

6-14-2018

Camera-Based Distance Sensor

Kai Schmidt

Santa Clara University, kschmidt@scu.edu

Evan Holmes

Santa Clara University, eholmes@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/idp_senior



Part of the [Computer Engineering Commons](#), and the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Schmidt, Kai and Holmes, Evan, "Camera-Based Distance Sensor" (2018). *Interdisciplinary Design Senior Theses*. 37.
https://scholarcommons.scu.edu/idp_senior/37

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Interdisciplinary Design Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING

Date: June 13, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Kai Schmidt
Evan Holmes

ENTITLED


Camera-Based Distance Sensor

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

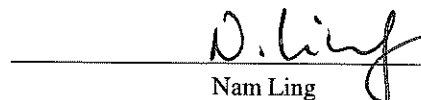
BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING



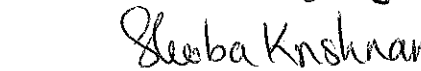
Angela Musurlian



Sally Wood



Nam Ling



Shoba Krishnan

Camera-Based Distance Sensor

by

Kai Schmidt
Evan Holmes

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Electrical Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 14, 2018

Camera-Based Distance Sensor

Kai Schmidt
Evan Holmes

Department of Computer Engineering
Department of Electrical Engineering
Santa Clara University
June 14, 2018

ABSTRACT

While working on a robotics project at the electrical contracting company for which we work, we discovered a gap in the electronic distance sensor market in terms of range, accuracy, precision, and cost. We designed and constructed a prototype for an electronic distance sensing component which utilizes a camera, laser, and image processor to measure distances. The laser is pointed at a surface and an image of the laser dot is captured. An image processing algorithm determines the pixel position of the dot in the image, and this position is compared to a lookup table of known values to determine the distance to the dot.

In measuring our prototype's performance, we found that it was capable of measuring distances up to 5 meters with greater than 90% accuracy. We also discuss some possible ways to improve the viability of the technology, including ways to improve the refresh rate as well as the reliability.

Table of Contents

List of Figures	vi
1 Introduction	1
1.1 Motivation	1
1.2 Background	1
1.3 Applications	2
2 Requirements and Constraints	3
3 Solution and Feasibility	5
3.1 Proposed Solution	5
3.2 Feasibility	6
3.3 Point to Distance Conversion	6
3.4 Finding the Best Constants	7
3.5 Limits	8
3.6 Accuracy	8
4 Software	11
4.1 Design	11
4.2 Algorithm	11
4.3 Challenges and Iterations	15
5 Construction	17
5.1 Introduction	17
5.2 Case Design	18
5.2.1 Modularity	18
5.2.2 Camera	18
5.2.3 Laser	19
5.2.4 Raspberry Pi and PCB	20
5.3 Case Construction	20
5.3.1 3D Printing	21
5.3.2 Tolerance	22
5.4 Printed Circuit Board	23
6 The Final Product	25
6.1 Specifications	25
6.1.1 Met Requirements	25
6.1.2 Unmet requirements	25
6.2 Accuracy	26
7 Budget	27

8	Ethics	28
8.1	Ethical Justification for the Project	28
8.2	Ethics, our Project, and What it Means to be a Good Engineer	28
8.3	Ethical Pitfalls in the Project Itself	29
9	Sustainability	31
9.1	Environmental Impact	31
9.2	Social Sustainability	32
9.3	Economic Sustainability	32
10	Conclusions and Future Work	34
	Acknowledgements	35
	Appendix	36
	Bibliography	37

List of Figures

3.1	Diagram of Point to Distance Conversion	6
3.2	Error and Accuracy Calculations	8
3.3	Error Table	10
4.1	High-level Software Loop Diagram	12
4.2	A section of a typical image of the laser dot taken by the camera.	12
4.3	The narrow strip of the image that is actually loaded into the program's memory.	12
4.4	The array of pixels after columns of pixels have had their colors averaged. Vertical dimension not to scale.	13
4.5	A representation of the buckets' positions and simple colors.	13
4.6	A representation of the buckets after they have been amalgamated.	14
5.1	Final Case	17
5.2	Design for Base	18
5.3	Camera Mount	19
5.4	Laser Mount	19
5.5	All Case Iterations	20
5.6	3D Printers	21
5.7	Tolerance Test	22
5.8	Printed Circuit Board	23
5.9	EagleCAD Files	23
5.10	Bantam Tools OtherMill	24
6.1	A scatter plot of the prototype's error at different distances.	26

Chapter 1

Introduction

1.1 Motivation

We work for an electrical prefabrication contractor in Santa Clara doing research and development. One of our projects included making a robot able to identify nuanced features along a wall. This required a distance sensor that had a decent range as well as high accuracy and precision.

We tried several electronic ranging technologies including sonar, infrared, and time-of-flight, but we found them all to be lacking in one way or another, and we ended up having to redefine the scope of the project.

1.2 Background

In today's market, there is a wide variety of distance sensors for different purposes. Different sensors vary in range, method, accuracy, precision, durability, interfacing, and price, among others. Specifically, this project is targeted towards low cost sensors, meaning under \$200. This excludes more intricate components such as 3D LiDAR used on self-driving cars which cost thousands of dollars [10].

Sensor	Range	Price	Accuracy	Precision
Sonar	5m	\$5 - \$35	90%	Conical
Infrared	2m	\$15 - \$40	80%	Narrow Cone
ToF	1.5m	\$15 - \$40	99%	Dot
2D LiDAR	10m	\$130+	95%	Dot

Table 1.1: Comparable Products [2] [1] [3] [5] [4] [8]

The largest problem that we found in our work with sensors is that many have conical precision. This means that the point to which the sensor measures gets larger and larger the further away it is. This is very helpful for finding walls, but not very useful for finding specific points, or creating 2D or 3D maps. If the user were to point a sensor with

conical precision at a cluttered group of objects, they would not know what the device is measuring.

There is clearly a gap in the \$50 to \$100 range, and above \$100 is usually out of hobbyists budgets. Those products over \$100 are usually very precise and have high range (above 5 m) and accuracy (over 95%) but the main problem is price. Perhaps these solutions will go down in price in the next few years, but for now there are not any range finding devices which have high precision and accuracy for under \$100.

1.3 Applications

There are a variety of applications for distance sensors. Distance sensors and range finders are typically used as components of larger systems, and not as independent devices. Sensors are used by industry, academics, and hobbyists alike. This section details a few applications across a variety of industries, but this is not comprehensive.

One common use is in robotics, and that is where we found the need for a new sensor. Typically robotics projects use distance sensors for obstacle detection. Many use infrared or sonar sensors because their low precision is great for avoiding walls and large obstacles[2] [1] [3] [5]. More precise measurements are needed to detect smaller objects. Currently, the only low cost option is a Time-of-Flight (ToF) sensor, which is very accurate, but only at low ranges. [4] Otherwise, high cost LiDAR sensors will work, as they do work at distances up to 10 meters or higher, depending on the specific device. However most long range distance sensor lack short range (under 1m) abilities.

There are also a few ways that distance sensors could be used in construction, especially if they have high accuracy and precision. They could be used to create laser tape measures which would decrease the amount of labor required when making measurements. This requires very high accuracy (99%), especially at high distances(3m to 5m, or more).

Security is another possible applications. Distance sensors can be used to detect motion, or if an object is present. Parking lots often use small distance sensors to tell if cars are in spaces, or if a car pulls up to a toll booth. Most applications only require low precision, but high reliability sensors. However more advanced object detection could require higher precision sensors.

Chapter 2

Requirements and Constraints

Before actually designing a device, a number of requirements that it should fulfill were identified. One is that it should work as a component of a larger system. There are plenty of standalone laser distance sensors on the market, but the niche it is targeting is in electronic distance sensors that report measurements to some other device. This means it can be used by electronics hobbyists and professionals alike. There are certain considerations when designing a device for this purpose rather than one that could be used easily by an end user.

The device should be easy to integrate into an existing project or system. The person using it in their design should have to jump through as few hoops as possible to make it work. The easiest way to do this is to make the interface familiar. Distance measurements should be output on input/output pins using various common electronic communication protocols, similar to other distance sensing components on the market. Common protocols for most electronic sensors include pulse-width modulation, asynchronous serial, and i²c.

To support a wide range of applications the device should have a range of 10 meters but should still be very accurate under 1 meter. If necessary, it is okay if some accuracy is sacrificed at higher distances if it can be gained at lower ones.

The goal of a 10 meter range is for ideal conditions. Most distance sensing technologies have certain conditions that reduce their performance. For example, sonar sensors can be interfered with by other nearby sonar sensors, resulting in less accurate measurements. Some infrared sensors can be thrown off by large amounts of ambient sunlight. Acknowledging that the technology used may have certain conditions which are unideal for its operation, it should still have a range of up to 5 meters even in these conditions.

In tests of distance sensors, one of the primary issues with certain technologies is their inaccuracy. The device should be usable for a wide range of applications, so the accuracy of its distance measurement should not be lower than 90%.

The device should also be usable in systems that have real-time constraints. This means that distance measurements

should be reported with a high frequency so that the data is always as new as possible. 10 Hz is a good baseline refresh rate for distance measurements, but 20-30 Hz would be ideal for systems that really need to react to their environment as quickly as possible.

One of the problems with better distance sensing technologies is their price. A 1-dimensional LiDAR sensor is usually well over \$100. 2D and 3D LiDAR can have costs in the \$1000's. To ensure that the device is comparatively affordable, the total cost of the parts needed for its construction should be under \$75, preferably under \$50.

There are many applications for distance sensing where the system itself is mobile. When using electronics in mobile applications, there are two primary considerations: power draw and size. The device should have as low a power draw as possible to maximize the life of any battery that is powering it in a mobile scenario. Under 250 mA is a good current limit, as it can still be powered by a Raspberry Pi or an Arduino.

While a commercially available distance sensor should have a size of only a few centimeters, the designed device will simply be a prototype, so it is acceptable for it to be up to 5 centimeters in height and depth. Even for this larger prototype size, a device of this size could still fit comfortably on many types of systems.

Chapter 3

Solution and Feasibility

3.1 Proposed Solution

We propose a device that uses a camera, laser, and image processor to measure distance. The laser projects a dot, and the processor uses the camera to take a picture of it. The processor then runs a simple image processing algorithm to determine how far away the laser's dot is based on its position in the image. It outputs this information on output pins in multiple different protocols. A Raspberry Pi Zero is used as the image processor to keep costs low while still having the computational capability required. Because of how the camera and laser are aligned and the way the geometry works out, such a device can theoretically have very high accuracy at short distances while still being able to measure longer ones (albeit with decreased precision).

3.2 Feasibility

The feasibility of this design has already been partially proven [12] [11]. In order to prove that this project is itself viable, some additional calculations were necessary to find out how exactly the image processor could turn a pixel into a distance. This section explains the theoretical mathematics behind how this device works, explores its physical limits, and calculates the theoretical accuracy. In the end, these calculations were not as applicable as previously anticipated, and a lookup table created from calibrated measurements ended up being used. However these calculations still provide insight on how tolerant the design could be.

3.3 Point to Distance Conversion

The software should find the position of the laser dot, ideally with as little pixel error as possible. Figure 3.1 to the right shows the angles and values the user needs to have defined in order to measure distances successfully. θ_{FoV} is the Field of View angle, which varies by camera and lens. θ_{cen} is the angle to the center of the camera, which was chosen to be $90^\circ - \theta_{FoV}$.

Originally, the camera was going to be mounted parallel to the laser, but that method would only take advantage of half the pixels available. It was then decided to turn the camera so that the furthest right pixel is at 90 degrees, maximizing the viewing angle of the camera, and therefore the range which it can detect. N_p is the number of pixels from the center, with a maximum value of $N/2$ and minimum value of $-N/2$, where N is the number of horizontal pixels in the image. The pixel width, θ_{pw} is the angular width of each pixel, calculated as θ_{FoV}/N .

Finally, D_{CL} is the distance between the camera focal point and the laser line. This distance is somewhat arbitrarily picked: The larger it is, the more accurate the device at long distances (over 1 m), as the number of pixels between each meter would increase. However the device must be kept to a reasonable size for practical uses, and a large D_{CL} would likely make the case start to bend. These variables can be used to create a function to find θ_p the angle from the

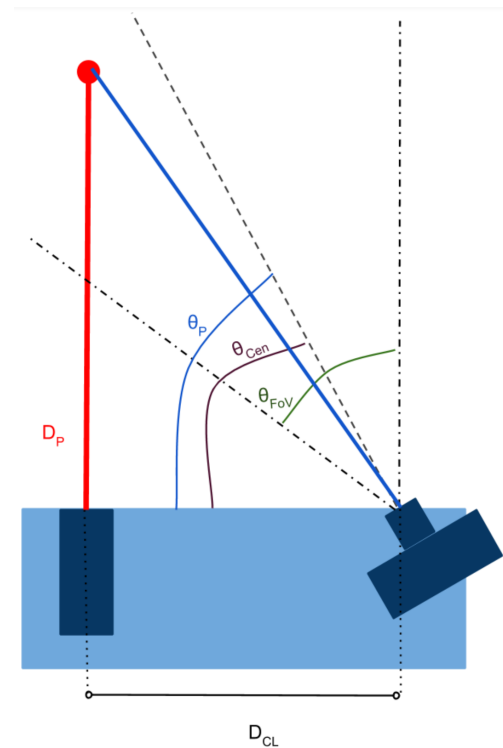


Figure 3.1: Diagram of Point to Distance Conversion

laser dot to the camera to the laser lens. It is defined as

$$\theta_p = \theta_{cen} \pm \theta_{pw}(N_p + .5)$$

To calculate the final distance D_p , use

$$D_p = D_{cl} * \tan(\theta_p)$$

3.4 Finding the Best Constants

The calculations require five different values, dependent on three constants: N , θ_{FoV} , and D_{CL} . D_{CL} is probably the easiest to figure out. D_{CL} should be low enough to keep the device on the smaller side, but it should also be high enough to leave room to create a reliable mount and have higher accuracy. A high D_{CL} would lower the values of θ_p , especially at long ranges. Because the function is tangential, lower θ_{pw} values increases the number of pixels between each meter, which is especially important at longer ranges. D_{CL} values of 9 cm to 12 cm leaves some room on each side for the mounts. 10 cm is for the sake of making calculations easy. However, when accounting for a PCB being added into the case design, the mounts have to be moved out, so the final value is 11 cm.

θ_{FoV} defines θ_{cen} as well as θ_{pw} (along with N). A larger Field of View allows very short distances to be seen, especially under half a meter, but also decreases the pixel width, lowering the accuracy slightly. The maximum usable Field of View is 90° as the left most pixel would be along the device itself. The minimum distance is

$$0.11 * \tan(90 - \theta_{FoV})$$

so a good θ_{FoV} is somewhere between 40 and 80 degrees. For sake of calculations, $\theta_{FoV} = 60^\circ$ is used.

In reality, the Field of View is determined by the camera, and the number of horizontal pixels N given by the camera is much more important to accuracy and range than θ_{FoV} , so it is generally just an accepted value of the best camera, and not as much of a conscious choice. The camera chosen has a 5 megapixel OV5647 sensor, which claims to have a θ_{FoV} of 65° , but when the camera was tested, the θ_{FoV} was actually closer to 55° , so the mounts were redesigned.

The most important variable is N , the number of horizontal pixels. This is especially important for long ranges. The requirement is for 90% accuracy at up to 10 m. To calculate error, the following equation is used:

$$\%error = \frac{1}{2} * \left(\left(\frac{n+1}{n} - 1 \right) + \left(1 - \frac{n}{n+1} \right) \right)$$

To have an error of under 10% at 10 m, the calculated value must be between 9.05 m and 11.05 m. The θ_p for 9.05 m, 10 m, and 11.05 m respectively are 89.303° , 89.370° , and 89.430° , so the error of angle must be less than 0.06° .

Therefore the maximum pixel width is 0.06° , and with θ_{FOV} of 60° , $N \geq 1000$. However that assumes that there is no mechanical error and the image processor gets the exact center pixel. If the dot finding algorithm is allowed to be off by up to two pixels, then the maximum θ_{pw} is cut in half to 0.03° . A standard HD image has an N of 1920, which would give a θ_{pw} of 0.031° .

The 5 megapixel OV5647 Sensor that is used can take pictures at a maximum resolution of 2592 x 1944 pixels. The sensor has a θ_{FOV} of 55° , so for a 2592 pixel wide image, each pixel has a width of 0.02123° (θ_{pw}), almost one third the maximum pixel width of 0.06° . This camera was chosen because it has about 2.5x the minimum required amount of horizontal pixels, and it is designed to interface with the Raspberry Pi Zero.

3.5 Limits

The main requirement of this project is that this device can detect distances up to 10 m with 90% accuracy. Because $\theta_c = 57.5^\circ$ and $\theta_F = 55^\circ$, the max camera angle is 90° and the minimum camera angle is 35° . Therefore the minimum observable distance D_{pmin} is just under 5 cm. The maximum required distance is 10 m, but because the center of the edge pixel is $.0106^\circ$ off, the max calculated distance is 225 m. However, that would be very inaccurate.

3.6 Accuracy

The requirement for this project is that the sensor maintains at least 10% accuracy up to 10 m. The table in figure 3.3 shows the pixel angle, calculated distance, and maximum error if it is within one pixel. There are three sections: max, middle and min distances. The angles are from the pixel centers, which is why they are not rational numbers. The error is calculated using the same equation as section 3.4.

$$\%error = \frac{1}{2} * \left(\left(\frac{n+1}{n} - 1 \right) + \left(1 - \frac{n}{n+1} \right) \right)$$

The maximum calculated error is calculated as if the value is off by one entire exact pixel. Assuming the actual angle is, at maximum, half a pixel away from the

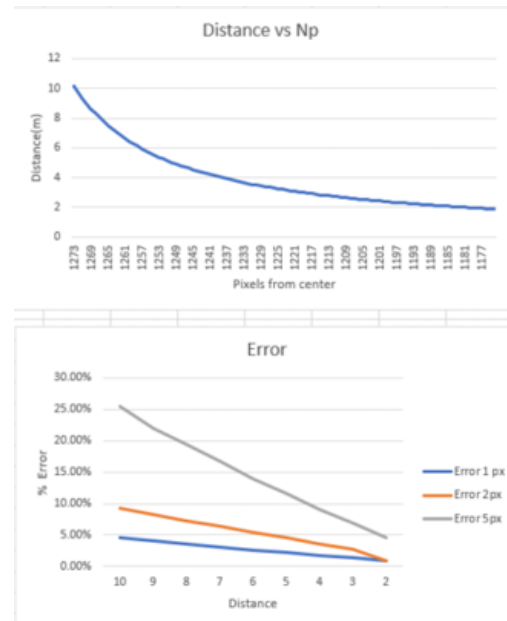


Figure 3.2: Error and Accuracy Calculations

center of the pixel, the real error is about half that of the error calculated in figure 3.3. This gives about a 95% accuracy at 10 meters. The error increases exponentially and 10 meters is about the limit of acceptable error. This also means that close range (under 1 meter) detection will be extremely accurate.

Figure 3.2 shows two graphs. The top is Distance vs N_p (or number of pixels from center). It shows that the number of pixels between each meter rises exponentially, so it is less precise and less accurate, even if the device picks up the exact correct pixel. The error graph shows the error percentage if the device is off by one, two or six pixels. Theoretically, the device is sufficiently accurate at up to 10 meters as long as the camera and dot finding algorithm are accurate to two pixels.

These theoretical calculations show that the 2592 pixel wide camera is just barely enough to be 90% accurate at 10 meters. However the bigger challenge will be making sure the casing is robust and rigid enough to not allow the camera or laser to move.

N p	Theta P	Dp	Max Error
1296	90.01254	-456.956	
1295	89.98746	456.9559	
1294	89.96238	152.3186	
1293	89.93731	91.39114	66.667%
1292	89.91223	65.27936	40.000%
1291	89.88715	50.77281	28.571%
1290	89.86208	41.54136	22.222%
1289	89.837	35.15036	18.182%
1288	89.81192	30.46362	15.385%
1287	89.78684	26.87963	13.333%
1286	89.76177	24.05017	11.765%
1285	89.73669	21.75965	10.526%
1284	89.71161	19.86748	9.524%
1283	89.68654	18.27805	8.696%
1282	89.66146	16.92409	8.000%
1281	89.63638	15.75689	7.408%
1280	89.6113	14.74029	6.897%
1279	89.58623	13.84691	6.452%
1278	89.56115	13.05563	6.061%
1277	89.53607	12.34989	5.715%
1276	89.511	11.71653	5.406%
1275	89.48592	11.14497	5.128%
1274	89.46084	10.62657	4.878%
1273	89.43576	10.15425	4.651%
1272	89.41069	9.722123	4.445%
653	73.88792	0.346184	
650	73.81269	0.344487	0.493%
649	73.78762	0.343925	0.164%
648	73.76254	0.343364	0.163%
329	65.76292	0.222126	
326	65.68769	0.221349	0.351%
325	65.66262	0.221091	0.117%
324	65.63754	0.220833	0.117%
5	57.63792	0.157806	
2	57.56269	0.157348	0.291%
1	57.53762	0.157196	0.097%
0	57.51254	0.157044	0.097%

Figure 3.3: Error Table

Chapter 4

Software

A large part of the device's functionality comes from the image processing program. This program is responsible for invoking the built-in Raspberry Pi camera software, processing the captured image, and reporting the measured distance.

4.1 Design

While the device is on, there is a software loop that is constantly running within the image processing program. Figure 4.1 shows this process, which is meant to run several times each second. The process is as follows:

1. The program invokes the Raspberry Pi camera software, `raspistill`, to take a picture which is saved on the Pi's memory card.
2. A narrow strip of the image is loaded into the program's memory.
3. The image processing algorithm is run on the image strip to determine the position of the dot.
4. The dot position is converted to an actual distance via a lookup table.
5. The distance measurement is output on the device's pins as pulse-width modulation and asynchronous serial.

The program was written in C++ because of the language's speed as well as the availability of libraries to access the Raspberry Pi's GPIO pins.

4.2 Algorithm

The image processing algorithm itself has its own steps. When designing it, compromises had to be made between accuracy, reliability, and speed. By ensuring that the laser and the camera are well aligned, it can be guaranteed that the

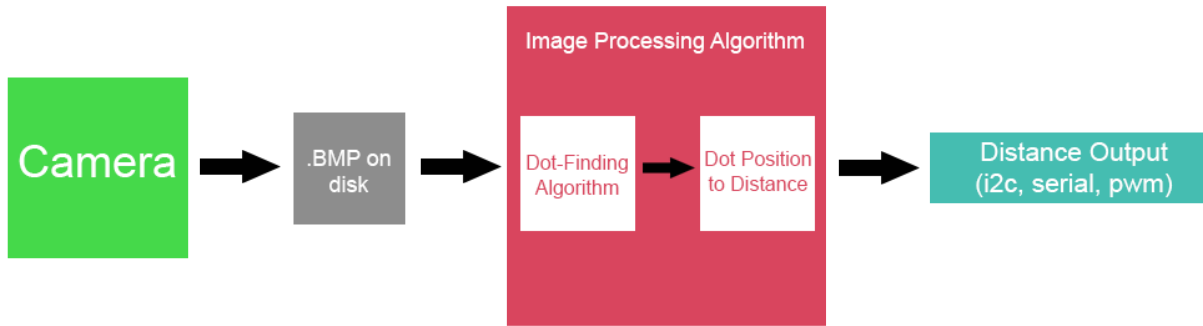


Figure 4.1: High-level Software Loop Diagram

laser’s dot always appears at the same vertical pixel coordinate in the image. This makes it possible for the algorithm to search a much smaller section for the dot.

The first step in the program’s main loop is invoking the Raspberry Pi’s camera application, raspistill. raspistill saves this image to the Pi’s micro SD card, and our program reads a small subsection of the image into memory. Figure 4.2 shows a small section of the image that the camera takes. The dot appears as very white in the center with red fringes on the edges, and the program takes advantage of this basic color configuration.



Figure 4.2: A section of a typical image of the laser dot taken by the camera.

Figure 4.3 show the portion of this image that would actually be loaded into the program’s memory. It is 5 pixels high because that was found to be a good compromise for a height small enough to fall inside dots that are further away but tall enough to still have a good sample of the dot’s interior. However, the fact that this image has any verticality at all means that processing it would require above-linear time.



Figure 4.3: The narrow strip of the image that is actually loaded into the program’s memory.

To flatten this image even more and make it able to be processed in linear time, every column of pixels is averaged so that the resulting image is effectively one pixel high. This can be seen in figure 4.4, an image that is stretched in the vertical dimension so that it is easier to see.

This simple array of colors is then converted to one of “simple colors”. Simple colors are, unsurprisingly, a simplified representation of each color in the array. The defined simple colors are white, red, green, and other. The RGB values of each pixel in the array are converted into simple colors based on some simple threshold equations.



Figure 4.4: The array of pixels after columns of pixels have had their colors averaged. Vertical dimension not to scale.

Once each pixel is assigned a simple color, it is inserted into a bucket. The bucket into which it is inserted depends on its position in the array. Every bucket is the same size. Smaller bucket sizes can have more nuance, but larger bucket sizes lead to fewer buckets, which means that can be processed in less time. Experimentation found the ideal compromise to be a bucket size of 15-20 pixels.

Each bucket is itself assign a simple color. This color is the maximum of all of its constituent pixels. With simple colors, comparisons are dictated by the simply ordering: white > red > green > other. This means that the addition of, say, a white pixel into a red bucket will turn the bucket white, etc. Figure 4.5 shows a representation of the above image after all the pixels have been assigned a simple color and inserted into a bucket.



Figure 4.5: A representation of the buckets' positions and simple colors.

The code snippet below shows the loop which averages each pixel and inserts it into a bucket.

```
// Intialize buckets
unordered_map<unsigned, Bucket> buckets;

// Get average pixels and add them to buckets
for(int i = 0; i < width; i++) {
    vector<unsigned> avg_pixel(3, 0);
    for(int j = 0; j < scan_line_height; j++) {
        auto color = bmp_pixel(image, width, i, j);
        avg_pixel[0] += color[0];
        avg_pixel[1] += color[1];
        avg_pixel[2] += color[2];
    }
    avg_pixel[0] /= scan_line_height;
    avg_pixel[1] /= scan_line_height;
    avg_pixel[2] /= scan_line_height;
    unsigned main_key = i / bucket_width;
    if(buckets.find(main_key) == buckets.end())
        buckets.insert(pair<unsigned, Bucket>(main_key, Bucket()));
    buckets[main_key].insert(main_key, i, simplifyColor(avg_pixel));
}
```

Once all pixels have been inserted into buckets, the buckets are amalgamated. The amalgamation process involves combining all buckets that can be combined. Two buckets are considered combinable if:

1. The buckets are adjacent.
2. The buckets have the same simple color.

The code snippet below shows the bucket amalgamation loop.

```
// Amalgamate buckets
bool done = false;
while(!done) {
    vector<pair<unsigned, unsigned>> keys_to_merge;

    // For each pair of different buckets, check if they are adjacent.
    // If they are, add the pair of their keys to the keys_to_merge
    for(auto &a: buckets) {
        for(auto &b: buckets) {
            if(a.first != b.first && a.second.adjacent(b.second)) {
                keys_to_merge.push_back(pair<unsigned, unsigned>(a.first, b.first));
            }
        }
    }

    // For each pair of keys to merge, merge the corresponding buckets
    for(auto &pair: keys_to_merge) {
        if(buckets.find(pair.first) != buckets.end()
            && buckets.find(pair.second) != buckets.end())
        {
            Bucket a = buckets[pair.first];
            Bucket b = buckets[pair.second];
            buckets.erase(pair.first);
            buckets.erase(pair.second);
            a.merge(b);
            unsigned new_key = a.mainKey();
            buckets[new_key] = a;
        }
    }
    // We are done amalgomating buckets when there are no more keys to marge.
    done = keys_to_merge.empty();
}
```

Figure 4.6 shows the a representation of the buckets in the example once the amalgamation process is complete.



Figure 4.6: A representation of the buckets after they have been amalgamated.

With this representation, patterns in the bucket's simple colors that suggest the position of a laser dot can be searched for. The bucket patterns searched for are listed below in order of precedence.

1. red, white, red
2. white, red / red, white
3. red
4. white
5. green

If the program fails to match any of these patterns in the buckets' configuration, then the algorithm has failed on this iteration, and a distance of zero is immediately returned.

If one of the patterns is found, then the program determines the dot position to be the pixel in the center of the most prominent bucket in the pattern. For single-color patterns, the single bucket is the most prominent. If the pattern has multiple buckets, then the most prominent bucket is the white one. In the above example, the center of the laser dot is found to be at the center of the white bucket in figure 4.6.

If the dot position has been found, then the last step is to convert this number into an actual distance. This is done via a lookup table. The lookup table was calibrated by taking many pictures with the device at different distances from a wall and determining the dot position by manual image inspection. 90 calibration points up to a range of 7.4 meters were recorded. However, in practice, there were problems with the device being able to consistently find the dot for distances greater than 3 meters. If the dot is found at a position that matches one of the calibration points in the lookup table, then that point's distance value is returned. If the position lies between two calibration points, then the distance is interpolated from the points' associated distances.

Once the program determines the distance, it writes this distance to the output pins as a pulse-width modulated signal as well as an asynchronous serial message using a library designed for the Raspberry Pi. The program then begins the loop again by taking the next picture.

4.3 Challenges and Iterations

A number of challenges were encountered when writing the software, and different attempts to overcome these challenges led to the software having a few iterations before the final design was decided. The primary attributes that needed to be balanced were accuracy, reliability, and speed.

In the first iteration of the software, the program simply searched the image strip for very white pixels and then averaged their positions. While this was very fast and simple, it was subject to often failing in certain ambient light conditions. The camera's color correction often meant that the dot was not very white. Also, other white light sources like computer screens or reflected light would throw this algorithm off. The failure of this algorithm is what made it clear that the redness of the laser dot needed to be factored in in order for the algorithm to be able to find it accurately.

The second main iteration of the program was very close to the current one. Buckets and simple colors were implemented as a way of organizing pixels in a way that could be processed quickly to find the dot positions. Every example image used until then had a laser dot that seemed to follow the red-white-red pattern, so this was the only bucket pattern for which was searched. This is when a problem with reliability was first discovered. The dot's vertical position

in the image was constantly fluctuating, making it impossible for the algorithm to find the dot unless the height of the image strip was constantly updated manually. In an attempt to remedy this, the algorithm was made to search a large range of heights. While this greatly improved reliability, it decreased the speed by many orders of magnitude.

It is likely that the reason the vertical position of the dot varied by so much was because of the fact that the resin-like plastic the case was 3D printed from was subject to a very small amount of bending. By reprinting the case in a more rigid plastic, it was possible to improve reliability while being able to check for the dot at only one vertical position.

The third iteration of the software improved both reliability and speed. By adding checks for buckets patterns other than red-white-red, edge cases that were not previously anticipated were able to be accounted for.

The first step in improving speed was to make the camera take a smaller image. The camera used has hardware-accelerated JPEG encoding, but images needed to be read into the program as BMP so that they would be easy to process. However, this meant that raspistill had to convert the JPEG image file the camera gave it to BMP. For the default 2592 x 1944 pixels images that the camera took, this took a few seconds. By running the raspistill capture command with arguments to crop the image vertically, it had to convert a much smaller image to BMP, greatly increasing capture speed.

The second step in improving speed was simple code optimizations. The code was combed through, and any unnecessary data copies and redundant data structures that were left over from previous iterations of the software were eliminated.

The last challenge that was encountered was another of reliability. For unknown reasons, the camera occasionally took drastically underexposed and green-shifted images. While root of this problem was not discovered, a workaround in the algorithm was implemented. The green simple color introduced, and the single green bucket pattern was added as the lowest priority pattern. In the event that the camera takes one of these bad images, this case accounts for it and still accurately finds the dot.

Chapter 5

Construction

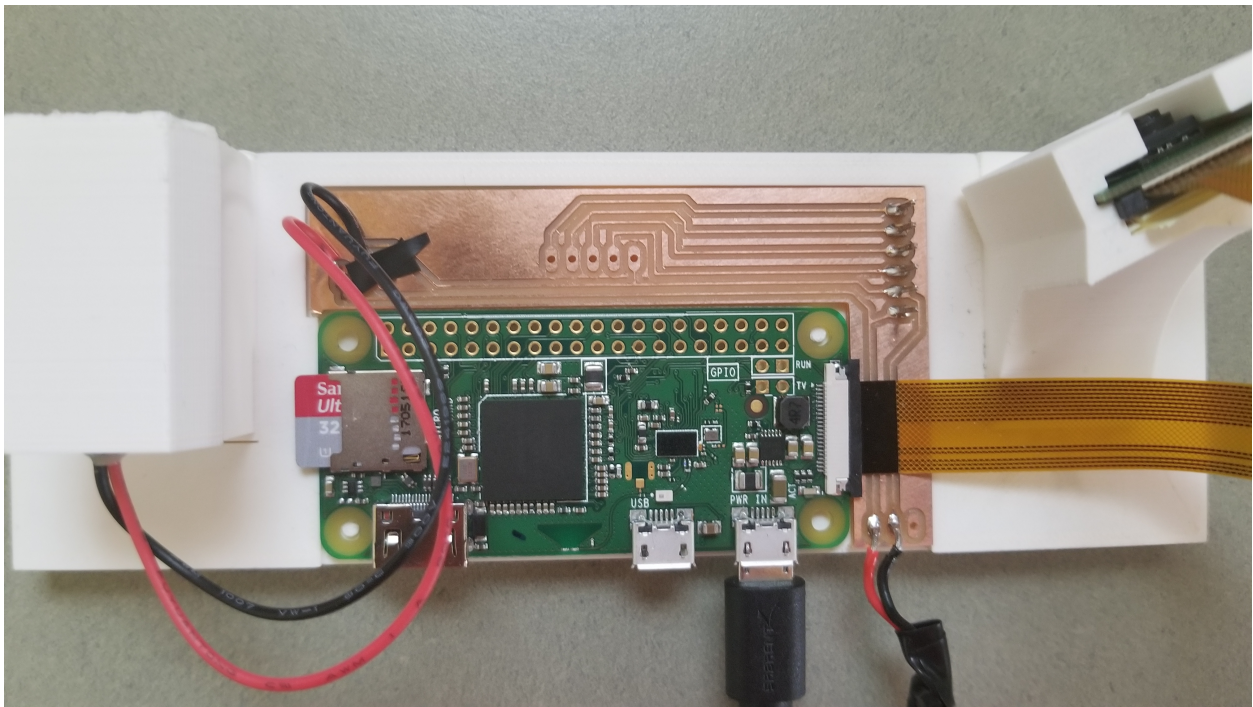


Figure 5.1: Final Case

5.1 Introduction

The casing which holds all of the different components together is one of the most important parts of the Camera-Based Distance Sensor. It is vital that the components do not fall out, do not move, and are protected. The case was designed in Autodesk Inventor and then 3D printed. The Printed Circuit Board is used to connect the user interfacing pins with the Raspberry Pi Zero and power the laser.

The mathematics behind the device rely heavily on the angle at which the camera picks up the center of the laser

dot. As shown in the feasibility section, the angle between the camera/laser line and the camera/dot line is dependent on the angle of the camera center being exactly 57.5 degrees from the camera/laser line and the laser being exactly perpendicular to the device. Because each pixel is about 2% of a degree, the tolerance for movement of any part of the system is very low. In reality, a lookup table created from careful and tedious calibrations is used, but for those calibrations to be accurate, it is still incredibly important for the pieces not to move.

This section will detail the design decisions and prototyping processes to create the case and printed circuit board.

5.2 Case Design

5.2.1 Modularity

A modular approach to design and construction was used in order to minimize prototyping time. Because the highest precision settings were used on the 3D printers, the total print time for the full case was around 20 hours. It was expected that many parts would not work the first time, would need to be updated, and may break. The case was designed with module mounts for the laser and camera so that an entire case did not have to be reprinted if one of the mounts needed to be changed for any reason.

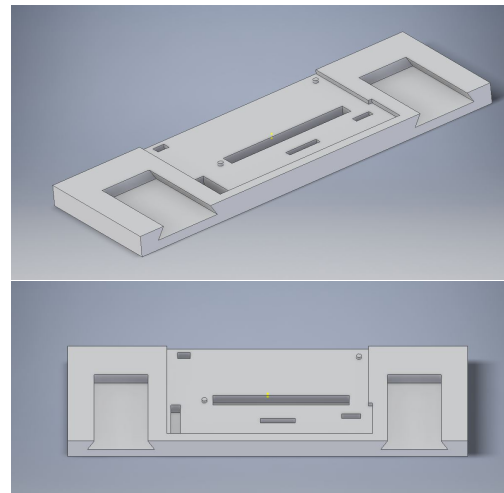


Figure 5.2: Design for Base

After doing a small amount of internet research and discussing with mechanical engineering students, a trapezoidal shape was chosen to hold the camera and laser component modules. This design was chosen because it is simple, and the 3D printer can build it without any supports. Originally there were plans to have other small pieces attach around the modules to keep them from sliding out, but once a working model was produced, it was obvious that if a module was loose enough that it may fall out, it is too loose anyway and should be reprinted.

5.2.2 Camera

The camera mount was without a doubt the most complicated piece of this case to design. The specific 57.5 degree angle at which the camera had to be aligned was not impossible, but the software did not make it intuitive, and certainly not easy to edit. The camera placement had to be offset by a specific amount so that the camera lens was in

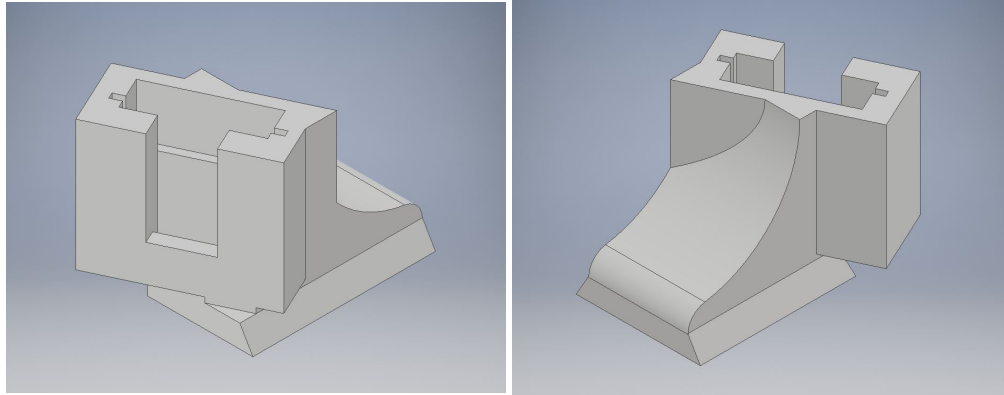


Figure 5.3: Camera Mount

the geometric center of the mount so as to keep a specified distance of 11 cm between the lens and the laser. It also had to be exactly in line with the front of the device so that the user can be sure that the measurements start at the front of the device.

To ensure the stability of the camera relative to the rest of the device, there are small slits which hold the edges of the camera circuit board. There are SMD components on the front and back of the camera so that only the edges, (where there are no SMD components) are used to secure the camera. The camera fits very tightly, so to remove it, there is a small hole in the bottom box so that a pen or tweezers can be used to push the camera out. Lastly, there is a curve on the back. While it may not seem necessary, it is used to give the box another layer of stability.

5.2.3 Laser

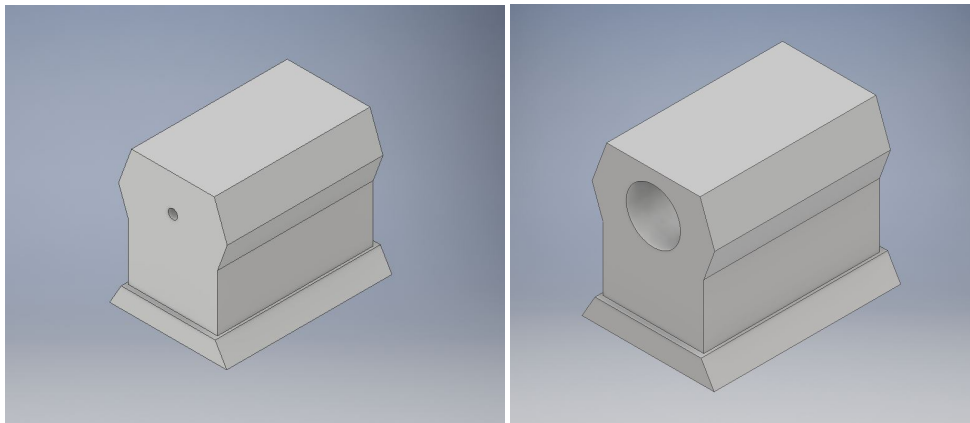


Figure 5.4: Laser Mount

The laser mount is fairly simple. The laser is perpendicular with the front of the device and is the same height as the camera lens. The first iteration had an open front and a smaller opening in the back for wires to come out of. However, after some testing, the shape of the laser was found to be more ovular and abstracted. The design was changed to

cover the front, save for a 1-2 mm circular pinhole. The back is also open for the laser to slide into. The pinhole shape worked and the laser became more circular, solving the problem.

5.2.4 Raspberry Pi and PCB

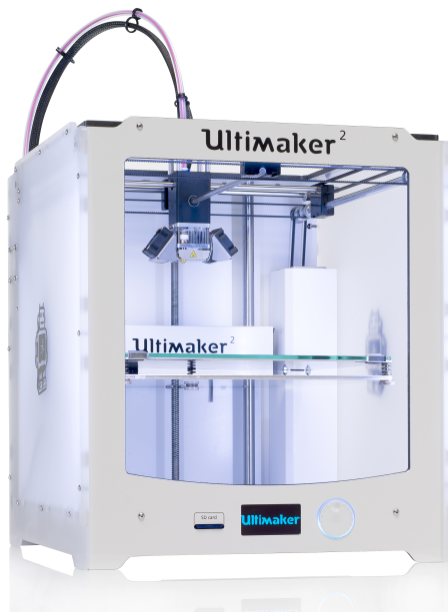
The center of the board is used to store the Raspberry Pi Zero and the printed circuit board which surrounds it. The first few iterations only had room for the Raspberry Pi Zero, and it was not very secure. The second iteration of case two small cylinders which fit into the screw holes of the Raspberry Pi, and several holes for the pins to fit into. In the new design, the direction of the Raspberry Pi is changed so that the MicroUSB and MiniHDMI ports faced the back. This creates less of a burden on the user if they wish to interact with the PI directly while using the device.

5.3 Case Construction



Figure 5.5: All Case Iterations

In total, five different version of the case were created (Figure 5.5). They were 3D printed in the Santa Clara University IMakerLab, on two different machines: the Ultimaker 3, and the Formlab Form 2.



(a) Ultimaker 3



(b) Formlab Form 2

Figure 5.6: 3D Printers

5.3.1 3D Printing

The case construction began by 3D printing the first iteration of the case design. The Ultimaker 3 Extended Rapid Prototyping Machine was chosen because of its high accuracy, easy to use software, and accessibility at the MakerLab. It can use a variety of materials but for this project, only PLA in different colors were used. The PLA, or PolyLactic Acid material is pretty standard. It is used commonly because it is low cost, lightweight, and easy to use [9]. After the first print, the Formlab Form 2 printer, which uses stereolithography (SLA), meaning the printer converts a liquid resin into solid parts layer by layer [6]. The 'Tough' resin was used on the Form 2 printer because it was the strongest and most rigid resin available at the maker lab [7]. These prints were more accurate, but complicated to complete, as each piece needed several supporting structures. Even after the print the pieces needed to be cured in UV light for several hours, then the supports had to be cut off.

The Form 2 print was slightly malleable, which helped to grip the laser and camera mounts. However it also meant that the base was starting to bend, which is not acceptable as it would throw off all of the calibrations. So for the final design, the Ultimaker was chosen because of its rigidity. The downside to the rigidity is that the mounts don't grip to the case as much, so the mounts were given less tolerance, and then the pieces were filed down if they did not fit. In the end, the PLA print works very well for this project.

5.3.2 Tolerance

Perhaps the most tedious aspect of the case design and construction was finding the correct tolerance for the camera and laser mounts. As stated many times before, stability is incredibly important in the design of the case, as a change of a fraction of an angle can throw the device off.

When the first iteration of the case was printed, space was not left for tolerance, and therefore, none of the pieces fit together. The 3D printer is not perfect, and because it uses additive manufacturing, the pieces are fractions of a millimeter larger than they are supposed to be, even on the highest settings. Figure 5.7 shows the two different tolerance tests that were used. The top images are the female parts, which are slightly smaller, and the bottom are the male parts which are slightly larger. The left images have a tolerance of 1 mm, while the right images have a tolerance of 0.2mm. These were printed on both 3D printers and were very helpful in creating good mounts.

Ideally, these tests could have been done several more times with more specific tolerances until the perfect amount was found. However 3D printers are not perfect and do not create the exact same amount each time. On multiple occasions, there were two different mounts printed with the same exact tolerance in the design, but one would fit in almost loosely while the other had to be filed down and forced in.

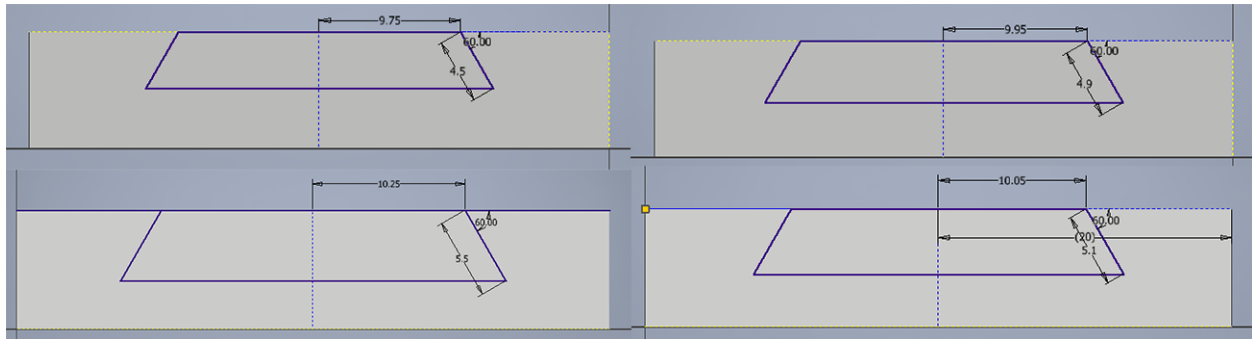


Figure 5.7: Tolerance Test

5.4 Printed Circuit Board

The purpose of the printer circuit board is to connect the input/output pins with the rest of the device. The 5V and Ground pins power the Raspberry Pi Zero and the laser. The other pins connect the I2C, Serial, and PWM pins, though in the end, only Serial and PWM were working. The board is designed to fit around the Raspberry Pi Zero, connect to it using 30 AWG flexible silicone wires.

The design began with a basic design on Autodesk EagleCAD software. The design is very simple, as there are no SMD or through hole components, just pinholes. Figure 5.9 shows the schematic layout on the left, as well as the board layout on the right. The schematic uses 3 hole pinholes when only two are needed because there were no 2 pinhole pieces available in the standard part libraries. Otherwise, it is fairly straightforward: 5V and Ground are connected to Laser and Pi Power, while all other pins, as well as ground, are connected to the GPIO.

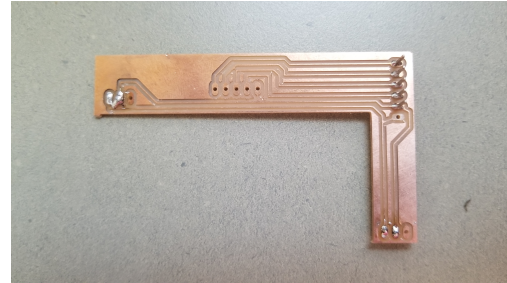


Figure 5.8: Printed Circuit Board

There are 5 pins for GPIO: one for ground, one for PWM, and two for asynchronous serial. There is one left unused, as there should be two for I2C, but one had to be removed in order to fit in the desired dimensions. In the board layout, the wires must be a certain distance apart for production. In figure 5.8, it shows that the wires go almost to the edge of the board, and there was no more room for another wire. The I2C was taken out because it caused the most trouble. In the board layout it is clear that it is designed to wrap around the Raspberry Pi.

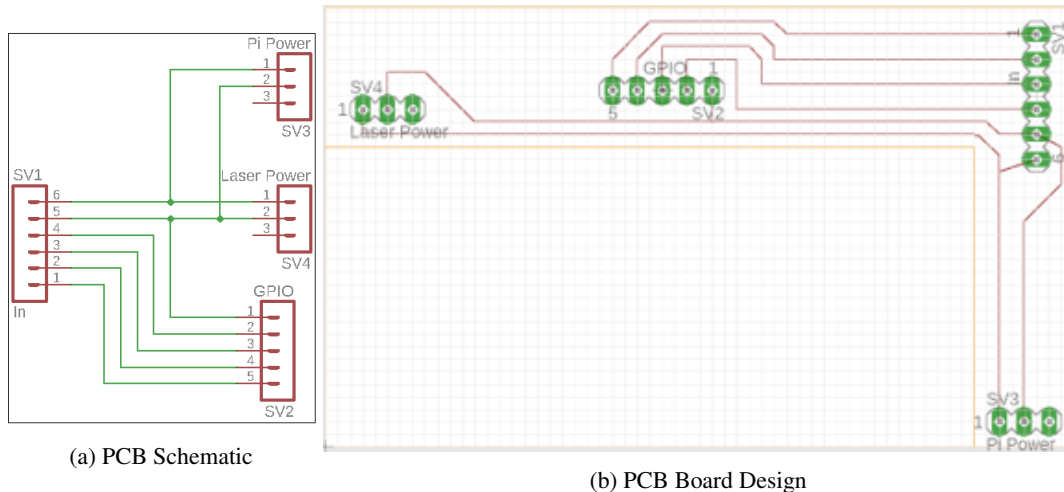


Figure 5.9: EagleCAD Files

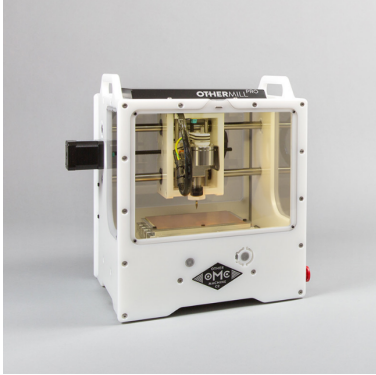


Figure 5.10: Bantam Tools OtherMill

To fabricate the PCB, a single sided FR-1 board and the Bantam Tools OtherMill milling machine (Figure 5.10) were used. The SCU Makerlab sold the FR-1 board and allowed use of the OtherMill machine. The FR-1 board has a thin copper layer on a base of fiberglass. The OtherMill mills out small strips for wires and can cut holes to shape the board and create pinholes. The process only takes about four minutes.

The process of manufacturing the boards in house was very convenient, but not incredibly reliable. The OtherMill takes less than 20 minutes from setup to clean up to mill out two boards. However the downside is reliability. The copper sheet is not always reliably applied, and while soldering pins and wires to the holes, several boards were broken from the heat of the soldering iron. The leads burned off very easily. In total, 4 different boards were used, and the last board had minor leads break off as well.

Chapter 6

The Final Product

While the final prototype did not meet every initially desired requirement, it showed that camera-based distance sensing with a laser does work as a technology.

6.1 Specifications

6.1.1 Met Requirements

The device dimensions ended up being 15 x 5 x 3.5 cm. This was just within the initial bounds of 15 x 5 x 5 cm, and made the device light and easy to maneuver.

The device draws 200 mA of current when running, which is well below the constraint of 250 mA. This allows it to run for a much longer time on just a battery.

The total cost of all parts used to construct the final device was only \$43, much less than the desired \$75. This production cost would make it easier for the device to compete in a real market.

6.1.2 Unmet requirements

While the initial software refresh rate was 1 sample every 12 seconds, it was reduced to 1 sample per second through software optimizations. However, this is still far lower than the desired 10 samples per second, and does not work for any real-time applications. Some suggestions to improve this sample rate will be discussed later.

The prototype outputs the distance information as a pulse-width modulated signal and as asynchronous serial messages, each at 3.3v. However, there was not enough time to implement i²c as it is more complicated and would likely require the writing of an API if the user had to be able to use it ergonomically.

The device is more than 90% accurate in ideal conditions, but only up to 4 meters. Some optimizations that might improve this range will be discussed later.

6.2 Accuracy

To test accuracy, several distance measurements at different distances were taken and compared the real and reported values. Figure 6.1 shows a scatter plot of the error in the prototype's measurements for distances up to 4.3 meters. The error remains under 9% for this entire range, with an average error of about 3.5%. Keep in mind, however, that at the higher distances, the device has a much more difficult time finding the dot at all.

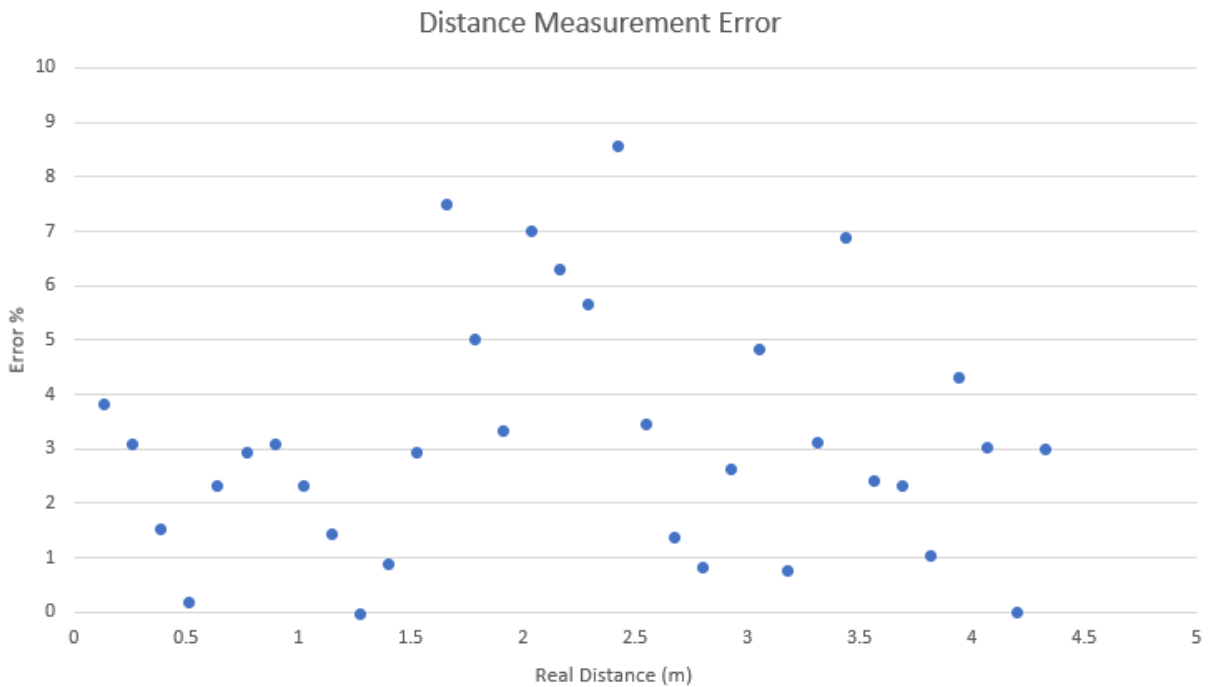


Figure 6.1: A scatter plot of the prototype's error at different distances.

Chapter 7

Budget

Because the total cost of parts for the device needed to be kept below \$75, the budget for the project was a particular concern. Table 7.1 shows a full account of expenses for the project. Table 7.2 shows the total cost of all parts used to build the final prototype. The total part cost of \$43 is well within the desired cost constraint, and it could likely be lowered even more by buying or manufacturing certain parts in bulk.

Item	Cost
2x Raspberry Pi Zero W	\$20
8Gb SD card	\$10
2x Camera	\$30
Laser	\$6
Circuit Boards	\$4
3D Printing	\$40
Tape Measure	\$10
Total	\$120

Table 7.1: Full Project Budget

Item	Cost
Raspberry Pi Zero W	\$10
8Gb SD card	\$10
Camera	\$10
Laser	\$6
Power Distribution Board	\$2
3D Printed Case	\$5
Total	\$43

Table 7.2: Prototype Part Cost

Chapter 8

Ethics

8.1 Ethical Justification for the Project

At first glance, a distance sensor does not appear to have a lot of ethical issues, as it is just a single component and is not very useful by itself. But looking through the IEEE Code of ethics, it appears that there is a lot more ethical considerations which must be taken. The first statement “ to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;” is most important to the final product, as it does use a laser which can harm people or animals if directed into eyes. Therefore a notice is displayed to the user of the dangers of laser pointers. Also, as when creating any electrical component, the circuits were built safely and robustly so that there are no shorts or anything else that could endanger the user or damage/destroy anything around it. The other statements mostly have to do with how we go about making the product. We must accept criticisms, and be honest about our results. It is vitally important that we are honest about reliability, accuracy, and working conditions as to not mislead a possible customer or the engineering community. We also must not steal work that is not our own, or mislead anyone during the design, creation, or implementation of our product.

8.2 Ethics, our Project, and What it Means to be a Good Engineer

This project was an opportunity to exercise good engineering ethics. Most importantly, it must be ensured that the device does not do less than stated. We very carefully measured the accuracy and tolerance of all final reported characteristics. This ethically covers our device in its use as part of a larger system or project. If the system using the device pushed some aspect of it to the reported limit, we can be sure that the device will not fail or be inaccurate

as long as its use is within the given parameters. As soon as the user tries to use the device for something beyond its limits, the liability of failure falls from us to them.

We will give our best effort to use parts that are ethically sourced, while still making an attempt to keep cost low for ourselves, and by extensions, the end user. By designing a device made of inexpensive parts, we can ensure that a manufacturer can still make a profit while still selling the device for what it is worth.

The ethical considerations of our project were not taught to us by the project itself, but inferred from some simple good will and common sense. When designing something for an end user, an engineer should never lie. Lie about designs can lead to all sorts of problems for the user. As far as our project goes, a lie, if told, has the potential to cause people physical harm, depending on the circumstances of how our device is used as part of a larger whole.

8.3 Ethical Pitfalls in the Project Itself

Every engineering project has issues of risk and safety to both the creators and the users, and while the project is relatively low risk, it is still critically important to identify and analyze those issues. Starting with safety, the two largest concerns are the laser and the circuitry. When purchasing any laser, there is usually a bright warning telling the user not to point directly at any other person. Laser pointers, such as the one in the project, can be very harmful when pointed in the eyes of a person or animal. It is important that there is a warning on the device to do our duty to help. As for the circuitry, it is important that there are no shorts which may destroy components. If there is too much current, it may destroy the components inside or even start a small fire, which is extremely dangerous. This means we, as creators must make sure the circuits are built to handle much more current than expected, as well as putting in current limiters or short protectors into the PCB. It is also wise to clearly label which pins are used for what to help the user prevent creating their own shorts.

The risk factors when building the device are pretty low. The components do not cost much, and cannot do a ton of damage if used with common sense. The real risk factor comes when using the product. It is very important that the accuracy and reliability is as good or better that what is claimed. The distance sensor is designed to be used for numerous applications, and some may rely very heavily on the accuracy that is claimed for the sensor. For example, if it is being used as a laser tape measure on a construction sight, a difference of a few inches could greatly jeopardize the construction process and in some cases, possibly the structural integrity of something that thousands of people could be using. The overall risk greatly depends on how the component is used in other engineering applications, so it is incredibly important that we are very truthful and accurate about our claims about the product, especially when it comes to accuracy and reliability.

The project has two layers of people that could be considered “the public”. First, there are fellow engineers that would

use the device in some project of their own. Although these engineers may be well-versed in the principles that make the device work and may even be able to guess about its underlying implementation, they still lack the full details of its conception, design, and construction. This means that in using the device in their own project, they are at our mercy ethically. These are the people that we are trying to benefit with the device and any lie about it nullifies this benefit.

The second set of people in the public are the end users of devices that use our device. If we lie to the engineers that use our device, they could then report use parameters that are false, and this deception of the end-user public would be on us, not the engineers.

Chapter 9

Sustainability

When most people think of the word “sustainability” they think of environmental impacts. While environmental sustainability is important, economic and social sustainability should also be examined. In this section, the ways in which the Camera-Based Distance Sensor impacts the planet (environmental), people (social), and profit (economic) will be examined.

9.1 Environmental Impact

The environmental impact of the Camera Based Distance Sensor is relatively low. Most of the impact is from manufacturing of electronics. The Raspberry Pi Zero W and the Camera used are very small in volume. All computer electronics do have an environmental impact. It is important to study the environmental impact of any product because in an increasingly globalized society, even the smallest object can have large carbon footprints, affect millions of people, and consume many natural resources. The components of the CBDS are: Raspberry Pi Zero, Camera, PCB, laser, and case. The Raspberry Pi, Camera, PCB, and laser can, for the most part be classified together as “computer hardware”, though the laser contains more metals and plastics.

During the manufacturing of computer hardware, mining rare material such as gold, tungsten, cobalt, and many others cause pollution and exploits the natural environment. Large mines that produce such materials are often devastating to local environments. The manufacturing process can create chemical and biological waste, polluting the environment and using up a lot of water. Most factories for the raw PCB material are in poorer countries in Asia and so the pollution unfortunately affects mostly very poor populations. The 3D printed material used is a Photopolymer Resin from FormLabs. This is more environmentally friendly than normal ABS or PLA plastics because it does not use volatile organic solvents that thermally cured polymerization does. Also, 3D printing does not waste very much material or create a carbon footprint from shipping like injection mold pieces may do. For most electronic devices,

such as computer or cell phones, the lithium ion batteries are the most devastating component to the environment. The CBDS does not have a battery built into it as it is a component, but it is expected that it would be used with a battery for most applications, so it is important to minimize the power draw of the system to further the longevity of any battery which it is used with. In an increasingly globalized economy, a large carbon footprint is created from shipping materials through supply chains. Most of the computer hardware used was likely shipped to the United States from China or other parts of Asia. Cargo ships and cargo planes both use large amounts of fuel, but are relatively efficient given how much they carry.

The CBDS is expected to work for at least 3-5 years, but may need maintenance or new parts. Lead, plastics, and other materials may be toxic to the environment if not properly disposed of. To minimize the effects, it is built to be modular. So if the camera mount, for example, fails, then a new one can be printed and the entire component does not need to be thrown away. This also makes the system upgradable. When a new Raspberry Pi Zero or camera comes out, the user can swap components without purchasing an entire new system. When disposing of old parts, users can recycle the 3D printed resin case, and the computer hardware can be recycled properly at e-waste bins. Through smart design, efficient use of resources, and modularity, the Camera Based Distance Sensor can have a very minimal impact on the environment.

9.2 Social Sustainability

Social sustainability concerns how useful and accessible a product will be for the users. When envisioning the CBDS, one of the largest goals is to make the project low cost. There are available options that are better than the device, but cost hundreds of dollars. It was also design to be as user friendly, and like other inexpensive distance sensors, the device is “plug and play” so there is no additional setup required. The only interface is input/output pins that cover power, Serial, and PWM communication. Also, unlike most competing distance sensors, the user can see exactly what the device is measuring because the laser is in a visible wavelength. All of the code is open source, and a user can use the micro-USB and HDMI ports on the Raspberry Pi Zero to change the code. Users are free to 3D print different cases or use the technology to look for multiple lasers. The same accuracy and range standards are not ensured if the code or case are tampered with, but users are free to do so.

9.3 Economic Sustainability

Low cost components were focused on to make the device work. For example, the Raspberry Pi Zero is only \$5 (or \$10 for the W with built in Wi-Fi) for a 1GHz processor and 512 Mb of RAM. A \$43 price point is in the higher end of low-cost distance sensors, but it has a much larger range and offers very high precision. Additionally, it supports

two different communications standards. The cost of the components are fairly low, none of which are more than \$15 each. Each of the pieces could likely be purchased at much lower price points if bought in bulk. The high cost of 3D printed materials could be replaced by injection molding if a high number of the devices were to be produced. If the device were to be produced at a very large scale, the Raspberry Pi Zero and the Camera could be replaced with a low cost FPGA or Integrated Circuit, and a more specialized sensor. As for long term use, this device should last for a long time as long as it is used properly. As technology progresses, newer versions of the Raspberry Pi Zero and the camera will probably come out, which can mean that the hardware is cheaper, better or both.

The goals of the Camera Based Distance Sensor primarily targeted social and economic sustainability, as it fills a market gap, is easy to use, and cheap to produce and modify. However, because of the small hardware and casing, there is very little environmental impact. This project is possible because of very good and inexpensive technologies such as computer hardware, digital camera sensors, and 3D printing. As these technologies continuously improve, the potential for an improved CBDS also improves. A better camera and more precise 3D printing methods would yield higher accuracy, and better or even specialized hardware such as the Raspberry Pi Zero would allow for faster processing, and the ability to use multiple lasers or cameras at once. Overall, it is hoped that this project will successfully fill a gap in the distance sensor market and will be used to help people in a variety of different industries and areas in a sustainable fashion.

Chapter 10

Conclusions and Future Work

The project was, on the whole, a success. A working prototype was constructed, and it was shown that distance sensing via a camera and laser can be done by a device. It was also proven that such a device can, with a few caveats, fill an accuracy/precision/range/cost gap in the distance sensing market.

While the prototype was functional, it was far from marketable. There are a number of improvements that would need to be made before a sensor using the same technology could be actually sold. Most of these improvements revolve around streamlining the design and removing superfluous elements.

The primary task of any group wishing to fully realize camera-based distance sensing technology would be to integrate the various parts of the final product into a single, unified device. This would likely require access to better circuit board and chip manufacturing processes. Most, if not all, of the software could be translated into hardware, which would make the sample rate much, much faster.

If a specialized camera could be designed, one that takes pictures at the exact resolution the algorithm requires, it would likely be smaller and be able to capture more quickly. It may not even need to take pictures at all as it could instead simply read the raw imaging data into memory, eliminating the need to write the image data and then read it immediately after.

If all of the components could be integrated into a smaller, unified device, it would likely be possible to completely eliminate the problem of laser/camera misalignment, which could make the device reliable at much further ranges.

Acknowledgements

We would like to thank everyone who helped us during the course of our project.

Thank you to our advisors, Dr. Sally Wood and Dr. Angela Musurlian, who gave us sage guidance at every step of the process.

Thank you to the SCU MakerLab and its staff for allowing us to use their resources and for helping us with the case and circuit board manufacturing.

Thank you to our employer, Delucchi Electric, Inc. for allowing us to use some of their resources.

Appendix

All of the code for our image processing algorithm can be found at the GitHub repository: <https://github.com/kaikalii/cbds.git>.

Bibliography

- [1] adafruit. Ir distance sensor gp2y0a710k0f.
- [2] adafruit. Ir distance sensor gp2y0s21yk0f.
- [3] adafruit. Maxbotix ultrasonic rangefinder.
- [4] adafruit. V16180x.
- [5] ElecFreaks. 2pcs hc-sr04 ultrasonic distance measuring sensor.
- [6] formlabs. The ultimate guide to stereolithography (sla) 3d printing.
- [7] formlabs Support. Using tough resin.
- [8] Garmin. Lidar-lite 3 laser rangefinder.
- [9] Brendan Hesse. Abs or pla: Which 3d printing filament should you use?, 2015.
- [10] Sean Higgins. Velodyne cuts vlp-16 lidar price to \$4k.
- [11] Bushra Mhdi, F Hamode Jamal, Nahla A. Aljaber, and Abeer H. Kalad. Laser distance measurement by using web camera. 08 2015.
- [12] Christian Portugal-Zambrano and Jesus Mena-Chalco. Robust Range Finder Through a Laser Pointer and a Webcam. *Electronic Notes in Theoretical Computer Science*, 281:143–157, 2011.