program: Wet Specimens II* (PEER-CEA Project). Brandon Schiller, Tara Hutchinson, and Kelly Cobeen. December 2020.

PEER 2020/17 *Cripple Wall Small-Component Test Program: Dry Specimens* (PEER-CEA Project). Brandon Schiller, Tara Hutchinson, and Kelly Cobeen. December 2020.

PEER 2020/16 *Cripple Wall Small-Component Test Program: Wet Specimens I* (PEER-CEA Project). Brandon Schiller, Tara Hutchinson, and Kelly Cobeen. December 2020.

PEER 2020/15 *Development of Testing Protocol for Cripple Wall Components* (PEER-CEA Project). Farzin Zareian and Joel Lanning. December 2020.

PEER 2020/14 *Probabilistic Seismic Hazard Analysis and Selecting and Scaling of Ground-Motion Records* (PEER-CEA Project). Silvia Mazzoni, Nicholas Gregor, Linda Al Atik, Yousef Bozorgnia, David P. Welch, Gregory G. Deierlein. December 2020.

PEER 2020/13 *Development of Index Buildings* (PEER-CEA Project). Evan Reis. December 2020.

PEER 2020/12 *Project Technical Summary* (PEER-CEA Project). Evan Reis,in collaboration with Yousef Bozorgnia, Henry Burton, Kelly Cobeen, Gregory G. Deierlein, Tara Hutchinson, Grace S. Kang, Bret Lizundia, Silvia Mazzoni, Sharyl Rabinovici, Brandon Schiller, David P. Welch, and Farzin Zareian. December 2020.

PEER 2020/11 *Hybrid Simulations for the Seismic Evaluation of Resilient Highway Bridge Systems*. Yingjie Wu, Selim Gunay, and Khalid M. Mosalam. November 2020**.**

PEER 2020/10 *Low Seismic Damage Columns for Accelerated Bridge Construction*. Arpit Nema and Jose I. Restrepo. November 2020**.**

PEER 2020/09 *Blind Prediction of Shaking Table Tests of a New Bridge Ben Design.* Selim Gunay, Fan Hu, Khalid Mosalam, Arpit Nema, Jose Restrepo, Adam Zsarnoczay, and Jack Baker, November 2020.

PEER 2020/08 *PEER Activities Report 2018_2020.* Khalid M. Mosalam and Amarnath Kasalanati, November 2020.

PEER 2020/07 *Comparison of NGA-Sub Ground-Motion Models.* Nicholas Gregor, Kofi Addo, Linda Al Atik, David M. Boore, Yousef Bozorgnia, Kenneth W. Campbell, Brian S.-J. Choiu, Zeynep Gülerce, Behzad Hassani, Tadahiro Kishida, Nico Kuehn, Saburoh Midorikawa, Silvia Mazzoni, Grace A. Parker, Hongjun Si, Jonathan P. Stewart, and Robert R. Youngs. November 2020.

PEER 2020/06 *Development of NGA-Sub Ground-Motion Model of 5%-Damped Pseudo-Spectral Acceleration based on Database for Subduction Earthquakes in Japan.* Hongjun Si, Saburoh Midorikawa, and Tadahiro Kishida. November 2020.

PEER 2020/05 *Conditional Ground-Motion Model for Peak Ground Velocity for Active Crustal Regions.* Norman A. Abrahamson and Sarabjot Bhasin. October 2020.

PEER 2020/05 *Conditional Ground-Motion Model for Peak Ground Velocity for Active Crustal Regions.* Norman A. Abrahamson and Sarabjot Bhasin. October 2020.

PEER 2020/04 *Partially Non-Ergodic Ground-Motion Model for Subduction Regions using the NGA-Subduction Database.* Nicolas Kuehn, Yousef Bozognia, Kenneth W. Campbell, and Nicholas Gregor. September 2020.

PEER 2020/03 *NGA-Subduction Global Ground-Motion Models with Regional Adjustment Factors.* Grace A. Parker, Jonathan P. Stewart, David M. Boore, Gail M. Atkinson, and Behzad Hassani. September 2020.

PEER 2020/02 *Data Resources for NGA-Subduction Project.* Yousef Bozorgnia (PI) and Jonathan P. Stewart (Editor). March 2020.

PEER 2020/01 *Modeling Viscous Damping in Nonlinear Response History Analysis for Steel Moment-Frame Buildings.* Xin Qian, Anil K. Chopra, and Frank McKenna. June 2020.

PEER 2019/09 Seismic Behavior of Special Concentric Braced Frames under Short- and Long-Duration Ground Motions. Ali Hammad and Mohamed A. Moustafa. December 2019.

PEER 2019/08 Influence of Vertical Ground Motion on Bridges Isolated with Spherical Sliding Bearings. Rushil Mojidra and Keri L. Ryan. December 2019.

PEER 2019/07 *PEER Hub ImageNet* (*ϕ*-Net): *A Large-Scale Multi-Attribute Benchmark Dataset of Structural Images*. Yuqing Gao, and Khalid. M. Mosalam. November 2019.

PEER 2019/06 *Fluid-Structure Interaction and Python-Scripting Capabilities in OpenSees*. Minjie Zhu and Michael H. Scott. August 2019.

PEER 2019/05 *Expected Earthquake Performance of Buildings Designed to the California Building Code (California Alfred E. Alquist Seismic Safety Publication 19-01).* Grace S. Kang, Sifat Muin, Jorge Archbold, Bitanoosh Woods, and Khalid Mosalam. July 2019.

PEER 2019/04 *Aftershock Seismic Vulnerability and Time-Dependent Risk Assessment of Bridges*. Sujith Mangalathu, Mehrdad Shokrabadi, and Henry V. Burton. May 2019.

PEER 2019/03 *Ground-Motion Directivity Modeling for Seismic Hazard Applications.* Jennifer L. Donahue, Jonathan P. Stewart, Nicolas Gregor, and Yousef Bozorgnia. Review Panel: Jonathan D. Bray, Stephen A. Mahin, I. M. Idriss, Robert W. Graves, and Tom Shantz. May 2019.

PEER 2019/02 *Direct-Finite-Element Method for Nonlinear Earthquake Analysis of Concrete Dams Including Dam–Water–Foundation Rock Interaction.* Arnkjell Løkke and Anil K. Chopra. March 2019.

PEER 2019/01 *Flow-Failure Case History of the Las Palmas, Chile, Tailings Dam.* R. E. S. Moss, T. R. Gebhart, D. J. Frost, and C. Ledezma. January 2019.

PEER 2018/08 *Central and Eastern North America Ground-Motion Characterization: NGA-East Final Report*. Christine Goulet, Yousef Bozorgnia, Norman Abrahamson, Nicolas Kuehn, Linda Al Atik, Robert Youngs, Robert Graves, and Gail Atkinson. December 2018.

PEER 2018/07 *An Empirical Model for Fourier Amplitude Spectra using the NGA-West2 Database*. Jeff Bayless, and Norman A. Abrahamson. December 2018.

PEER 2018/06 *Estimation of Shear Demands on Rock-Socketed Drilled Shafts subjected to Lateral Loading*. Pedro Arduino, Long Chen, and Christopher R. McGann. December 2018.

PEER 2018/05 *Selection of Random Vibration Procedures for the NGA-East Project*. Albert Kottke, Norman A. Abrahamson, David M. Boore, Yousef Bozorgnia, Christine Goulet, Justin Hollenback, Tadahiro Kishida, Armen Der Kiureghian, Olga-Joan Ktenidou, Nicolas Kuehn, Ellen M. Rathje, Walter J. Silva, Eric Thompson, and Xiaoyue Wang. December 2018.

PEER 2018/04 *Capturing Directivity Effects in the Mean and Aleatory Variability of the NGA-West 2 Ground Motion Prediction Equations.* Jennie A. Watson-Lamprey. November 2018.

PEER 2018/03 *Probabilistic Seismic Hazard Analysis Code Verification*. Christie Hale, Norman Abrahamson, and Yousef Bozorgnia. July 2018.

PEER 2018/02 *Update of the BCHydro Subduction Ground-Motion Model using the NGA-Subduction Dataset.* Norman Abrahamson, Nicolas Kuehn, Zeynep Gulerce, Nicholas Gregor, Yousef Bozorgnia, Grace Parker, Jonathan Stewart, Brian Chiou, I. M. Idriss, Kenneth Campbell, and Robert Youngs. June 2018.

PEER 2018/01 *PEER Annual Report 2017–2018*. Khalid Mosalam, Amarnath Kasalanati, and Selim Günay. June 2018.

PEER 2017/12 *Experimental Investigation of the Behavior of Vintage and Retrofit Concentrically Braced Steel Frames under Cyclic Loading*. Barbara G. Simpson, Stephen A. Mahin, and Jiun-Wei Lai, December 2017.

PEER 2017/11 *Preliminary Studies on the Dynamic Response of a Seismically Isolated Prototype Gen-IV Sodium-Cooled Fast Reactor* (*PGSFR*). Benshun Shao, Andreas H. Schellenberg, Matthew J. Schoettler, and Stephen A. Mahin. December 2017.

PEER 2017/10 *Development of Time Histories for IEEE693 Testing and Analysis (including Seismically Isolated Equipment).* Shakhzod M. Takhirov, Eric Fujisaki, Leon Kempner, Michael Riley, and Brian Low. December 2017.

PEER 2017/09 *""R" Package for Computation of Earthquake Ground-Motion Response Spectra.* Pengfei Wang, Jonathan P. Stewart, Yousef Bozorgnia, David M. Boore, and Tadahiro Kishida. December 2017.

PEER 2017/08 *Influence of Kinematic SSI on Foundation Input Motions for Bridges on Deep Foundations*. Benjamin J. Turner, Scott J. Brandenberg, and Jonathan P. Stewart. November 2017.

PEER 2017/07 *A Nonlinear Kinetic Model for Multi-Stage Friction Pendulum Systems*. Paul L. Drazin and Sanjay Govindjee. September 2017.

PEER 2017/06 *Guidelines for Performance-Based Seismic Design of Tall Buildings, Version 2.02*. TBI Working Group led by co-chairs Ron Hamburger and Jack Moehle: Jack Baker, Jonathan Bray, C.B. Crouse, Greg Deierlein, John Hooper, Marshall Lew, Joe Maffei, Stephen Mahin, James Malley, Farzad Naeim, Jonathan Stewart, and John Wallace. May 2017.

PEER 2017/05 *Recommendations for Ergodic Nonlinear Site Amplification in Central and Eastern North America*. Youssef M.A. Hashash, Joseph A. Harmon, Okan Ilhan, Grace A. Parker, and Jonathan P. Stewart. March 2017.

PEER 2017/04 *Expert Panel Recommendations for Ergodic Site Amplification in Central and Eastern North America*. Jonathan P. Stewart, Grace A. Parker, Joseph P. Harmon, Gail M. Atkinson, David M. Boore, Robert B. Darragh, Walter J. Silva, and Youssef M.A. Hashash. March 2017.

PEER 2017/03 *NGA-East Ground-Motion Models for the U.S. Geological Survey National Seismic Hazard Maps*. Christine A. Goulet, Yousef Bozorgnia, Nicolas Kuehn, Linda Al Atik, Robert R. Youngs, Robert W. Graves, and Gail M. Atkinson. March 2017.

PEER 2017/02 *U.S.–New Zealand–Japan Workshop: Liquefaction-Induced Ground Movements Effects, University of California, Berkeley, California, 2–4 November 2016.* Jonathan D. Bray, Ross W. Boulanger, Misko Cubrinovski, Kohji Tokimatsu, Steven L. Kramer, Thomas O'Rourke, Ellen Rathje, Russell A. Green, Peter K. Robinson, and Christine Z. Beyzaei. March 2017.

PEER 2017/01 *2016 PEER Annual Report*. Khalid M. Mosalam, Amarnath Kasalanati, and Grace Kang. March 2017.

PEER 2016/10 *Performance-Based Robust Nonlinear Seismic Analysis with Application to Reinforced Concrete Bridge Systems*. Xiao Ling and Khalid M. Mosalam. December 2016.

PEER 2017/09 *Detailing Requirements for Column Plastic Hinges subjected to Combined Flexural, Axial, and Torsional Seismic Loading.* Gabriel Hurtado and Jack P. Moehle. December 2016.

PEER 2016/08 *Resilience of Critical Structures, Infrastructure, and Communities*. Gian Paolo Cimellaro, Ali Zamani-Noori, Omar Kamouh, Vesna Terzic, and Stephen A. Mahin. December 2016.

PEER 2016/07 *Hybrid Simulation Theory for a Classical Nonlinear Dynamical System*. Paul L. Drazin and Sanjay Govindjee. September 2016.

PEER 2016/06 *California Earthquake Early Warning System Benefit Study*. Laurie A. Johnson, Sharyl Rabinovici, Grace S. Kang, and Stephen A. Mahin. July 2006.

PEER 2016/05 *Ground-Motion Prediction Equations for Arias Intensity Consistent with the NGA-West2 Ground-Motion Models.* Charlotte Abrahamson, Hao-Jun Michael Shi, and Brian Yang. July 2016.

PEER 2016/04 *The M_W 6.0 South Napa Earthquake of August 24, 2014: A Wake-Up Call for Renewed Investment in Seismic Resilience Across California.* Prepared for the California Seismic Safety Commission, Laurie A. Johnson and Stephen A. Mahin. May 2016.

PEER 2016/03 *Simulation Confidence in Tsunami-Driven Overland Flow.* Patrick Lynett. May 2016.

PEER 2016/02 *Semi-Automated Procedure for Windowing time Series and Computing Fourier Amplitude Spectra for the NGA-West2 Database*. Tadahiro Kishida, Olga-Joan Ktenidou, Robert B. Darragh, and Walter J. Silva. May 2016.

PEER 2016/01 *A Methodology for the Estimation of Kappa (κ) from Large Datasets: Example Application to Rock Sites in the NGA-East Database and Implications on Design Motions.* Olga-Joan Ktenidou, Norman A. Abrahamson, Robert B. Darragh, and Walter J. Silva. April 2016.

PEER 2015/13 *Self-Centering Precast Concrete Dual-Steel-Shell Columns for Accelerated Bridge Construction: Seismic Performance, Analysis, and Design*. Gabriele Guerrini, José I. Restrepo, Athanassios Vervelidis, and Milena Massari. December 2015.

PEER 2015/12 *Shear-Flexure Interaction Modeling for Reinforced Concrete Structural Walls and Columns under Reversed Cyclic Loading*. Kristijan Kolozvari, Kutay Orakcal, and John Wallace. December 2015.

PEER 2015/11 *Selection and Scaling of Ground Motions for Nonlinear Response History Analysis of Buildings in Performance-Based Earthquake Engineering*. N. Simon Kwong and Anil K. Chopra. December 2015.

PEER 2015/10 *Structural Behavior of Column-Bent Cap Beam-Box Girder Systems in Reinforced Concrete Bridges Subjected to Gravity and Seismic Loads. Part II: Hybrid Simulation and Post-Test Analysis*. Mohamed A. Moustafa and Khalid M. Mosalam. November 2015.

PEER 2015/09 *Structural Behavior of Column-Bent Cap Beam-Box Girder Systems in Reinforced Concrete Bridges Subjected to Gravity and Seismic Loads. Part I: Pre-Test Analysis and Quasi-Static Experiments*. Mohamed A. Moustafa and Khalid M. Mosalam. September 2015.

PEER 2015/08 *NGA-East: Adjustments to Median Ground-Motion Models for Center and Eastern North America*. August 2015.

PEER 2015/07 *NGA-East: Ground-Motion Standard-Deviation Models for Central and Eastern North America*. Linda Al Atik. June 2015.

PEER 2015/06 *Adjusting Ground-Motion Intensity Measures to a Reference Site for which V_{S30} = 3000 m/sec*. David M. Boore. May 2015.

PEER 2015/05 *Hybrid Simulation of Seismic Isolation Systems Applied to an APR-1400 Nuclear Power Plant.* Andreas H. Schellenberg, Alireza Sarebanha, Matthew J. Schoettler, Gilberto Mosqueda, Gianmario Benzoni, and Stephen A. Mahin. April 2015.

PEER 2015/04 *NGA-East: Median Ground-Motion Models for the Central and Eastern North America Region*. April 2015.

PEER 2015/03 *Single Series Solution for the Rectangular Fiber-Reinforced Elastomeric Isolator Compression Modulus*. James M. Kelly and Niel C. Van Engelen. March 2015.

PEER 2015/02 *A Full-Scale, Single-Column Bridge Bent Tested by Shake-Table Excitation.* Matthew J. Schoettler, José I. Restrepo, Gabriele Guerrini, David E. Duck, and Francesco Carrea. March 2015.

PEER 2015/01 *Concrete Column Blind Prediction Contest 2010: Outcomes and Observations.* Vesna Terzic, Matthew J. Schoettler, José I. Restrepo, and Stephen A Mahin. March 2015.

PEER 2014/20 *Stochastic Modeling and Simulation of Near-Fault Ground Motions for Performance-Based Earthquake Engineering*. Mayssa Dabaghi and Armen Der Kiureghian. December 2014.

PEER 2014/19 *Seismic Response of a Hybrid Fiber-Reinforced Concrete Bridge Column Detailed for Accelerated Bridge Construction*. Wilson Nguyen, William Trono, Marios Panagiotou, and Claudia P. Ostertag. December 2014.

PEER 2014/18 *Three-Dimensional Beam-Truss Model for Reinforced Concrete Walls and Slabs Subjected to Cyclic Static or Dynamic Loading*. Yuan Lu, Marios Panagiotou, and Ioannis Koutromanos. December 2014.

PEER 2014/17 *PEER NGA-East Database*. Christine A. Goulet, Tadahiro Kishida, Timothy D. Ancheta, Chris H. Cramer, Robert B. Darragh, Walter J. Silva, Youssef M.A. Hashash, Joseph Harmon, Jonathan P. Stewart, Katie E. Wooddell, and Robert R. Youngs. October 2014.

PEER 2014/16 *Guidelines for Performing Hazard-Consistent One-Dimensional Ground Response Analysis for Ground Motion Prediction*. Jonathan P. Stewart, Kioumars Afshari, and Youssef M.A. Hashash. October 2014.

PEER 2014/15 *NGA-East Regionalization Report: Comparison of Four Crustal Regions within Central and Eastern North America using Waveform Modeling and 5%-Damped Pseudo-Spectral Acceleration Response*. Jennifer Dreiling, Marius P. Isken, Walter D. Mooney, Martin C. Chapman, and Richard W. Godbee. October 2014.

PEER 2014/14 *Scaling Relations between Seismic Moment and Rupture Area of Earthquakes in Stable Continental Regions*. Paul Somerville. August 2014.

PEER 2014/13 *PEER Preliminary Notes and Observations on the August 24, 2014, South Napa Earthquake*. Grace S. Kang and Stephen A. Mahin, Editors. September 2014.

PEER 2014/12 *Reference-Rock Site Conditions for Central and Eastern North America: Part II – Attenuation (Kappa) Definition*. Kenneth W. Campbell, Youssef M.A. Hashash, Byungmin Kim, Albert R. Kottke, Ellen M. Rathje, Walter J. Silva, and Jonathan P. Stewart. August 2014.

PEER 2014/11 *Reference-Rock Site Conditions for Central and Eastern North America: Part I - Velocity Definition*. Youssef M.A. Hashash, Albert R. Kottke, Jonathan P. Stewart, Kenneth W. Campbell, Byungmin Kim, Ellen M. Rathje, Walter J. Silva, Sissy Nikolaou, and Cheryl Moss. August 2014.

PEER 2014/10 *Evaluation of Collapse and Non-Collapse of Parallel Bridges Affected by Liquefaction and Lateral Spreading*. Benjamin Turner, Scott J. Brandenberg, and Jonathan P. Stewart. August 2014.

PEER 2014/09 *PEER Arizona Strong-Motion Database and GMPEs Evaluation*. Tadahiro Kishida, Robert E. Kayen, Olga-Joan Ktenidou, Walter J. Silva, Robert B. Darragh, and Jennie Watson-Lamprey. June 2014.

PEER 2014/08 *Unbonded Pretensioned Bridge Columns with Rocking Detail*. Jeffrey A. Schaefer, Bryan Kennedy, Marc O. Eberhard, and John F. Stanton. June 2014.

PEER 2014/07 *Northridge 20 Symposium Summary Report: Impacts, Outcomes, and Next Steps*. May 2014.

PEER 2014/06 *Report of the Tenth Planning Meeting of NEES/E-Defense Collaborative Research on Earthquake Engineering*. December 2013.

PEER 2014/05 *Seismic Velocity Site Characterization of Thirty-One Chilean Seismometer Stations by Spectral Analysis of Surface Wave Dispersion.* Robert Kayen, Brad D. Carkin, Skye Corbet, Camilo Pinilla, Allan Ng, Edward Gorbis, and Christine Truong. April 2014.

PEER 2014/04 Effect of Vertical Acceleration on Shear Strength of Reinforced Concrete Columns. Hyerin Lee and Khalid M. Mosalam. April 2014.

PEER 2014/03 *Retest of Thirty-Year-Old Neoprene Isolation Bearings.* James M. Kelly and Niel C. Van Engelen. March 2014.

PEER 2014/02 *Theoretical Development of Hybrid Simulation Applied to Plate Structures.* Ahmed A. Bakhaty, Khalid M. Mosalam, and Sanjay Govindjee. January 2014.

PEER 2014/01 *Performance-Based Seismic Assessment of Skewed Bridges.* Peyman Kaviani, Farzin Zareian, and Ertugrul Taciroglu. January 2014.

PEER 2013/26 *Urban Earthquake Engineerin*g. Proceedings of the U.S.-Iran Seismic Workshop. December 2013.

PEER 2013/25 *Earthquake Engineering for Resilient Communities: 2013 PEER Internship Program Research Report Collection.* Heidi Tremayne (Editor), Stephen A. Mahin (Editor), Jorge Archbold Monterossa, Matt Brosman, Shelly Dean, Katherine deLaveaga, Curtis Fong, Donovan Holder, Rakeeb Khan, Elizabeth Jachens, David Lam, Daniela Martinez Lopez, Mara Minner, Geffen Oren, Julia Pavicic, Melissa Quinonez, Lorena Rodriguez, Sean Salazar, Kelli Slaven, Vivian Steyert, Jenny Taing, and Salvador Tena. December 2013.

PEER 2013/24 *NGA-West2 Ground Motion Prediction Equations for Vertical Ground Motions*. September 2013.

PEER 2013/23 *Coordinated Planning and Preparedness for Fire Following Major Earthquakes*. Charles Scawthorn. November 2013.

PEER 2013/22 *GEM-PEER Task 3 Project: Selection of a Global Set of Ground Motion Prediction Equations.* Jonathan P. Stewart, John Douglas, Mohammad B. Javanbarg, Carola Di Alessandro, Yousef Bozorgnia, Norman A. Abrahamson, David M. Boore, Kenneth W. Campbell, Elise Delavaud, Mustafa Erdik, and Peter J. Stafford. December 2013.

PEER 2013/21 *Seismic Design and Performance of Bridges with Columns on Rocking Foundations*. Grigorios Antonellis and Marios Panagiotou. September 2013.

PEER 2013/20 *Experimental and Analytical Studies on the Seismic Behavior of Conventional and Hybrid Braced Frames*. Jiun-Wei Lai and Stephen A. Mahin. September 2013.

PEER 2013/19 *Toward Resilient Communities: A Performance-Based Engineering Framework for Design and Evaluation of the Built Environment.* Michael William Mieler, Bozidar Stojadinovic, Robert J. Budnitz, Stephen A. Mahin, and Mary C. Comerio. September 2013.

PEER 2013/18 *Identification of Site Parameters that Improve Predictions of Site Amplification*. Ellen M. Rathje and Sara Navidi. July 2013.

PEER 2013/17 *Response Spectrum Analysis of Concrete Gravity Dams Including Dam-Water-Foundation Interaction*. Arnkjell Løkke and Anil K. Chopra. July 2013.

PEER 2013/16 *Effect of Hoop Reinforcement Spacing on the Cyclic Response of Large Reinforced Concrete Special Moment Frame Beams.* Marios Panagiotou, Tea Visnjic, Grigorios Antonellis, Panagiotis Galanis, and Jack P. Moehle. June 2013.

PEER 2013/15 *A Probabilistic Framework to Include the Effects of Near-Fault Directivity in Seismic Hazard Assessment*. Shrey Kumar Shahi, Jack W. Baker. October 2013.

PEER 2013/14 *Hanging-Wall Scaling using Finite-Fault Simulations.* Jennifer L. Donahue and Norman A. Abrahamson. September 2013.

PEER 2013/13 *Semi-Empirical Nonlinear Site Amplification and its Application in NEHRP Site Factors*. Jonathan P. Stewart and Emel Seyhan. November 2013.

PEER 2013/12 *Nonlinear Horizontal Site Response for the NGA-West2 Project.* Ronnie Kamai, Norman A. Abramson, Walter J. Silva. May 2013.

PEER 2013/11 *Epistemic Uncertainty for NGA-West2 Models.* Linda Al Atik and Robert R. Youngs. May 2013.

PEER 2013/10 *NGA-West 2 Models for Ground-Motion Directionality*. Shrey K. Shahi and Jack W. Baker. May 2013.

PEER 2013/09 *Final Report of the NGA-West2 Directivity Working Group.* Paul Spudich, Jeffrey R. Bayless, Jack W. Baker, Brian S.J. Chiou, Badie Rowshandel, Shrey Shahi, and Paul Somerville. May 2013.

PEER 2013/08 *NGA-West2 Model for Estimating Average Horizontal Values of Pseudo-Absolute Spectral Accelerations Generated by Crustal Earthquakes*. I. M. Idriss. May 2013.

PEER 2013/07 *Update of the Chiou and Youngs NGA Ground Motion Model for Average Horizontal Component of Peak Ground Motion and Response Spectra*. Brian Chiou and Robert Youngs. May 2013.

PEER 2013/06 *NGA-West2 Campbell-Bozorgnia Ground Motion Model for the Horizontal Components of PGA, PGV, and 5%-Damped Elastic Pseudo-Acceleration Response Spectra for Periods Ranging from 0.01 to 10 sec.* Kenneth W. Campbell and Yousef Bozorgnia. May 2013.

PEER 2013/05 *NGA-West 2 Equations for Predicting Response Spectral Accelerations for Shallow Crustal Earthquakes*. David M. Boore, Jonathan P. Stewart, Emel Seyhan, and Gail M. Atkinson. May 2013.

PEER 2013/04 *Update of the AS08 Ground-Motion Prediction Equations Based on the NGA-West2 Data Set*. Norman Abrahamson, Walter Silva, and Ronnie Kamai. May 2013.

PEER 2013/03 *PEER NGA-West2 Database.* Timothy D. Ancheta, Robert B. Darragh, Jonathan P. Stewart, Emel Seyhan, Walter J. Silva, Brian S.J. Chiou, Katie E. Wooddell, Robert W. Graves, Albert R. Kottke, David M. Boore, Tadahiro Kishida, and Jennifer L. Donahue. May 2013.

PEER 2013/02 *Hybrid Simulation of the Seismic Response of Squat Reinforced Concrete Shear Walls*. Catherine A. Whyte and Bozidar Stojadinovic. May 2013.

PEER 2013/01 *Housing Recovery in Chile: A Qualitative Mid-program Review*. Mary C. Comerio. February 2013.

PEER 2012/08 *Guidelines for Estimation of Shear Wave Velocity.* Bernard R. Wair, Jason T. DeJong, and Thomas Shantz. December 2012.

PEER 2012/07 *Earthquake Engineering for Resilient Communities: 2012 PEER Internship Program Research Report Collection.* Heidi Tremayne (Editor), Stephen A. Mahin (Editor), Collin Anderson, Dustin Cook, Michael Erceg, Carlos Esparza, Jose Jimenez, Dorian Krausz, Andrew Lo, Stephanie Lopez, Nicole McCurdy, Paul Shipman, Alexander Strum, Eduardo Vega. December 2012.

PEER 2012/06 *Fragilities for Precarious Rocks at Yucca Mountain.* Matthew D. Purvance, Rasool Anooshehpoor, and James N. Brune. December 2012.

PEER 2012/05 *Development of Simplified Analysis Procedure for Piles in Laterally Spreading Layered Soils.* Christopher R. McGann, Pedro Arduino, and Peter Mackenzie–Helnwein. December 2012.

PEER 2012/04 *Unbonded Pre-Tensioned Columns for Bridges in Seismic Regions.* Phillip M. Davis, Todd M. Janes, Marc O. Eberhard, and John F. Stanton. December 2012.

PEER 2012/03 *Experimental and Analytical Studies on Reinforced Concrete Buildings with Seismically Vulnerable Beam-Column Joints*. Sangjoon Park and Khalid M. Mosalam. October 2012.

PEER 2012/02 *Seismic Performance of Reinforced Concrete Bridges Allowed to Uplift during Multi-Directional Excitation*. Andres Oscar Espinoza and Stephen A. Mahin. July 2012.

PEER 2012/01 *Spectral Damping Scaling Factors for Shallow Crustal Earthquakes in Active Tectonic Regions.* Sanaz Rezaeian, Yousef Bozorgnia, I. M. Idriss, Kenneth Campbell, Norman Abrahamson, and Walter Silva. July 2012.

PEER 2011/10 *Earthquake Engineering for Resilient Communities: 2011 PEER Internship Program Research Report Collection*. Heidi Faison and Stephen A. Mahin, Editors. December 2011.

PEER 2011/09 *Calibration of Semi-Stochastic Procedure for Simulating High-Frequency Ground Motions*. Jonathan P. Stewart, Emel Seyhan, and Robert W. Graves. December 2011.

PEER 2011/08 *Water Supply in regard to Fire Following Earthquake*. Charles Scawthorn. November 2011.

PEER 2011/07 *Seismic Risk Management in Urban Areas*. Proceedings of a U.S.-Iran-Turkey Seismic Workshop. September 2011.

PEER 2011/06 *The Use of Base Isolation Systems to Achieve Complex Seismic Performance Objectives*. Troy A. Morgan and Stephen A. Mahin. July 2011.

PEER 2011/05 *Case Studies of the Seismic Performance of Tall Buildings Designed by Alternative Means.* Task 12 Report for the Tall Buildings Initiative. Jack Moehle, Yousef Bozorgnia, Nirmal Jayaram, Pierson Jones, Mohsen Rahnama, Nilesh Shome, Zeynep Tuna, John Wallace, Tony Yang, and Farzin Zareian. July 2011.

PEER 2011/04 *Recommended Design Practice for Pile Foundations in Laterally Spreading Ground.* Scott A. Ashford, Ross W. Boulanger, and Scott J. Brandenberg. June 2011.

PEER 2011/03 *New Ground Motion Selection Procedures and Selected Motions for the PEER Transportation Research Program*. Jack W. Baker, Ting Lin, Shrey K. Shahi, and Nirmal Jayaram. March 2011.

PEER 2011/02 *A Bayesian Network Methodology for Infrastructure Seismic Risk Assessment and Decision Support*. Michelle T. Bensi, Armen Der Kiureghian, and Daniel Straub. March 2011.

PEER 2011/01 *Demand Fragility Surfaces for Bridges in Liquefied and Laterally Spreading Ground.* Scott J. Brandenberg, Jian Zhang, Pirooz Kashighandi, Yili Huo, and Minxing Zhao. March 2011.

PEER 2010/05 *Guidelines for Performance-Based Seismic Design of Tall Buildings.* Developed by the Tall Buildings Initiative. November 2010.

PEER 2010/04 *Application Guide for the Design of Flexible and Rigid Bus Connections between Substation Equipment Subjected to Earthquakes.* Jean-Bernard Dastous and Armen Der Kiureghian. September 2010.

PEER 2010/03 *Shear Wave Velocity as a Statistical Function of Standard Penetration Test Resistance and Vertical Effective Stress at Caltrans Bridge Sites.* Scott J. Brandenberg, Naresh Bellana, and Thomas Shantz. June 2010.

PEER 2010/02 *Stochastic Modeling and Simulation of Ground Motions for Performance-Based Earthquake Engineering.* Sanaz Rezaeian and Armen Der Kiureghian. June 2010.

PEER 2010/01 *Structural Response and Cost Characterization of Bridge Construction Using Seismic Performance Enhancement Strategies.* Ady Aviram, Božidar Stojadinović, Gustavo J. Parra-Montesinos, and Kevin R. Mackie. March 2010.

PEER 2009/03 *The Integration of Experimental and Simulation Data in the Study of Reinforced Concrete Bridge Systems Including Soil-Foundation-Structure Interaction.* Matthew Dryden and Gregory L. Fenves. November 2009.

PEER 2009/02 *Improving Earthquake Mitigation through Innovations and Applications in Seismic Science, Engineering, Communication, and Response*. Proceedings of a U.S.-Iran Seismic Workshop. October 2009.

PEER 2009/01 *Evaluation of Ground Motion Selection and Modification Methods: Predicting Median Interstory Drift Response of Buildings*. Curt B. Haselton, Editor. June 2009.

PEER 2008/10 *Technical Manual for Strata.* Albert R. Kottke and Ellen M. Rathje. February 2009.

PEER 2008/09 *NGA Model for Average Horizontal Component of Peak Ground Motion and Response Spectra.* Brian S.-J. Chiou and Robert R. Youngs. November 2008.

PEER 2008/08 *Toward Earthquake-Resistant Design of Concentrically Braced Steel Structures*. Patxi Uriz and Stephen A. Mahin. November 2008.

PEER 2008/07 *Using OpenSees for Performance-Based Evaluation of Bridges on Liquefiable Soils*. Stephen L. Kramer, Pedro Arduino, and HyungSuk Shin. November 2008.

PEER 2008/06 *Shaking Table Tests and Numerical Investigation of Self-Centering Reinforced Concrete Bridge Columns*. Hyung IL Jeong, Junichi Sakai, and Stephen A. Mahin. September 2008.

PEER 2008/05 *Performance-Based Earthquake Engineering Design Evaluation Procedure for Bridge Foundations Undergoing Liquefaction-Induced Lateral Ground Displacement*. Christian A. Ledezma and Jonathan D. Bray. August 2008.

PEER 2008/04 *Benchmarking of Nonlinear Geotechnical Ground Response Analysis Procedures.* Jonathan P. Stewart, Annie On-Lei Kwok, Youssef M. A. Hashash, Neven Matasovic, Robert Pyke, Zhiliang Wang, and Zhaohui Yang. August 2008.

PEER 2008/03 *Guidelines for Nonlinear Analysis of Bridge Structures in California*. Ady Aviram, Kevin R. Mackie, and Božidar Stojadinović. August 2008.

PEER 2008/02 *Treatment of Uncertainties in Seismic-Risk Analysis of Transportation Systems*. Evangelos Stergiou and Anne S. Kiremidjian. July 2008.

PEER 2008/01 *Seismic Performance Objectives for Tall Buildings.* William T. Holmes, Charles Kircher, William Petak, and Nabih Youssef. August 2008.

PEER 2007/12 *An Assessment to Benchmark the Seismic Performance of a Code-Conforming Reinforced Concrete Moment-Frame Building.* Curt Haselton, Christine A. Goulet, Judith Mitrani-Reiser, James L. Beck, Gregory G. Deierlein, Keith A. Porter, Jonathan P. Stewart, and Ertugrul Taciroglu. August 2008.

PEER 2007/11 *Bar Buckling in Reinforced Concrete Bridge Columns.* Wayne A. Brown, Dawn E. Lehman, and John F. Stanton. February 2008.

PEER 2007/10 *Computational Modeling of Progressive Collapse in Reinforced Concrete Frame Structures*. Mohamed M. Talaat and Khalid M. Mosalam. May 2008.

PEER 2007/09 *Integrated Probabilistic Performance-Based Evaluation of Benchmark Reinforced Concrete Bridges*. Kevin R. Mackie, John-Michael Wong, and Božidar Stojadinović. January 2008.

PEER 2007/08 *Assessing Seismic Collapse Safety of Modern Reinforced Concrete Moment-Frame Buildings*. Curt B. Haselton and Gregory G. Deierlein. February 2008.

PEER 2007/07 *Performance Modeling Strategies for Modern Reinforced Concrete Bridge Columns*. Michael P. Berry and Marc O. Eberhard. April 2008.

PEER 2007/06 *Development of Improved Procedures for Seismic Design of Buried and Partially Buried Structures*. Linda Al Atik and Nicholas Sitar. June 2007.

PEER 2007/05 *Uncertainty and Correlation in Seismic Risk Assessment of Transportation Systems*. Renee G. Lee and Anne S. Kiremidjian. July 2007.

PEER 2007/04 *Numerical Models for Analysis and Performance-Based Design of Shallow Foundations Subjected to Seismic Loading.* Sivapalan Gajan, Tara C. Hutchinson, Bruce L. Kutter, Prishati Raychowdhury, José A. Ugalde, and Jonathan P. Stewart. May 2008.

PEER 2007/03 *Beam-Column Element Model Calibrated for Predicting Flexural Response Leading to Global Collapse of RC Frame Buildings*. Curt B. Haselton, Abbie B. Liel, Sarah Taylor Lange, and Gregory G. Deierlein. May 2008.

PEER 2007/02 *Campbell-Bozorgnia NGA Ground Motion Relations for the Geometric Mean Horizontal Component of Peak and Spectral Ground Motion Parameters*. Kenneth W. Campbell and Yousef Bozorgnia. May 2007.

PEER 2007/01 *Boore-Atkinson NGA Ground Motion Relations for the Geometric Mean Horizontal Component of Peak and Spectral Ground Motion Parameters*. David M. Boore and Gail M. Atkinson. May 2007.

PEER 2006/12 Societal Implications of Performance-Based Earthquake Engineering. Peter J. May. May 2007.

PEER 2006/11 *Probabilistic Seismic Demand Analysis Using Advanced Ground Motion Intensity Measures, Attenuation Relationships, and Near-Fault Effects*. Polsak Tothong and C. Allin Cornell. March 2007.

PEER 2006/10 *Application of the PEER PBEE Methodology to the I-880 Viaduct*. Sashi Kunnath. February 2007.

PEER 2006/09 *Quantifying Economic Losses from Travel Forgone Following a Large Metropolitan Earthquake*. James Moore, Sungbin Cho, Yue Yue Fan, and Stuart Werner. November 2006.

PEER 2006/08 *Vector-Valued Ground Motion Intensity Measures for Probabilistic Seismic Demand Analysis*. Jack W. Baker and C. Allin Cornell. October 2006.

PEER 2006/07 *Analytical Modeling of Reinforced Concrete Walls for Predicting Flexural and Coupled–Shear-Flexural Responses*. Kutay Orakcal, Leonardo M. Massone, and John W. Wallace. October 2006.

PEER 2006/06 *Nonlinear Analysis of a Soil-Drilled Pier System under Static and Dynamic Axial Loading*. Gang Wang and Nicholas Sitar. November 2006.

PEER 2006/05 *Advanced Seismic Assessment Guidelines.* Paolo Bazzurro, C. Allin Cornell, Charles Menun, Maziar Motahari, and Nicolas Luco. September 2006.

PEER 2006/04 *Probabilistic Seismic Evaluation of Reinforced Concrete Structural Components and Systems*. Tae Hyung Lee and Khalid M. Mosalam. August 2006.

PEER 2006/03 *Performance of Lifelines Subjected to Lateral Spreading.* Scott A. Ashford and Teerawut Juirnarongrit. July 2006.

PEER 2006/02 *Pacific Earthquake Engineering Research Center Highway Demonstration Project.* Anne Kiremidjian, James Moore, Yue Yue Fan, Nesrin Basoz, Ozgur Yazali, and Meredith Williams. April 2006.

PEER 2006/01 *Bracing Berkeley. A Guide to Seismic Safety on the UC Berkeley Campus.* Mary C. Comerio, Stephen Tobriner, and Ariane Fehrenkamp. January 2006.

PEER 2005/17 *Earthquake Simulation Tests on Reducing Residual Displacements of Reinforced Concrete Bridges.* Junichi Sakai, Stephen A Mahin, and Andres Espinoza. December 2005.

PEER 2005/16 *Seismic Response and Reliability of Electrical Substation Equipment and Systems.* Junho Song, Armen Der Kiureghian, and Jerome L. Sackman. April 2006.

PEER 2005/15 *CPT-Based Probabilistic Assessment of Seismic Soil Liquefaction Initiation.* R. E. S. Moss, R. B. Seed, R. E. Kayen, J. P. Stewart, and A. Der Kiureghian. April 2006.

PEER 2005/14 *Workshop on Modeling of Nonlinear Cyclic Load-Deformation Behavior of Shallow Foundations.* Bruce L. Kutter, Geoffrey Martin, Tara Hutchinson, Chad Harden, Sivapalan Gajan, and Justin Phalen. March 2006.

PEER 2005/13 *Stochastic Characterization and Decision Bases under Time-Dependent Aftershock Risk in Performance-Based Earthquake Engineering*. Gee Liek Yeo and C. Allin Cornell. July 2005.

PEER 2005/12 *PEER Testbed Study on a Laboratory Building: Exercising Seismic Performance Assessment.* Mary C. Comerio, Editor. November 2005.

PEER 2005/11 *Van Nuys Hotel Building Testbed Report: Exercising Seismic Performance Assessment.* Helmut Krawinkler, Editor. October 2005.

PEER 2005/10 *First NEES/E-Defense Workshop on Collapse Simulation of Reinforced Concrete Building Structures.* September 2005.

PEER 2005/09 *Test Applications of Advanced Seismic Assessment Guidelines.* Joe Maffei, Karl Telleen, Danya Mohr, William Holmes, and Yuki Nakayama. August 2006.

PEER 2005/08 *Damage Accumulation in Lightly Confined Reinforced Concrete Bridge Columns.* R. Tyler Ranf, Jared M. Nelson, Zach Price, Marc O. Eberhard, and John F. Stanton. April 2006.

PEER 2005/07 *Experimental and Analytical Studies on the Seismic Response of Freestanding and Anchored Laboratory Equipment.* Dimitrios Konstantinidis and Nicos Makris. January 2005.

PEER 2005/06 *Global Collapse of Frame Structures under Seismic Excitations*. Luis F. Ibarra and Helmut Krawinkler. September 2005.

PEER 2005//05 *Performance Characterization of Bench- and Shelf-Mounted Equipment.* Samit Ray Chaudhuri and Tara C. Hutchinson. May 2006.

PEER 2005/04 *Numerical Modeling of the Nonlinear Cyclic Response of Shallow Foundations.* Chad Harden, Tara Hutchinson, Geoffrey R. Martin, and Bruce L. Kutter. August 2005.

PEER 2005/03 *A Taxonomy of Building Components for Performance-Based Earthquake Engineering.* Keith A. Porter. September 2005.

PEER 2005/02 *Fragility Basis for California Highway Overpass Bridge Seismic Decision Making*. Kevin R. Mackie and Božidar Stojadinović. June 2005.

PEER 2005/01 *Empirical Characterization of Site Conditions on Strong Ground Motion.* Jonathan P. Stewart, Yoojoong Choi, and Robert W. Graves. June 2005.

PEER 2004/09 *Electrical Substation Equipment Interaction: Experimental Rigid Conductor Studies.* Christopher Stearns and André Filiatrault. February 2005.

PEER 2004/08 *Seismic Qualification and Fragility Testing of Line Break 550-kV Disconnect Switches.* Shakhzod M. Takhirov, Gregory L. Fenves, and Eric Fujisaki. January 2005.

PEER 2004/07 *Ground Motions for Earthquake Simulator Qualification of Electrical Substation Equipment.* Shakhzod M. Takhirov, Gregory L. Fenves, Eric Fujisaki, and Don Clyde. January 2005.

PEER 2004/06 *Performance-Based Regulation and Regulatory Regimes.* Peter J. May and Chris Koski. September 2004.

PEER 2004/05 *Performance-Based Seismic Design Concepts and Implementation: Proceedings of an International Workshop.* Peter Fajfar and Helmut Krawinkler, Editors. September 2004.

PEER 2004/04 *Seismic Performance of an Instrumented Tilt-up Wall Building.* James C. Anderson and Vitelmo V. Bertero. July 2004.

PEER 2004/03 *Evaluation and Application of Concrete Tilt-up Assessment Methodologies.* Timothy Graf and James O. Malley. October 2004.

PEER 2004/02 *Analytical Investigations of New Methods for Reducing Residual Displacements of Reinforced Concrete Bridge Columns.* Junichi Sakai and Stephen A. Mahin. August 2004.

PEER 2004/01 *Seismic Performance of Masonry Buildings and Design Implications.* Kerri Anne Taeko Tokoro, James C. Anderson, and Vitelmo V. Bertero. February 2004.

PEER 2003/18 *Performance Models for Flexural Damage in Reinforced Concrete Columns.* Michael Berry and Marc Eberhard. August 2003.

PEER 2003/17 *Predicting Earthquake Damage in Older Reinforced Concrete Beam-Column Joints.* Catherine Pagni and Laura Lowes. October 2004.

PEER 2003/16 *Seismic Demands for Performance-Based Design of Bridges.* Kevin Mackie and Božidar Stojadinović. August 2003.

PEER 2003/15 *Seismic Demands for Nondeteriorating Frame Structures and Their Dependence on Ground Motions.* Ricardo Antonio Medina and Helmut Krawinkler. May 2004.

PEER 2003/14 *Finite Element Reliability and Sensitivity Methods for Performance-Based Earthquake Engineering*. Terje Haukaas and Armen Der Kiureghian. April 2004.

PEER 2003/13 *Effects of Connection Hysteretic Degradation on the Seismic Behavior of Steel Moment-Resisting Frames.* Janise E. Rodgers and Stephen A. Mahin. March 2004.

PEER 2003/12 *Implementation Manual for the Seismic Protection of Laboratory Contents: Format and Case Studies*. William T. Holmes and Mary C. Comerio. October 2003.

PEER 2003/11 *Fifth U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. February 2004.

PEER 2003/10 *A Beam-Column Joint Model for Simulating the Earthquake Response of Reinforced Concrete Frames.* Laura N. Lowes, Nilanjan Mitra, and Arash Altoontash. February 2004.

PEER 2003/09 *Sequencing Repairs after an Earthquake: An Economic Approach.* Marco Casari and Simon J. Wilkie. April 2004.

PEER 2003/08 *A Technical Framework for Probability-Based Demand and Capacity Factor Design (DCFD) Seismic Formats*. Fatemeh Jalayer and C. Allin Cornell. November 2003.

PEER 2003/07 *Uncertainty Specification and Propagation for Loss Estimation Using FOSM Methods.* Jack W. Baker and C. Allin Cornell. September 2003.

PEER 2003/06 *Performance of Circular Reinforced Concrete Bridge Columns under Bidirectional Earthquake Loading*. Mahmoud M. Hachem, Stephen A. Mahin, and Jack P. Moehle. February 2003.

PEER 2003/05 *Response Assessment for Building-Specific Loss Estimation.* Eduardo Miranda and Shahram Taghavi. September 2003.

PEER 2003/04 *Experimental Assessment of Columns with Short Lap Splices Subjected to Cyclic Loads*. Murat Melek, John W. Wallace, and Joel Conte. April 2003.

PEER 2003/03 *Probabilistic Response Assessment for Building-Specific Loss Estimation*. Eduardo Miranda and Hesameddin Aslani. September 2003.

PEER 2003/02 *Software Framework for Collaborative Development of Nonlinear Dynamic Analysis Program*. Jun Peng and Kincho H. Law. September 2003.

PEER 2003/01 *Shake Table Tests and Analytical Studies on the Gravity Load Collapse of Reinforced Concrete Frames.* Kenneth John Elwood and Jack P. Moehle. November 2003.

PEER 2002/24 *Performance of Beam to Column Bridge Joints Subjected to a Large Velocity Pulse.* Natalie Gibson, André Filiatrault, and Scott A. Ashford. April 2002.

PEER 2002/23 *Effects of Large Velocity Pulses on Reinforced Concrete Bridge Columns.* Greg L. Orozco and Scott A. Ashford. April 2002.

PEER 2002/22 *Characterization of Large Velocity Pulses for Laboratory Testing.* Kenneth E. Cox and Scott A. Ashford. April 2002.

PEER 2002/21 *Fourth U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. December 2002.

PEER 2002/20 *Barriers to Adoption and Implementation of PBEE Innovations.* Peter J. May. August 2002.

PEER 2002/19 *Economic-Engineered Integrated Models for Earthquakes: Socioeconomic Impacts.* Peter Gordon, James E. Moore II, and Harry W. Richardson. July 2002.

PEER 2002/18 *Assessment of Reinforced Concrete Building Exterior Joints with Substandard Details.* Chris P. Pantelides, Jon Hansen, Justin Nadauld, and Lawrence D. Reaveley. May 2002.

PEER 2002/17 *Structural Characterization and Seismic Response Analysis of a Highway Overcrossing Equipped with Elastomeric Bearings and Fluid Dampers: A Case Study*. Nicos Makris and Jian Zhang. November 2002.

PEER 2002/16 *Estimation of Uncertainty in Geotechnical Properties for Performance-Based Earthquake Engineering*. Allen L. Jones, Steven L. Kramer, and Pedro Arduino. December 2002.

PEER 2002/15 *Seismic Behavior of Bridge Columns Subjected to Various Loading Patterns*. Asadollah Esmaeily-Gh. and Yan Xiao. December 2002.

PEER 2002/14 *Inelastic Seismic Response of Extended Pile Shaft Supported Bridge Structures.* T.C. Hutchinson, R.W. Boulanger, Y.H. Chai, and I.M. Idriss. December 2002.

PEER 2002/13 *Probabilistic Models and Fragility Estimates for Bridge Components and Systems.* Paolo Gardoni, Armen Der Kiureghian, and Khalid M. Mosalam. June 2002.

PEER 2002/12 *Effects of Fault Dip and Slip Rake on Near-Source Ground Motions: Why Chi-Chi Was a Relatively Mild M7.6 Earthquake*. Brad T. Aagaard, John F. Hall, and Thomas H. Heaton. December 2002.

PEER 2002/11 *Analytical and Experimental Study of Fiber-Reinforced Strip Isolators*. James M. Kelly and Shakhzod M. Takhirov. September 2002.

PEER 2002/10 *Centrifuge Modeling of Settlement and Lateral Spreading with Comparisons to Numerical Analyses***.** Sivapalan Gajan and Bruce L. Kutter. January 2003.

PEER 2002/09 *Documentation and Analysis of Field Case Histories of Seismic Compression during the 1994 Northridge, California, Earthquake*. Jonathan P. Stewart, Patrick M. Smith, Daniel H. Whang, and Jonathan D. Bray. October 2002.

PEER 2002/08 *Component Testing, Stability Analysis and Characterization of Buckling-Restrained Unbonded Braces™*. Cameron Black, Nicos Makris, and Ian Aiken. September 2002.

PEER 2002/07 *Seismic Performance of Pile-Wharf Connections.* Charles W. Roeder, Robert Graff, Jennifer Soderstrom, and Jun Han Yoo. December 2001.

PEER 2002/06 *The Use of Benefit-Cost Analysis for Evaluation of Performance-Based Earthquake Engineering Decisions*. Richard O. Zerbe and Anthony Falit-Baiamonte. September 2001.

PEER 2002/05 *Guidelines, Specifications, and Seismic Performance Characterization of Nonstructural Building Components and Equipment*. André Filiatrault, Constantin Christopoulos, and Christopher Stearns. September 2001.

PEER 2002/04 *Consortium of Organizations for Strong-Motion Observation Systems and the Pacific Earthquake Engineering Research Center Lifelines Program: Invited Workshop on Archiving and Web Dissemination of Geotechnical Data, 4–5 October 2001*. September 2002.

PEER 2002/03 *Investigation of Sensitivity of Building Loss Estimates to Major Uncertain Variables for the Van Nuys Testbed*. Keith A. Porter, James L. Beck, and Rustem V. Shaikhutdinov. August 2002.

PEER 2002/02 *The Third U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. July 2002.

PEER 2002/01 *Nonstructural Loss Estimation: The UC Berkeley Case Study.* Mary C. Comerio and John C. Stallmeyer. December 2001.

PEER 2001/16 *Statistics of SDF-System Estimate of Roof Displacement for Pushover Analysis of Buildings.* Anil K. Chopra, Rakesh K. Goel, and Chatpan Chintanapakdee. December 2001.

PEER 2001/15 *Damage to Bridges during the 2001 Nisqually Earthquake.* R. Tyler Ranf, Marc O. Eberhard, and Michael P. Berry. November 2001.

PEER 2001/14 *Rocking Response of Equipment Anchored to a Base Foundation.* Nicos Makris and Cameron J. Black. September 2001.

PEER 2001/13 *Modeling Soil Liquefaction Hazards for Performance-Based Earthquake Engineering*. Steven L. Kramer and Ahmed-W. Elgamal. February 2001.

PEER 2001/12 *Development of Geotechnical Capabilities in OpenSees*. Boris Jeremić. September 2001.

PEER 2001/11 *Analytical and Experimental Study of Fiber-Reinforced Elastomeric Isolators*. James M. Kelly and Shakhzod M. Takhirov. September 2001.

PEER 2001/10 *Amplification Factors for Spectral Acceleration in Active Regions.* Jonathan P. Stewart, Andrew H. Liu, Yoojoong Choi, and Mehmet B. Baturay. December 2001.

PEER 2001/09 *Ground Motion Evaluation Procedures for Performance-Based Design.* Jonathan P. Stewart, Shyh-Jeng Chiou, Jonathan D. Bray, Robert W. Graves, Paul G. Somerville, and Norman A. Abrahamson. September 2001.

PEER 2001/08 *Experimental and Computational Evaluation of Reinforced Concrete Bridge Beam-Column Connections for Seismic Performance*. Clay J. Naito, Jack P. Moehle, and Khalid M. Mosalam. November 2001.

PEER 2001/07 *The Rocking Spectrum and the Shortcomings of Design Guidelines.* Nicos Makris and Dimitrios Konstantinidis. August 2001.

PEER 2001/06 *Development of an Electrical Substation Equipment Performance Database for Evaluation of Equipment Fragilities*. Thalia Agnanos. April 1999.

PEER 2001/05 *Stiffness Analysis of Fiber-Reinforced Elastomeric Isolators.* Hsiang-Chuan Tsai and James M. Kelly. May 2001.

PEER 2001/04 *Organizational and Societal Considerations for Performance-Based Earthquake Engineering*. Peter J. May. April 2001.

PEER 2001/03 *A Modal Pushover Analysis Procedure to Estimate Seismic Demands for Buildings: Theory and Preliminary Evaluation*. Anil K. Chopra and Rakesh K. Goel. January 2001.

PEER 2001/02 *Seismic Response Analysis of Highway Overcrossings Including Soil-Structure Interaction*. Jian Zhang and Nicos Makris. March 2001.

PEER 2001/01 *Experimental Study of Large Seismic Steel Beam-to-Column Connections.* Egor P. Popov and Shakhzod M. Takhirov. November 2000.

PEER 2000/10 *The Second U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. March 2000.

PEER 2000/09 *Structural Engineering Reconnaissance of the August 17, 1999 Earthquake: Kocaeli (Izmit), Turkey*. Halil Sezen, Kenneth J. Elwood, Andrew S. Whittaker, Khalid Mosalam, John J. Wallace, and John F. Stanton. December 2000.

PEER 2000/08 *Behavior of Reinforced Concrete Bridge Columns Having Varying Aspect Ratios and Varying Lengths of Confinement*. Anthony J. Calderone, Dawn E. Lehman, and Jack P. Moehle. January 2001.

PEER 2000/07 *Cover-Plate and Flange-Plate Reinforced Steel Moment-Resisting Connections.* Taejin Kim, Andrew S. Whittaker, Amir S. Gilani, Vitelmo V. Bertero, and Shakhzod M. Takhirov. September 2000.

PEER 2000/06 *Seismic Evaluation and Analysis of 230-kV Disconnect Switches.* Amir S. J. Gilani, Andrew S. Whittaker, Gregory L. Fenves, Chun-Hao Chen, Henry Ho, and Eric Fujisaki. July 2000.

PEER 2000/05 *Performance-Based Evaluation of Exterior Reinforced Concrete Building Joints for Seismic Excitation*. Chandra Clyde, Chris P. Pantelides, and Lawrence D. Reaveley. July 2000.

PEER 2000/04 *An Evaluation of Seismic Energy Demand: An Attenuation Approach*. Chung-Che Chou and Chia-Ming Uang. July 1999.

PEER 2000/03 *Framing Earthquake Retrofitting Decisions: The Case of Hillside Homes in Los Angeles.* Detlof von Winterfeldt, Nels Roselund, and Alicia Kitsuse. March 2000.

PEER 2000/02 *U.S.-Japan Workshop on the Effects of Near-Field Earthquake Shaking*. Andrew Whittaker, Editor. July 2000.

PEER 2000/01 *Further Studies on Seismic Interaction in Interconnected Electrical Substation Equipment.* Armen Der Kiureghian, Kee-Jeung Hong, and Jerome L. Sackman. November 1999.

PEER 1999/14 *Seismic Evaluation and Retrofit of 230-kV Porcelain Transformer Bushings*. Amir S. Gilani, Andrew S. Whittaker, Gregory L. Fenves, and Eric Fujisaki. December 1999.

PEER 1999/13 *Building Vulnerability Studies: Modeling and Evaluation of Tilt-up and Steel Reinforced Concrete Buildings*. John W. Wallace, Jonathan P. Stewart, and Andrew S. Whittaker, Editors. December 1999.

PEER 1999/12 *Rehabilitation of Nonductile RC Frame Building Using Encasement Plates and Energy-Dissipating Devices*. Mehrdad Sasani, Vitelmo V. Bertero, James C. Anderson. December 1999.

PEER 1999/11 *Performance Evaluation Database for Concrete Bridge Components and Systems under Simulated Seismic Loads*. Yael D. Hose and Frieder Seible. November 1999.

PEER 1999/10 *U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. December 1999.

PEER 1999/09 *Performance Improvement of Long Period Building Structures Subjected to Severe Pulse-Type Ground Motions*. James C. Anderson, Vitelmo V. Bertero, and Raul Bertero. October 1999.

PEER 1999/08 *Envelopes for Seismic Response Vectors.* Charles Menun and Armen Der Kiureghian. July 1999.

PEER 1999/07 *Documentation of Strengths and Weaknesses of Current Computer Analysis Methods for Seismic Performance of Reinforced Concrete Members*. William F. Cofer. November 1999.

PEER 1999/06 *Rocking Response and Overturning of Anchored Equipment under Seismic Excitations*. Nicos Makris and Jian Zhang. November 1999.

PEER 1999/05 *Seismic Evaluation of 550 kV Porcelain Transformer Bushings.* Amir S. Gilani, Andrew S. Whittaker, Gregory L. Fenves, and Eric Fujisaki. October 1999.

PEER 1999/04 *Adoption and Enforcement of Earthquake Risk-Reduction Measures.* Peter J. May, Raymond J. Burby, T. Jens Feeley, and Robert Wood. August 1999.

PEER 1999/03 *Task 3 Characterization of Site Response General Site Categories*. Adrian Rodriguez-Marek, Jonathan D. Bray and Norman Abrahamson. February 1999.

PEER 1999/02 *Capacity-Demand-Diagram Methods for Estimating Seismic Deformation of Inelastic Structures: SDF Systems*. Anil K. Chopra and Rakesh Goel. April 1999.

PEER 1999/01 *Interaction in Interconnected Electrical Substation Equipment Subjected to Earthquake Ground Motions*. Armen Der Kiureghian, Jerome L. Sackman, and Kee-Jeung Hong. February 1999.

PEER 1998/08 *Behavior and Failure Analysis of a Multiple-Frame Highway Bridge in the 1994 Northridge Earthquake*. Gregory L. Fenves and Michael Ellery. December 1998.

PEER 1998/07 *Empirical Evaluation of Inertial Soil-Structure Interaction Effects.* Jonathan P. Stewart, Raymond B. Seed, and Gregory L. Fenves. November 1998.

PEER 1998/06 *Effect of Damping Mechanisms on the Response of Seismic Isolated Structures*. Nicos Makris and Shih-Po Chang. November 1998.

PEER 1998/05 *Rocking Response and Overturning of Equipment under Horizontal Pulse-Type Motions*. Nicos Makris and Yiannis Roussos. October 1998.

PEER 1998/04 *Pacific Earthquake Engineering Research Invitational Workshop Proceedings, May 14–15, 1998: Defining the Links between Planning, Policy Analysis, Economics and Earthquake Engineering*. Mary Comerio and Peter Gordon. September 1998.

PEER 1998/03 *Repair/Upgrade Procedures for Welded Beam to Column Connections*. James C. Anderson and Xiaojing Duan. May 1998.

PEER 1998/02 *Seismic Evaluation of 196 kV Porcelain Transformer Bushings.* Amir S. Gilani, Juan W. Chavez, Gregory L. Fenves, and Andrew S. Whittaker. May 1998.

PEER 1998/01 *Seismic Performance of Well-Confined Concrete Bridge Columns.* Dawn E. Lehman and Jack P. Moehle. December 2000.

PEER REPORTS: ONE HUNDRED SERIES

PEER 2012/103 *Performance-Based Seismic Demand Assessment of Concentrically Braced Steel Frame Buildings.* Chui-Hsin Chen and Stephen A. Mahin. December 2012.

PEER 2012/102 *Procedure to Restart an Interrupted Hybrid Simulation: Addendum to PEER Report 2010/103.* Vesna Terzic and Bozidar Stojadinovic. October 2012.

PEER 2012/101 *Mechanics of Fiber Reinforced Bearings.* James M. Kelly and Andrea Calabrese. February 2012.

PEER 2011/107 *Nonlinear Site Response and Seismic Compression at Vertical Array Strongly Shaken by 2007 Niigata-ken Chuetsu-oki Earthquake.* Eric Yee, Jonathan P. Stewart, and Kohji Tokimatsu. December 2011.

PEER 2011/106 *Self Compacting Hybrid Fiber Reinforced Concrete Composites for Bridge Columns.* Pardeep Kumar, Gabriel Jen, William Trono, Marios Panagiotou, and Claudia Ostertag. September 2011.

PEER 2011/105 *Stochastic Dynamic Analysis of Bridges Subjected to Spacially Varying Ground Motions.* Katerina Konakli and Armen Der Kiureghian. August 2011.

PEER 2011/104 *Design and Instrumentation of the 2010 E-Defense Four-Story Reinforced Concrete and Post-Tensioned Concrete Buildings.* Takuya Nagae, Kenichi Tahara, Taizo Matsumori, Hitoshi Shiohara, Toshimi Kabeyasawa, Susumu Kono, Minehiro Nishiyama (Japanese Research Team) and John Wallace, Wassim Ghannoum, Jack Moehle, Richard Sause, Wesley Keller, Zeynep Tuna (U.S. Research Team). June 2011.

PEER 2011/103 *In-Situ Monitoring of the Force Output of Fluid Dampers: Experimental Investigation.* Dimitrios Konstantinidis, James M. Kelly, and Nicos Makris. April 2011.

PEER 2011/102 *Ground-Motion Prediction Equations 1964–2010.* John Douglas. April 2011.

PEER 2011/101 *Report of the Eighth Planning Meeting of NEES/E-Defense Collaborative Research on Earthquake Engineering.* Convened by the Hyogo Earthquake Engineering Research Center (NIED), NEES Consortium, Inc. February 2011.

PEER 2010/111 *Modeling and Acceptance Criteria for Seismic Design and Analysis of Tall Buildings.* Task 7 Report for the Tall Buildings Initiative - Published jointly by the Applied Technology Council. October 2010.

PEER 2010/110 *Seismic Performance Assessment and Probabilistic Repair Cost Analysis of Precast Concrete Cladding Systems for Multistory Buildlings.* Jeffrey P. Hunt and Božidar Stojadinovic. November 2010.

PEER 2010/109 *Report of the Seventh Joint Planning Meeting of NEES/E-Defense Collaboration on Earthquake Engineering. Held at the E-Defense, Miki, and Shin-Kobe, Japan, September 18–19, 2009.* August 2010.

PEER 2010/108 *Probabilistic Tsunami Hazard in California.* Hong Kie Thio, Paul Somerville, and Jascha Polet, preparers. October 2010.

PEER 2010/107 *Performance and Reliability of Exposed Column Base Plate Connections for Steel Moment-Resisting Frames.* Ady Aviram, Božidar Stojadinovic, and Armen Der Kiureghian. August 2010.

PEER 2010/106 *Verification of Probabilistic Seismic Hazard Analysis Computer Programs.* Patricia Thomas, Ivan Wong, and Norman Abrahamson. May 2010.

PEER 2010/105 *Structural Engineering Reconnaissance of the April 6, 2009, Abruzzo, Italy, Earthquake, and Lessons Learned.* M. Selim Günay and Khalid M. Mosalam. April 2010.

PEER 2010/104 *Simulating the Inelastic Seismic Behavior of Steel Braced Frames, Including the Effects of Low-Cycle Fatigue.* Yuli Huang and Stephen A. Mahin. April 2010.

PEER 2010/103 *Post-Earthquake Traffic Capacity of Modern Bridges in California.* Vesna Terzic and Božidar Stojadinović. March 2010.

PEER 2010/102 *Analysis of Cumulative Absolute Velocity (CAV) and JMA Instrumental Seismic Intensity (I_{JMA}) Using the PEER–NGA Strong Motion Database.* Kenneth W. Campbell and Yousef Bozorgnia. February 2010.

PEER 2010/101 *Rocking Response of Bridges on Shallow Foundations.* Jose A. Ugalde, Bruce L. Kutter, and Boris Jeremic. April 2010.

PEER 2009/109 *Simulation and Performance-Based Earthquake Engineering Assessment of Self-Centering Post-Tensioned Concrete Bridge Systems.* Won K. Lee and Sarah L. Billington. December 2009.

PEER 2009/108 *PEER Lifelines Geotechnical Virtual Data Center.* J. Carl Stepp, Daniel J. Ponti, Loren L. Turner, Jennifer N. Swift, Sean Devlin, Yang Zhu, Jean Benoit, and John Bobbitt. September 2009.

PEER 2009/107 *Experimental and Computational Evaluation of Current and Innovative In-Span Hinge Details in Reinforced Concrete Box-Girder Bridges: Part 2: Post-Test Analysis and Design Recommendations.* Matias A. Hube and Khalid M. Mosalam. December 2009.

PEER 2009/106 *Shear Strength Models of Exterior Beam-Column Joints without Transverse Reinforcement.* Sangjoon Park and Khalid M. Mosalam. November 2009.

PEER 2009/105 *Reduced Uncertainty of Ground Motion Prediction Equations through Bayesian Variance Analysis.* Robb Eric S. Moss. November 2009.

PEER 2009/104 *Advanced Implementation of Hybrid Simulation.* Andreas H. Schellenberg, Stephen A. Mahin, Gregory L. Fenves. November 2009.

PEER 2009/103 *Performance Evaluation of Innovative Steel Braced Frames.* T. Y. Yang, Jack P. Moehle, and Božidar Stojadinovic. August 2009.

PEER 2009/102 *Reinvestigation of Liquefaction and Nonliquefaction Case Histories from the 1976 Tangshan Earthquake.* Robb Eric Moss, Robert E. Kayen, Liyuan Tong, Songyu Liu, Guojun Cai, and Jiaer Wu. August 2009.

PEER 2009/101 *Report of the First Joint Planning Meeting for the Second Phase of NEES/E-Defense Collaborative Research on Earthquake Engineering.* Stephen A. Mahin et al. July 2009.

PEER 2008/104 *Experimental and Analytical Study of the Seismic Performance of Retaining Structures.* Linda Al Atik and Nicholas Sitar. January 2009.

PEER 2008/103 *Experimental and Computational Evaluation of Current and Innovative In-Span Hinge Details in Reinforced Concrete Box-Girder Bridges. Part 1: Experimental Findings and Pre-Test Analysis.* Matias A. Hube and Khalid M. Mosalam. January 2009.

PEER 2008/102 *Modeling of Unreinforced Masonry Infill Walls Considering In-Plane and Out-of-Plane Interaction.* Stephen Kadysiewski and Khalid M. Mosalam. January 2009.

PEER 2008/101 *Seismic Performance Objectives for Tall Buildings.* William T. Holmes, Charles Kircher, William Petak, and Nabih Youssef. August 2008.

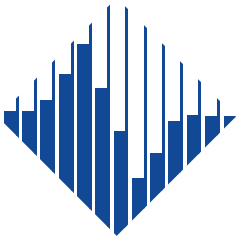
PEER 2007/101 *Generalized Hybrid Simulation Framework for Structural Systems Subjected to Seismic Loading*. Tarek Elkhoraibi and Khalid M. Mosalam. July 2007.

PEER 2007/100 *Seismic Evaluation of Reinforced Concrete Buildings Including Effects of Masonry Infill Walls*. Alidad Hashemi and Khalid M. Mosalam. July 2007.

The Pacifi c Earthquake Engineering Research Center (PEER) is a multi-institutional research and education center with headquarters at the University of California, Berkeley. Investigators from over 20 universities, several consulting companies, and researchers at various state and federal government agencies contribute to research programs focused on performance-based earthquake engineering.

These research programs aim to identify and reduce the risks from major earthquakes to life safety and to the economy by including research in a wide variety of disciplines including structural and geotechnical engineering, geology/seismology, lifelines, transportation, architecture, economics, risk management, and public policy.

PEER is supported by federal, state, local, and regional agencies, together with industry partners.



PEER Core Institutions

University of California, Berkeley (Lead Institution)
California Institute of Technology
Oregon State University
Stanford University
University of California, Davis
University of California, Irvine
University of California, Los Angeles
University of California, San Diego
University of Nevada, Reno
University of Southern California
University of Washington

PEER reports can be ordered at https://peer.berkeley.edu/peer-reports or by contacting

Pacific Earthquake Engineering Research Center
University of California, Berkeley
325 Davis Hall, Mail Code 1792
Berkeley, CA 94720-1792
Tel: 510-642-3437
Email: peer_center@berkeley.edu

ISSN 2770-8314
https://doi.org/10.55461/APRI8198