University of Illinois at Urbana-Champaign

3D Mapping using Laser Distancing

Design Review

Team Members
Soham Waychal
Vedanth Swain

Teaching Assistant
Yuchen He

ECE 445
October 14, 2016

Contents

List of Figures

1 Introduction

1.1 Statement of Purpose

To begin with, laser distancing is an prohibitively expensive technology for no reason. It requires a laser (a very cheap component) and some sort of camera (there are some cheap cameras in the range of \$10-\$15), and some basic trigonometry to calculate the distance. On top of that, 3D mapping is a difficult and open problem that has many solutions but none of them are complete or extremely robust. While we are not trying to make the greatest 3D mapping solution, we are aiming for a cheap and accessible solution that can be used by hobbyists for their robots to navigate and avoid obstacles.
Currently, there is no cheap 3D mapping solution. There have been various attempts at making a similar products using sonar and radar but both of these technologies have a major drawback that they waves spread out too quickly and cannot be used for precise mapping. We want to create a solution that is cheap, easy to make or access, and precise. Our project involves mounting a camera and a line laser in parallel on a stationary platform such that using a single stepper motor we can scan a certain angle in either horizontal/vertical direction.

1.2 Objectives

1.2.1 Goals and Benefits

- Decrease the cost of laser distancing
- Use laser distancing for 3D mapping
- Allow for an easy plugin for other devices to access the 3D map

1.2.2 Functions and Features

- Accurate motor movements
- Accurate laser distancing
- Syncing between microcontroller and CPU
- 3D point cloud using OpenGL

2 Design

2.1 Hardware Block Diagram

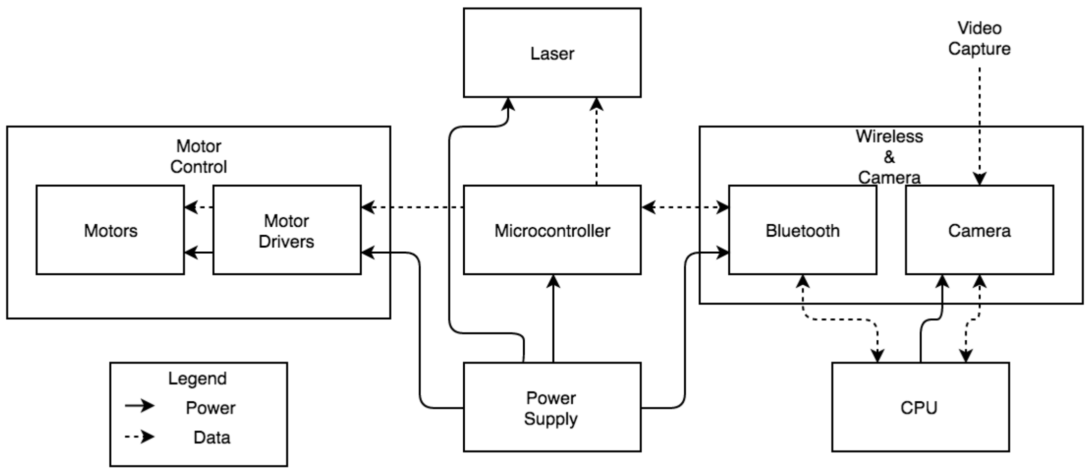
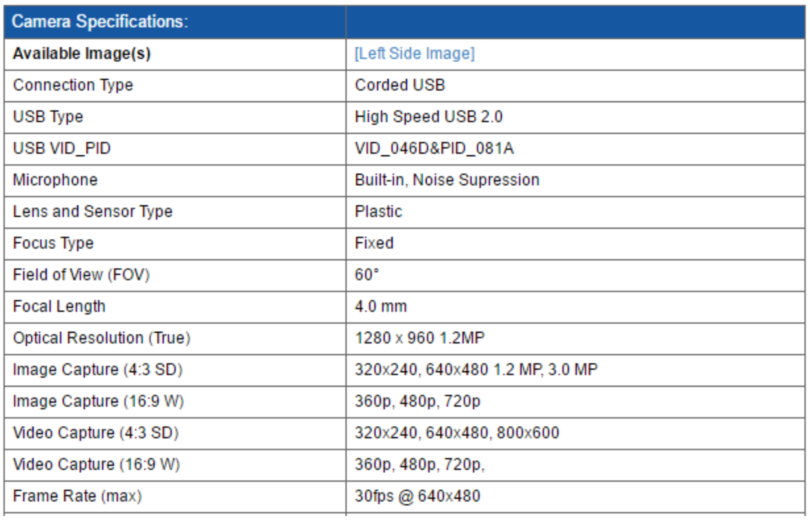


Figure 1: Hardware Block Diagram

2.2 Hardware Block Description

2.2.1 Webcam[4]

We will be using a Logitech C270 webcam for image capture. This camera captures 720p video stream and 3 megapixel images. We will not using the full capacity of this webcam since processing 345600 pixels will be quite CPU intensive and we do not need that many pixels to identify laser line and calculate distance to a set number of laser pixels. This webcam also has auto light correction which is incredibly helpful because this allows us to work in various light conditions. The webcam also comes with a built-in microphone and face tracking software that we will not be using for this project.

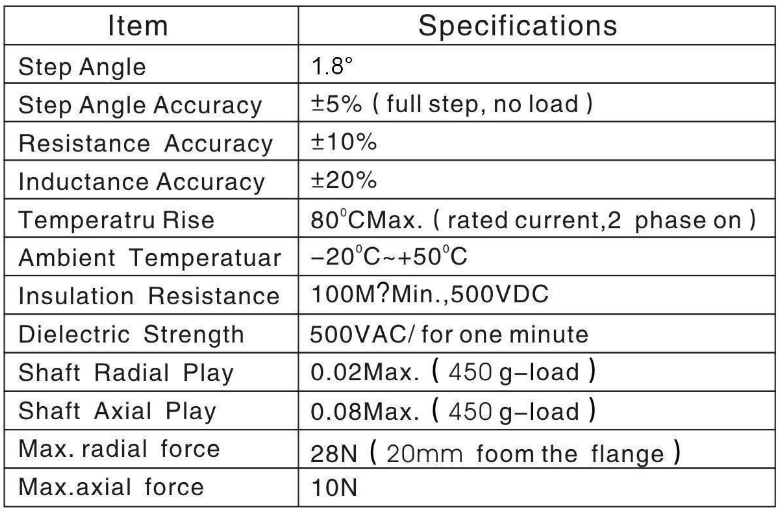


Camera Specifications:	
Available Image(s)	[Left Side Image]
Connection Type	Corded USB
USB Type	High Speed USB 2.0
USB VID_PID	VID_046D&PID_081A
Microphone	Built-in, Noise Supression
Lens and Sensor Type	Plastic
Focus Type	Fixed
Field of View (FOV)	60°
Focal Length	4.0 mm
Optical Resolution (True)	1280 x 960 1.2MP
Image Capture (4:3 SD)	320x240, 640x480 1.2 MP, 3.0 MP
Image Capture (16:9 W)	360p, 480p, 720p
Video Capture (4:3 SD)	320x240, 640x480, 800x600
Video Capture (16:9 W)	360p, 480p, 720p,
Frame Rate (max)	30fps @ 640x480

Figure 2: Logitech C270 Specifications

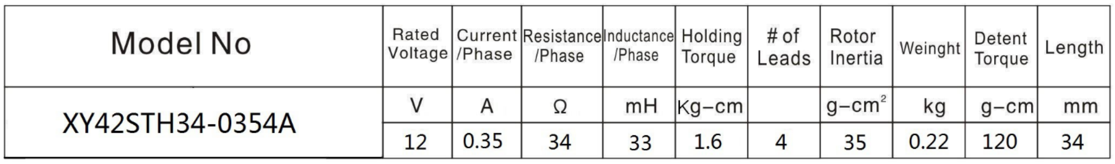
2.2.2 Stepper Motor[5]

We are planning on using the 17HS16-2004S, which is a high torque stepper motor, and it has a 1.8 degree step angle. They will receive the appropriate signals from the motor drivers and send position and velocity data back. We picked these motors because they were cheap, accurate, had high ratings, and could be easily integrated into our design.



Item	Specifications
Step Angle	1.8°
Step Angle Accuracy	±5% (full step, no load)
Resistance Accuracy	±10%
Inductance Accuracy	±20%
Temperatru Rise	80°CMax. (rated current,2 phase on)
Ambient Temperatuar	−20°C~+50°C
Insulation Resistance	100M?Min.,500VDC
Dielectric Strength	500VAC/ for one minute
Shaft Radial Play	0.02Max. (450 g−load)
Shaft Axial Play	0.08Max. (450 g−load)
Max. radial force	28N (20mm foom the flange)
Max.axial force	10N

Figure 3: Stepper Motor Specifications



Model No	Rated Voltage	Current /Phase	Resistance /Phase	Inductance /Phase	Holding Torque	# of Leads	Rotor Inertia	Weinght	Detent Torque	Length
XY42STH34-0354A	V	A	Ω	mH	Kg−cm		g−cm^2	kg	g−cm	mm
	12	0.35	34	33	1.6	4	35	0.22	120	34

Figure 4: Stepper Motor Holding Torque

2.2.3 Stepper Motor Driver[6]

The stepper motor drivers we shall be using are A3967 S. These drivers will be receiving signals for controlling direction and drive mode from the microcontroller. Then these signals are sent to the respective motors. We picked these motor drivers because they had an ample heat sink and could support up to 45V and up to 3amps which is more than enough to power the original idea of two motors. These stepper motor drivers were also highly rated which meant they did not have over-heating and thermal issues, short-circuits, or did not come damaged.

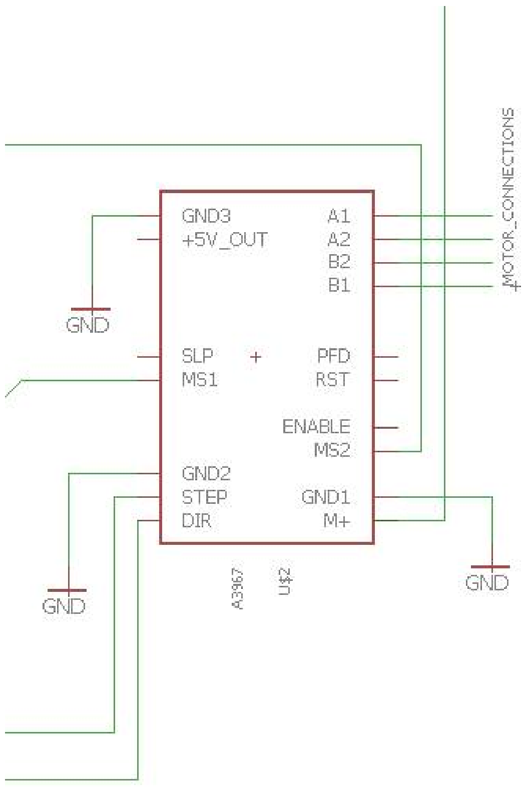
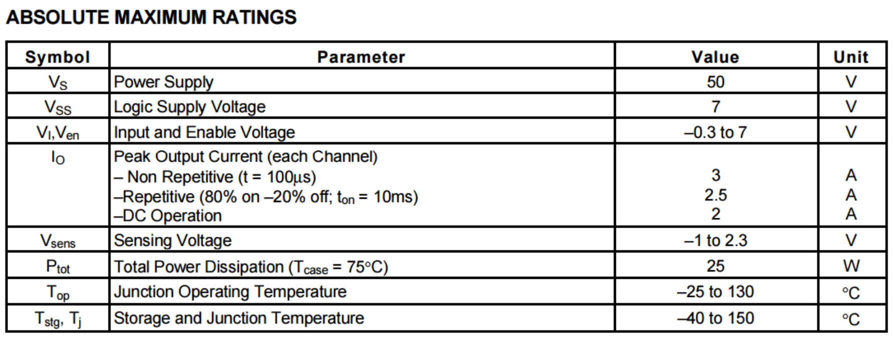


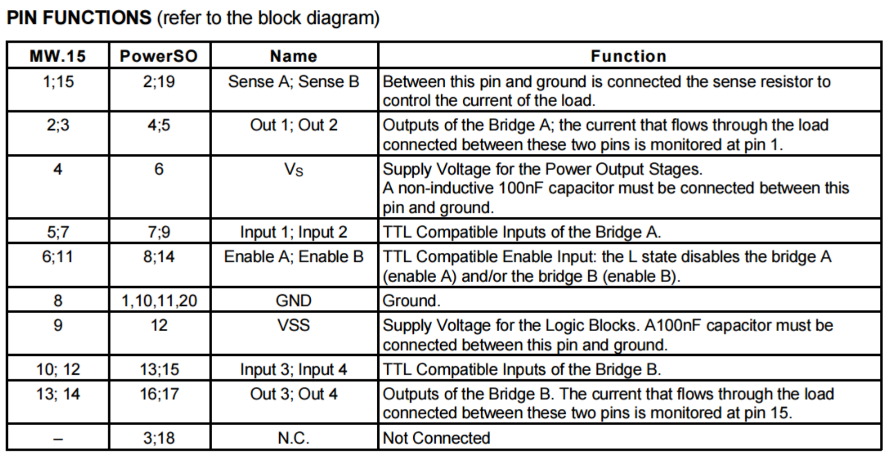
Figure 5: Stepper Motor Driver Schematic



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

Figure 6: Stepper Motor Driver Maximum Ratings



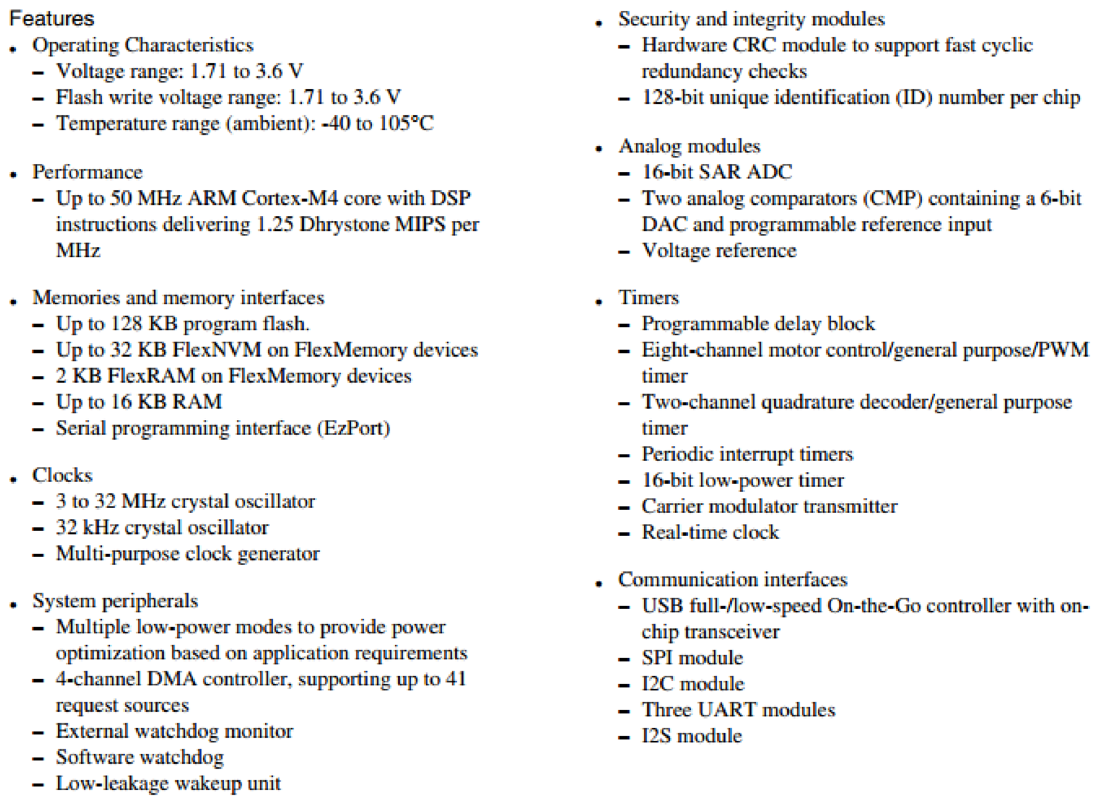
PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

Figure 7: Stepper Motor Driver Pin Functions

2.2.4 Microcontroller[7],[8]

For the microcontroller we are currently looking at using MK20DX256. It checks the current positoin of the stepper motors and sends the appropriate signals to the drivers. It controls the switching on and off of the laser. And it also sends the position information to the android phone via the HC-06 Bluetooth module. The microcontroller has extensive documentation, various compilers and loaders, and is overall a microcontroller that lots of people have used for their hobbyists projects. The MK20DX256 also has more RAM, clock speed, off, program flash RAM, and internal RAM than ATmega328 chip (another commonly used microcontroller).



Features

- Operating Characteristics
  - Voltage range: 1.71 to 3.6 V
  - Flash write voltage range: 1.71 to 3.6 V
  - Temperature range (ambient): -40 to 105°C
- Performance
  - Up to 50 MHz ARM Cortex-M4 core with DSP instructions delivering 1.25 Dhrystone MIPS per MHz
- Memories and memory interfaces
  - Up to 128 KB program flash.
  - Up to 32 KB FlexNVM on FlexMemory devices
  - 2 KB FlexRAM on FlexMemory devices
  - Up to 16 KB RAM
  - Serial programming interface (EzPort)
- Clocks
  - 3 to 32 MHz crystal oscillator
  - 32 kHz crystal oscillator
  - Multi-purpose clock generator
- System peripherals
  - Multiple low-power modes to provide power optimization based on application requirements
  - 4-channel DMA controller, supporting up to 41 request sources
  - External watchdog monitor
  - Software watchdog
  - Low-leakage wakeup unit
- Security and integrity modules
  - Hardware CRC module to support fast cyclic redundancy checks
  - 128-bit unique identification (ID) number per chip
- Analog modules
  - 16-bit SAR ADC
  - Two analog comparators (CMP) containing a 6-bit DAC and programmable reference input
  - Voltage reference
- Timers
  - Programmable delay block
  - Eight-channel motor control/general purpose/PWM timer
  - Two-channel quadrature decoder/general purpose timer
  - Periodic interrupt timers
  - 16-bit low-power timer
  - Carrier modulator transmitter
  - Real-time clock
- Communication interfaces
  - USB full-/low-speed On-the-Go controller with on-chip transceiver
  - SPI module
  - I2C module
  - Three UART modules
  - I2S module

Figure 8: MK20DX256 Features

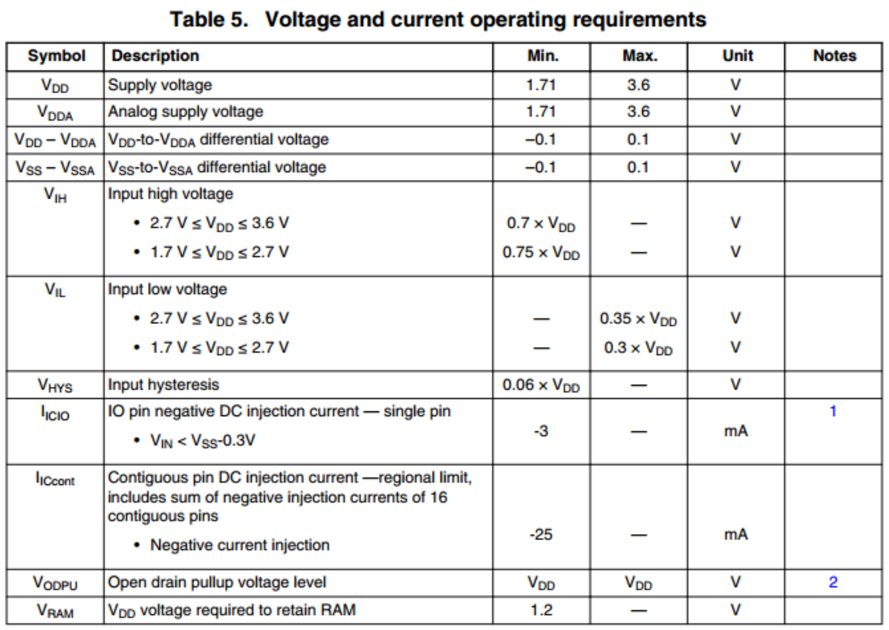


Table 5. Voltage and current operating requirements

Symbol	Description	Min.	Max.	Unit	Notes
V_{DD}	Supply voltage	1.71	3.6	V	
V_{DDA}	Analog supply voltage	1.71	3.6	V	
$V_{DD} – V_{DDA}$	V_{DD}-to-V_{DDA} differential voltage	–0.1	0.1	V	
$V_{SS} – V_{SSA}$	V_{SS}-to-V_{SSA} differential voltage	–0.1	0.1	V	
V_{IH}	Input high voltage				
	• $2.7 V \le V_{DD} \le 3.6 V$	$0.7 \times V_{DD}$	—	V	
	• $1.7 V \le V_{DD} \le 2.7 V$	$0.75 \times V_{DD}$	—	V	
V_{IL}	Input low voltage				
	• $2.7 V \le V_{DD} \le 3.6 V$	—	$0.35 \times V_{DD}$	V	
	• $1.7 V \le V_{DD} \le 2.7 V$	—	$0.3 \times V_{DD}$	V	
V_{HYS}	Input hysteresis	$0.06 \times V_{DD}$	—	V	
I_{ICIO}	IO pin negative DC injection current — single pin • $V_{IN} < V_{SS}$-0.3V	-3	—	mA	1
I_{ICcont}	Contiguous pin DC injection current —regional limit, includes sum of negative injection currents of 16 contiguous pins • Negative current injection	-25	—	mA	
V_{ODPU}	Open drain pullup voltage level	V_{DD}	V_{DD}	V	2
V_{RAM}	V_{DD} voltage required to retain RAM	1.2	—	V	

Figure 9: MK20DX256 Maximum Rating

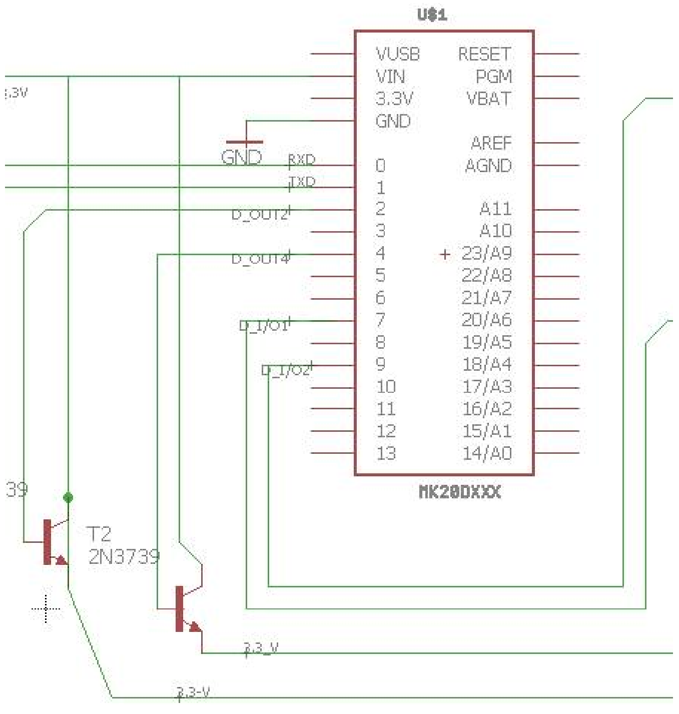
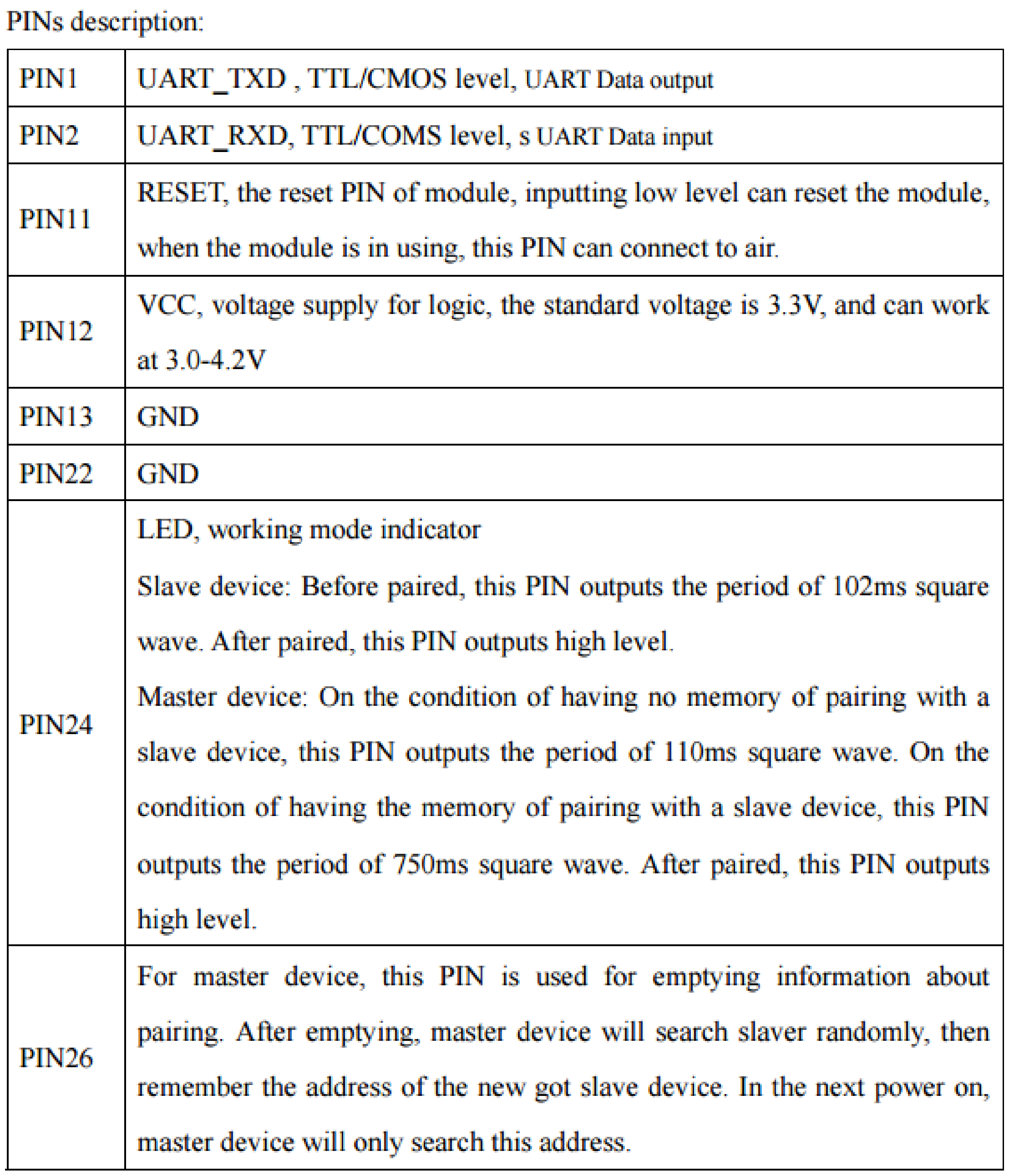


Figure 10: MK20DX256 Schematic

2.2.5 Bluetooth[9]

We will be using the HC-06 Bluetooth module for transmitting the position of the stepper motor per time-step to the android phone via the microcontroller. We picked this Bluetooth module because we have worked with it several times in the past and it is a very reliable and heavily documented piece of hardware.



PINs description:

PIN1	UART_TXD , TTL/CMOS level, UART Data output
PIN2	UART_RXD, TTL/COMS level, s UART Data input
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND
PIN22	GND
PIN24	LED, working mode indicator Slave device: Before paired, this PIN outputs the period of 102ms square wave. After paired, this PIN outputs high level. Master device: On the condition of having no memory of pairing with a slave device, this PIN outputs the period of 110ms square wave. On the condition of having the memory of pairing with a slave device, this PIN outputs the period of 750ms square wave. After paired, this PIN outputs high level.
PIN26	For master device, this PIN is used for emptying information about pairing. After emptying, master device will search slaver randomly, then remember the address of the new got slave device. In the next power on, master device will only search this address.

Figure 11: HC-06 Pin Descriptions

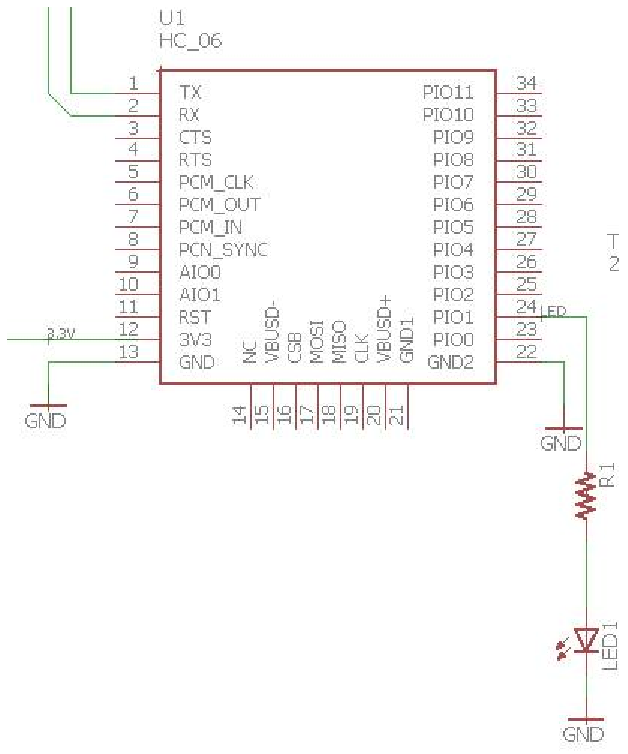


Figure 12: HC-06 Schematic

2.2.6 Laser[10]

We will be using the gn-0007 focusable red light laser. This will be mounted with and parallel to the camera and is controlled by the microcontroller. It's wavelength is 650nm, its operating voltage is 3-5V and the current is less than 45mA. This translates to a range of [0, 100, 100] to [10, 255, 255] in lower HSV values and [150, 200, 100] to [179, 255, 255] in upper HSV values. This also translates to [17,15,100] to [50,56,200] in RGB values.

2.3 Power and Consumption Calculation

For our power supply we will be connecting LiPo batteries in series so that we get a power supply of voltage greater than 12V. And we shall step these down using an L7812 voltage regulator whose output voltage is 12V for the motor. And a LD1117V33 which steps down the power supply to 3.3V for the rest of the circuit to function.

Part	Voltage	Current	Average Power Consumption
Laser	3-5V	35-45mA	40mA
Motor Driver logic	3-5.5V	0.02mA	0.02mA
Stepper Motor	12-13V	250-350mA	300mA
Bluetooth	3.1-4.2V	30-40mA	35mA
Microcontroller	1.7-3.6V	35-45mA	30mA
Total			405.02mA

For a regular 2200mAH LiPo battery, we can run this setup for $\frac{2200mAH}{405.02mA} = 5.43Hours$

2.4 Hardware Schematic

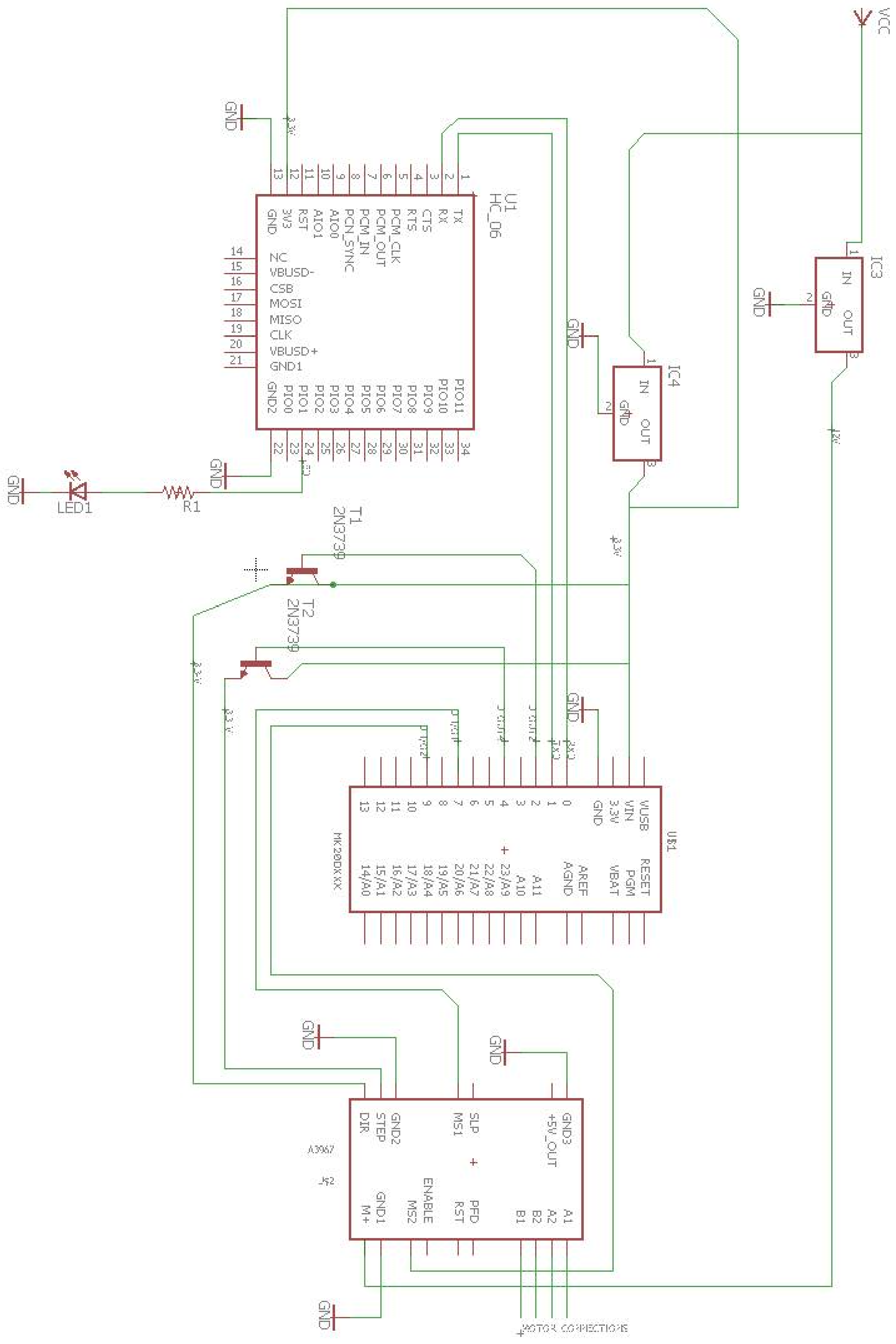


Figure 13: Hardware Schematic

2.5 Software Block Diagram

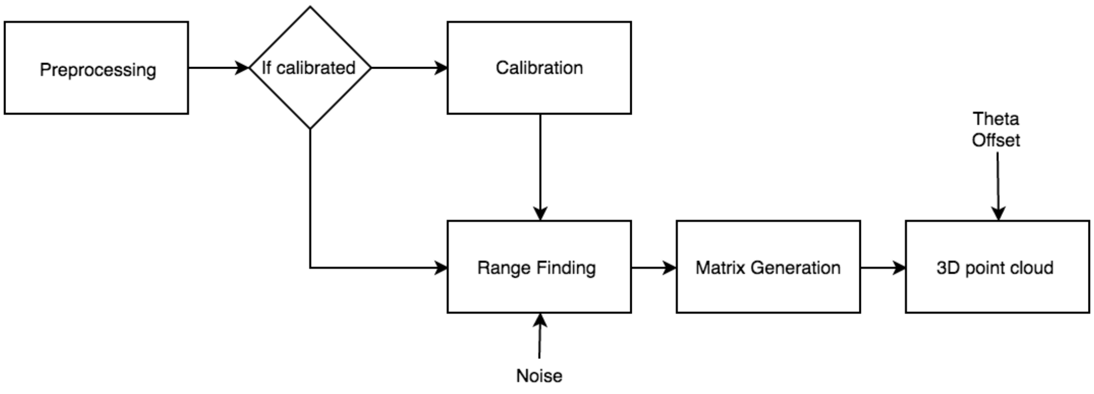


Figure 14: Software Block Diagram

2.6 Software Block Description

2.6.1 Pre-processing[12,[13]]

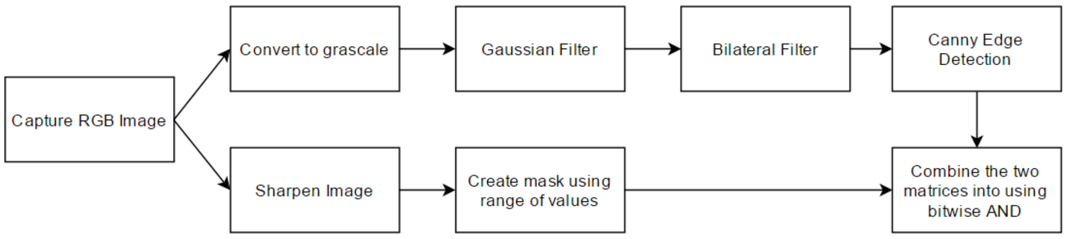


Figure 15: Image Preprocess Pipeline

We start with one RGB frame captured from the webcam. To do Canny Edge Detection, there are several preprocesses that are recommended to reduce noise and get cleaner lines. The first step is to covert the RGB image into grayscale since we do not need the extra color channel information. After this we apply a Gaussian filter which blurs the image by convolving a fixed matrix with the image. The next step is to apply a bilateral filter which is a non-linear, edge-preserving filter which also reduces noise. Each pixel when applying a bilateral filter is replaced by a weighted average from the nearly by pixels - however, the weighted average does not only rely on Euclidean distance but also radiometric differences between the pixels. The final step is to apply the Canny Edge detector which has the following steps:

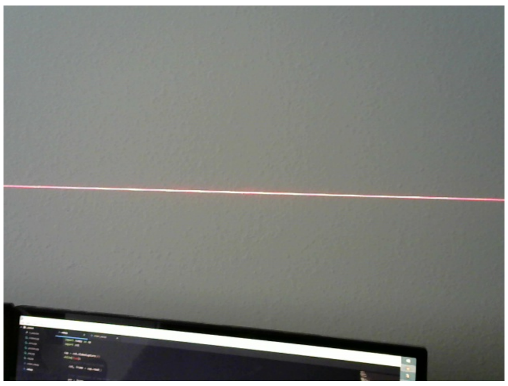
- Find intensity gradients of the image -
  - Apply a pair of convolution masks in the x and y directions to get G_x and G_y
  - Find the gradient magnitude by $G = \sqrt{G_x^2 + G_y^2}$
  - Find the gradient angle by $\theta_G = \arctan \frac{G_y}{G_x}$
- Apply non-maximum suppression (edge-thinning technique) - basically sharpens the blurred edges from the filters. The algorithm is:
  - Compare edge strength of current pixel with edge strength of the pixel in positive and negative gradient directions

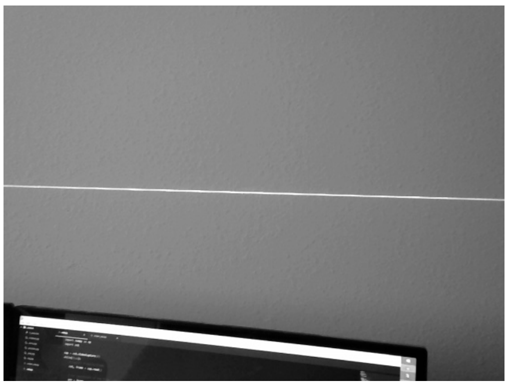
  - If edge strength of the current pixel is the largest compared to other pixels with same gradient directions, the value will be preserved and the rest will be suppressed
  - This ensures that pixels with lower magnitudes (blurred pixels on the outside of edge) get removed or lowered while the real edge gets sharpened

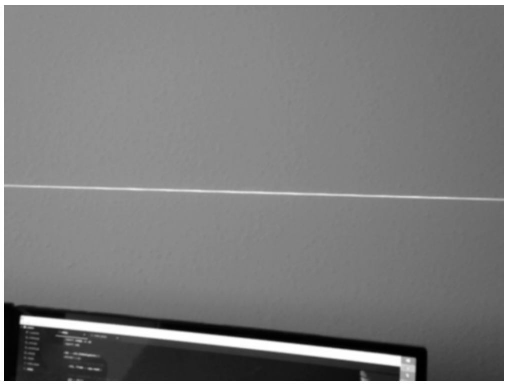
- Apply double threshold to determine potential edges - this step removes pixels caused by noise and color variations
  - This is done by setting two threshold values - high and low
  - If a pixel's gradient value is less than the low threshold value, it will be suppressed; otherwise it is left alone

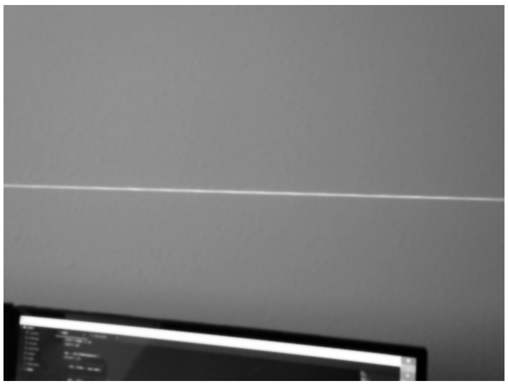
- The final step is to do edge tracking is to remove remaining weak pixels. This is done as following:
  - If a weak edge pixel is not surrounded by at least one strong edge pixel in it's 8-connected neighborhood pixels, it is removed.

After performing Canny Edge detection, we will have edges of all edges in the image. However we only want the red lines corresponding to the laser. This is done by first sharpening the image which involves convolving a sharpening matrix with the image. Next, the sharpened image is searched pixel by pixel in range of some threshold values of red. Finally, the two images, the Canny Edge detected image and search for red image, are combined by using a bitwise AND to produce the final result that displays the laser.


(a) RBG


(b) Gray


(c) Gaussian Filtered Applied Image


(d) Bilateral Filtered Applied Image

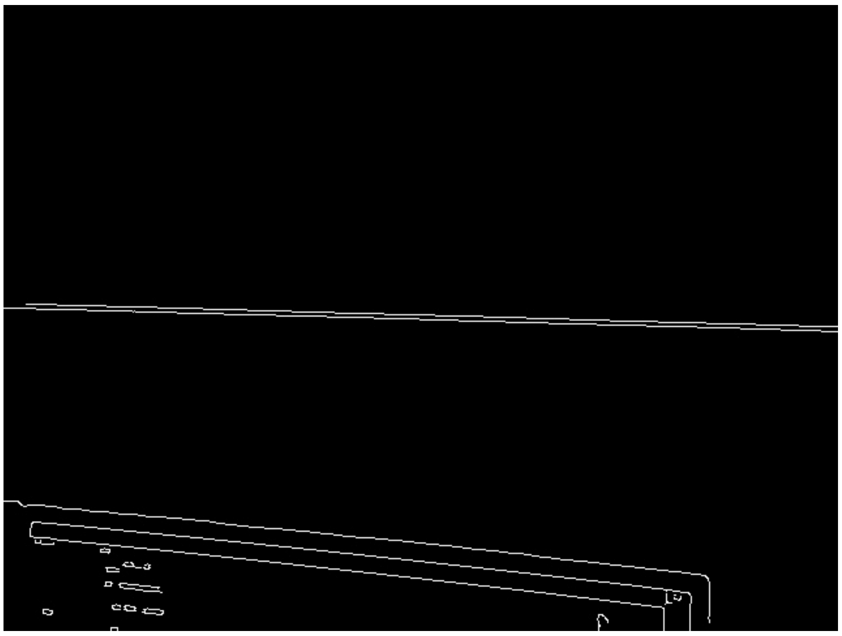
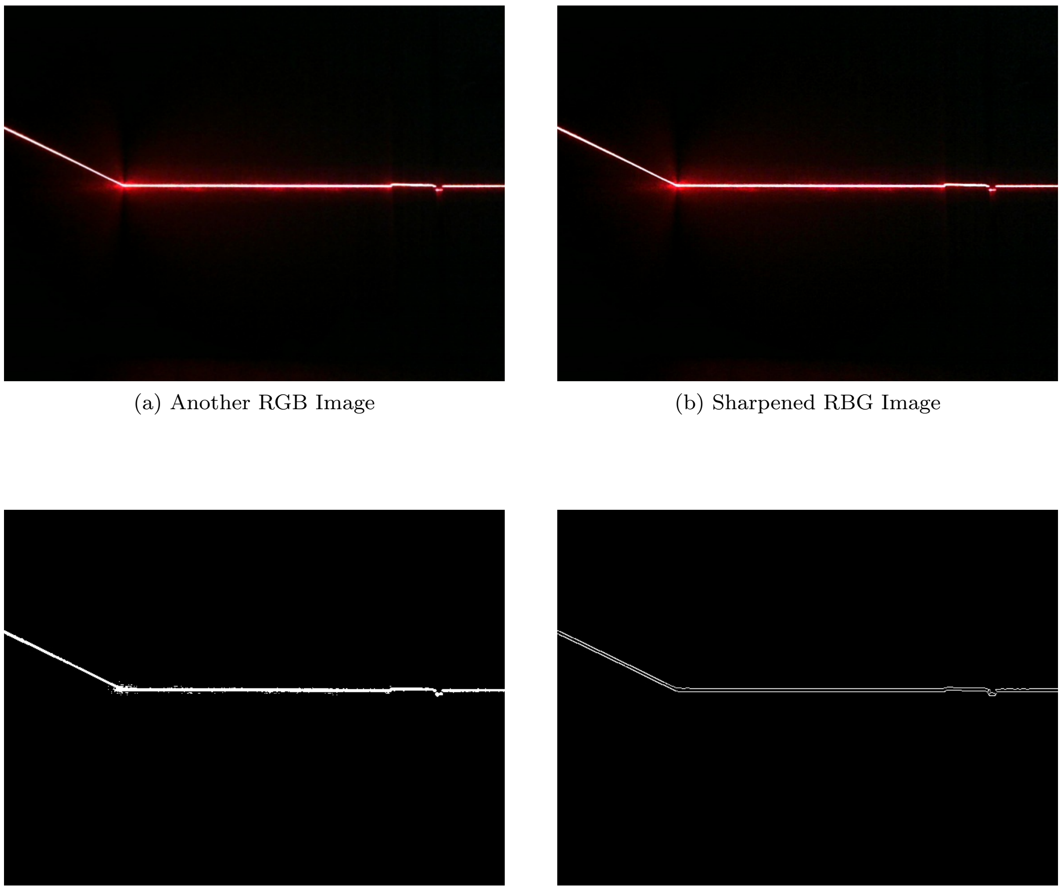


Figure 15: Canny Edge Detector Applied Image

Here, we present another example in a darkened room with image sharpening, red detection, and edge detection.



(a) Another RGB Image

(b) Sharpened RBG Image

(c) Red Detection

(d) Edge Detection

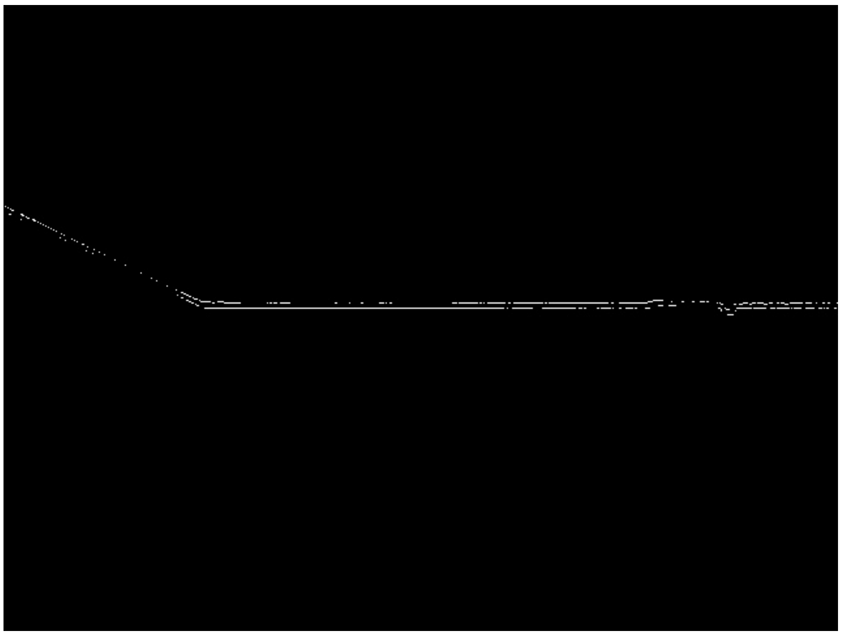


Figure 16: Final Image using Red Detection and Canny Edge Detection

2.6.2 Camera Calibration

Before we start the distance calculation we need to calibrate the camera (i.e. calculate the camera's *rpc* and *ro*). For this we will conduct a set of measurements where we know the distance to the target and also the the number of pixels where the red light is from the center of the image. After we get the actual theta we set up a linear relationship with number of pixels from image center and get the values for *ro* and *rpc*. Now the question arrives, what happens in different brightness setting and lighting environments.
To make sure this doesn't affect our calibration we take two simultaneous images (one without the laser and one with it). Then we normalize the histograms of the two, remove the blue and green channel to get rid of additional noise and then subtract one from another. This would give us a final image with a bright laser line. If there is any other noise on the background, we can remove them by calculating the mean and standard deviation of the line and ignore anything that's outside *mean* - *stddeviation*.

2.6.3 Camera-Microcontroller Sync

We want our camera to know when the microcontroller is going to turn the laser or when its going to shift the position of the stepper motors. We intend to send this data via bluetooth, hence we have to make sure the camera and the hardware system are in sync for accurate generation of the point cloud model.

2.6.4 Distance Calculation

This is the most crucial part of out experiment. After we have successfully calibrated the camera we know the theta from the image formed in the camera's focal plane. We also know the set distance between the laser and the android phone. So using trigonometric identities we can calculate the actual distance to the target. The current range of values for our algorithm (min/max) have been mentioned in the tolerance analysis. Once we have that we store the value and move on to point cloud generation.

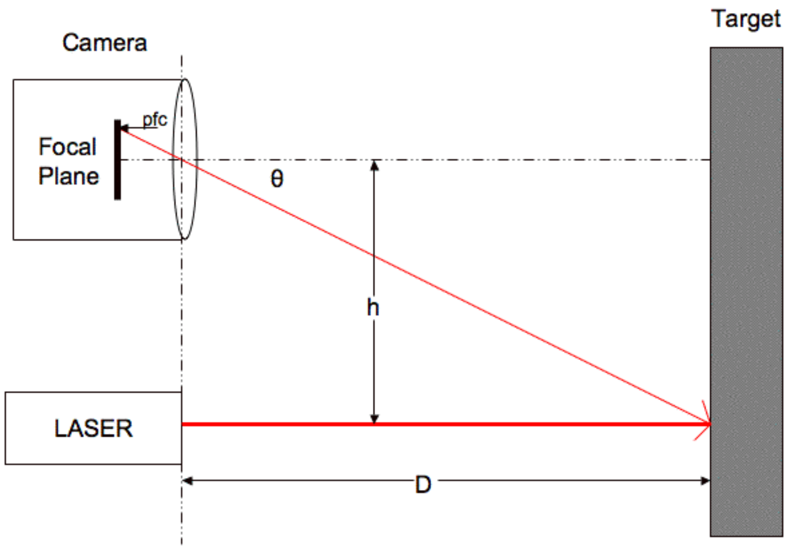
$$D = \frac{h}{\theta} = \frac{h}{\tan(pfc * rpc + ro)}$$ [1]

Where:

pfc = Number of pixels from center of focal plane

rpc = Radians per pixel inch

ro = Radians offset (compensates for alignment errors)



2.6.5 Point Cloud Generation[11]

After getting the set of distances we store it in a matrix. We also have all the position data from the microcontroller which we send via bluetooth. We will then be using OpenGL to create a 3d point cloud. OpenGL supports projection mapping; here we will explain and show how the transformation from a 3D point in some imaginary place to a 2D point on the monitor screen. To begin, take a look at these two pictures:

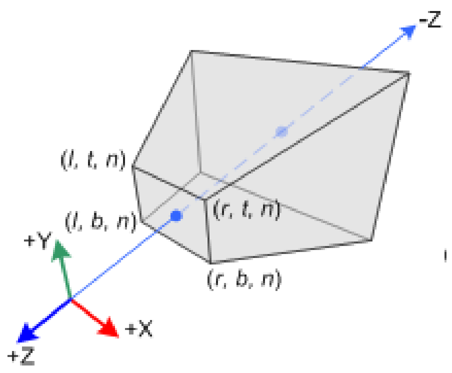


Figure 17: Perspective Frustum for Projection Mapping

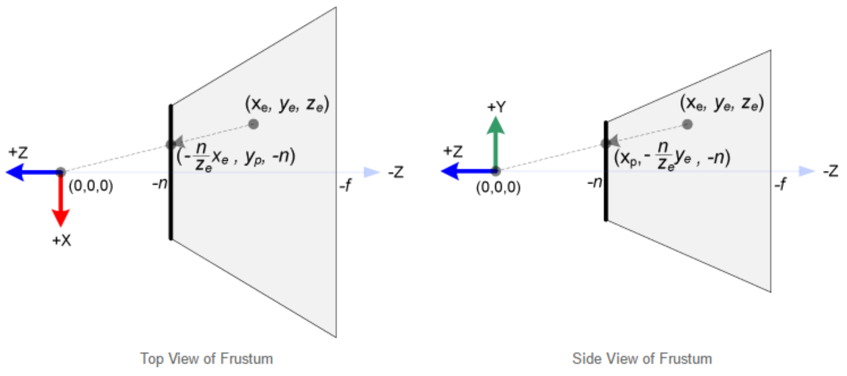


Figure 18: Top and Side View of the Frustum for Projection Mapping

As you can see, we are trying to map x_e to x_p which is on the screen. Next,

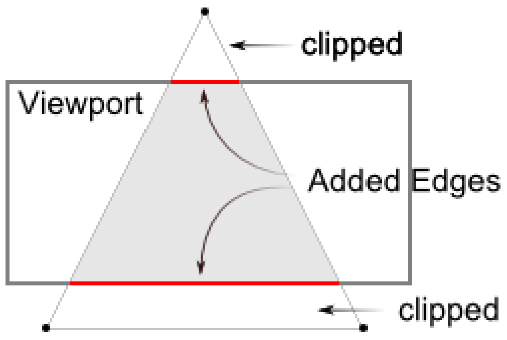


Figure 19: How projection should work

Therefore, we have to transform vertex data into clip coordinates into normalized device coordinates (NDC). The frustum culling (clipping) is performed in clip coordinates before dividing by w_c which is the normalization factor. In addition if if any clip coordinate is less than $-w_c$ or greater than w_c, it will be discarded as it is out of projection range.
We can finally define our projection matrix.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = M_{projection} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}$$

and

$$\begin{bmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{bmatrix} = \begin{bmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \end{bmatrix}$$

From Figure 18 we can see that $\frac{x_p}{x_e} = \frac{-n}{z_e}$ and $x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e}$. Note that x_p (and also y_p) is inversely proportional to $-z_e$. Using this, we can set the last line in our projection matrix since we have learned that $w_c = -z_e$.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}$$

After this, the calculations for the first three lines are a bit involved. However, the first two lines are extremely similar as you are dealing with inverse axis so we will only show calculation for x axis. The rest (and more detailed) calculations can be found in the references.To calculate the first line of the projection matrix, we will first map x_p to x_n such that $[l, r] = [-1, 1]$.

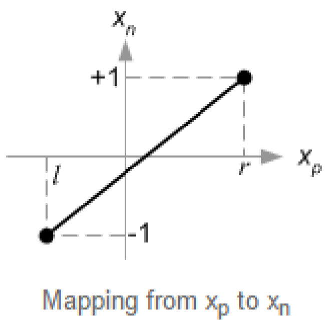


Figure 20: Mapping from projection to normal coordinates

Following the model we have can setup,

$$\begin{aligned} x_n =& \frac{1-(-1)}{x-l} \cdot x_p + \beta \\ 1 =& \frac{2r}{r-l} + \beta && \text{substitute } (r,1) \text{ for } (x_p, x_n) \\ \beta =& 1 - \frac{2r}{r-l} = \frac{r-l}{r-l} - \frac{2r}{r-l} = -\frac{r+l}{r-l} \\ x_n =& \frac{2x_p}{r-l} - \frac{r+l}{r-l} \\ =& \frac{2 \cdot \frac{n \cdot x_e}{-z_e}}{r-l} - \frac{r+l}{r-l} && \text{substitute } x_p = \frac{n \cdot x_e}{-z_e} \\ =& \frac{2n \cdot x_e}{(r-l)(-z_e)} - \frac{r+l}{r-l} \\ =& (\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e)/(-z_e) \end{aligned}$$

From this, we can fill in the first two rows of the projection matrix

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}$$

where t and b correspond to axis when converting from y_p to y_n. Finally to find the 3rd row, note that z_e in eye space is always projected to $-n$ which is the distance inside the monitor. As z does not depend on x and y, we can use the relationship between z_n and z_e to find a set of equations to calculate the last two entries of the 3rd row. As such, the projection matrix will look like this

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}$$

where

$$z_n = \frac{z_c}{w_c} = \frac{Az_e + Bw_e}{-z_e} = \frac{Az_e + B}{-z_e}$$

since $w_e = 1$ in eye space. To find A and B, we can use the (z_e, z_n) relationship in the coordinate plane of $(-n, 1)$ and $(-f, 1)$ to get the following set of equations

$$\begin{aligned} -An + B &= -n \\ -Af + B &= f \end{aligned}$$

Finally solving for A and B, we get $A = -\frac{f+n}{f-n}$ and $B = -\frac{2fn}{f-n}$. Putting these values in the projection matrix, we get the final matrix which is

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

3 Requirements and Verification

3.1 Table of Hardware Requirements and Verification

Requirements	Verification
Webcam 1. (5)Capture images form an input video stream and save it on computer. Check brightness and contrast in different light settings.	Verification for Webcam 1. Verification for 1. (a) Send a capture signal to the web cam. Check framerate via OpenCV (has inbuilt frame rate monitor). Make sure the framerate does not drop below 2 frames/sec after processing through visual confirmation. (b) The brightness and contrast calibration will be done by the webcam through its internal processes. We have no control over this part and will not be able verify this. (c) Make sure the image is successfully sent to the computer via USB. This is confirmed by visually confirming where the webcam is pointing and the picture on the screen.
Stepper Motor & Drivers 1. (3)Motor should receive the following voltage and current to rotate an object. (a) 12-13V. (b) 250-350mA. 2. (2)Rotate an object consistently at max by 5 degrees per step with a holding torque of 1.6 N-cm	Verification for Stepper Motor & Driver 1. Measure the output voltage and current of the motor from the motor driver by. (a) Placing a voltmeter in parallel to the load and verify the values. (b) Placing an ammeter in series with the load and verify the values. 2. Verification for 2. (a) Calculate weight of entire setup. (b) Calculate torque through radius of lever arm times weight to get necessary torque. (c) Ensure that the weight is below required torque. Reduce parts if necessary. (d) Make sure the motor performs 72 steps in a full rotation of 360 degrees by stepping through each rotation, measuring the total final degrees turned, and dividing it by 72 to get average degree turned per iteration. Calculate percentage error from 5 degrees.

Bluetooth Module 1. (3)The device can pair with the module within 1/2 meter. This will be the max range range in which we will test the setup. 2. (2)Receiver should receive data 90% of the time via UART and within 50ms. This is to ensure if we have any dropped data, we will not lose accuracy.	Verification for Bluetooth module 1. Verification for 1. (a) Connect the module to an arduino. (b) Pair up with the receiver 1/2 meter away. (c) Connect LED to the arduino via one of the output pins. (d) The LED should light up it it pairs successfully. 2. Verification for 2. (a) Pair the module with a phone. (b) Probe the RXD pin. (c) Send an integer or string on the phone. (d) Make sure it appears on the probe at least 9/10 times. (e) Have timer on computer to insure that the data transmission does not take more than 50ms. If it does, there is something wrong with the connection and we will have to fix that.
Power supply 1. (3)Ensure the power supply is in the range for the following components: (a) Laser 3-5V and 35-45 mA (b) Motor drivers logic 3-5.5V (c) Bluetooth 3.1-4.2V and 30-40 mA	Verification for power supply 1. Check that the voltage and the current at each output is sufficient for each module (a) Verification for laser i. Connect voltmeter in parallel to the laser and check the voltage displayed. ii. Check the current into the laser by using an ammeter in series. (b) Verification for Motor Driver logic i. Check the voltage by plugging in a voltmeter and verify the displayed voltage is in range. ii. Similarly plug in an ammeter to see if there is a current high. (c) Verification for Bluetooth i. Check the voltage by reading verifying the values observed from a voltmeter plugged in to the Vcc. ii. Check the current flow by plugging in an ammeter, and verify the values.
Laser 1. (1)Laser operating voltage is 3.6-5V. 2. (1)Don't point the laser at reflective or extremely absorbent surfaces.	1. Attach voltmeter across the laser and measure its in the range. 2. Extremely reflective and absorbent surfaces will not work with the laser due to lots of refraction or the laser not being visible. We will verify this by making sure the surface we project the laser onto is not reflective or light absorbent.

Requirements	Verification
Microcontroller 1. (3)Communication with the Bluetooth module via UART with 90% accuracy.We need 90% accuracy in case of dropped information along any path ways. The system should also be able to communicate in less than 50ms. 2. (3)Communication with stepper motor driver with 90% accuracy and takes less than 50ms (should be instant). 3. (3)Communication with the laser with 90% accuracy. This is again for any dropped information that may cause bugs. The system should take less than 50ms. 4. (6)Insure that processing time for bluetooth, stepper motor, and laser should be less than time between motor turns.	Verification for the Microcontroller 1. Verification for 1. (a) Probe the Bluetooth module's RXD pin through putty on computer. (b) Send ten ascii bytes from the microcontroller and make sure it receives on the computer at least 9/10 times. (c) Have a timer running on computer that averages the time it takes for 10 ascii bytes to arrive on computer. 2. Verification for 2. (a) Connect the motor driver to the microcontroller pin. (b) Send signal to activate the motor. (c) Ensure that the motor rotates physically 9/10 times and that it is less than 50ms by having a timer in the program. 3. Verification for 3. (a) Connect the laser to a power supply which supplies 3.6-5V via a BJT. (b) Pass a high through one of the the output pins of the microcontroller. (c) Makes sure that the laser turns on 9/10 times. Calculate time between on and off by having a timer. 4. Verification for 4. (a) The processing time is depended upon the clock cycle which for this microprocessor ranges up to 50MHz. Since we are not processing image data over Bluetooth, we will only require the microprocessor to process sync data, stepper motor turning angle and speed data, and laser power switch which should be turned on at the beginning of the mapping and should be turned off at the end. (b) Write timing program the compares runtime of sending image over USB, image processing, and mapping on the computer to bluetooth sync and motor control runtime on the microcontroller. Insure that the microcontroller processes data fast enough so that we do not move motors or drop sync bits over bluetooth.

3.2 Table of Software Requirements and Verification

Requirements	Verification
Image Processing 1. (10)The image, after the preprocessing is complete, should only contain the laser with all other extraneous noise filtered out. Extraneous information will be all other elements in image such as the background, other red objects, and other edges that we detect. 2. (10)The run-time video feed after the processing of each image should be a minimum of 5 fps.	Verification for image processing 1. Verification for 1. (a) Capture/Input image into the program. (b) Run the script which has all the filtering algorithm (see Image Processing in Software Description) (c) Visually confirm that there are no extraneous pixel values except for the laser in the resultant image. 2. Verification for 2. (a) Display counter of the run-time fps using OpenCV in the script. (b) The reason to use real-time run-time capture is because OpenCV when processing video holds camera open for the entire duration, but when dealing with image capture, shuts down camera after each picture take. In a sense, it would go camera open - image capture - camera close in a loop. Instead if we are just capturing video, camera will stay open. The frames per second is important is because if we drop below minimum requirements, we will not capture a frame per stepper motor rotation. (c) Run the script and make sure the fps is higher that 5.
Distance Calculation 1. (10)Convert Pixel location of the laser into actual distance and measure for 0.5, 1, 2, 4, 7 meters. The error shouldn't be more than 20% of actual distance for less than 4 meters (Check tolerance analysis for more details). 2. (5)Make sure the above distance calculation runs in dark and medium light settings, with a total of 80% accuracy.	Verification for Distance 1. Verification for 1. (a) Load all the images into the computer with the laser pointing at different distances. (b) Use the algorithm to generate distance matrix. (c) Compare with theoretical distance. 2. Verification for 2. (a) Take different 5 images with the line laser in each light setting. (b) Input the images into the algorithm that computes the distance matrix. (c) After getting the theoretical distance, make sure there is no more than 20% error form the actual distance. (Distance calculations are in tolerance analysis). (d) Verify that it gives us accurate distances in 8/10 images.

Requirements	Verification
3D Map 1. (5)Store and send the matrix of distances to a computer and generate a 3D point cloud model. 2. (5)Perform a 90 degree sweep using the stepper motor to collect minimum of 18 distance vectors. 3. (10)Should be able to generate a 3D point cloud model from a matrix of distances in at least 3 minutes. 4. (5)Make the 3D point cloud interactive.	Verification for 3D map 1. Verification for 1. (a) Load known data (e.g. distance matrix of a flat plane) onto a computer, run the algorithm (b) Plot the point cloud and verify that the figure is 90% flat. 2. Verification for 2. (a) Because the max step angle for us is 5 degrees, we perform 90/5 steps. (b) Run the distance calculation algorithm for each step and store the vector. (c) Repeat the above step 18 times (max), and verify that final matrix has 18 rows. 3. Verification for 3. (a) Using the same method as above, load sample data/frame from camera. (b) Check the time taken to compute the 3D point cloud. (c) Verify that the processing takes less than 3 mins. 4. Verification for 4. (a) Confirm that one can interact with the point cloud by rotating it and observe form different angle.
Microcontroller-Camera Sync 1. (5)The camera should take pictures after the motor stops moving and then start moving again as soon as the distance has been entered into and array.	Verification for Microcontroller-camera sync 1. Verification for 1. (a) We send Bluetooth signal to the computer. (b) After receiving the signal we run our script on the input data. (c) Once the matrix has been filled, send a signal back to the Bluetooth module. (d) Probe the Module's RXD pin and check for the receive bit. (e) After receiving, we signal the microcontroller to move update the motor driver.

4 Tolerance Analysis

4.1 Critical Component

Our critical component is finding the distance based on image processing and calibration we have done earlier, and hence finding out the min and max values for the algorithm.

4.2 Acceptable Tolerance

We will test it in dark or medium light settings and at different distances as mentioned in the R&V for distance calculation. So we will need to calibrate and find the *ro* and *rpc* values in these settings. After testing with some rough data, we were able to plot a linear regression model and extrapolate information regarding the maximum distance we are able to map for our technique. We found that the maximum value was 400cm and the min was around 15cm.

4.3 Test Procedure

At a distance of 300cm form the camera and laser the pfc value was 49. At 120cm it was it was about 112, at 80 it was 131 etc. The pfc were the y values while the distances were the x. Then using $stats.linregress()$ we obtained the $slope$, $intercept$, r_value, p_value, and std_err. We were then able to extrapolate information form this formula and found that around 401 cm the pfc value is 0.7, meaning its 0.7 pixels form the center of the image on the focal plane. This is not possible, hence at 4m which is our current maximum value it gives us a pfc of around 1.4. Similarly at 12cm the pixel difference is around 0.8 which is again not physically possible, hence 15cm is our current minimum value. These values are likely to change over time as we continue to update our data by testing the calibration procedure with different methods. This will likely result in varying *rpc* and *ro* values.

4.4 Presentation of Results

We will check and compare the results (actual distance vs obtained distance) obtained after the optimal '*ro*' and '*rpc*' calculations have been found, as they would give us the optimal min/max range for mapping and also keep the calculated distance withing 80% of the actual distance.

5 Cost and Schedule

5.1 Cost Analysis

5.1.1 Labor

Name	Hourly Rate	Hours Invested	Total Cost
Soham Waychal	$30	250	$18,750
Vedanth Swain	$30	250	$18,750
Total		500	$37,500

5.1.2 Parts

Part	Part Number	Unit Cost	Quantity	Total
Webcam	Logitech C270	$20.98	1	$20.98
Microcontroller	MK20DX256	$29.98	1	$29.98
Stepper Motor	17HS16-2004S	$12.99	2	$24.98
Stepper Driver	A3967 S	$6.45	2	$12.90
Bluetooth Module	HC-06	$10.05	1	$10.05
Laser	gn-0007	$9.95	1	$9.95
Total				$87.86

5.1.3 Grand Total

Parts	Labor	Grand Total
$87.86	$37,500	$37587.86

5.2 Schedule

Week	**Task - Vedanth**	**Task - Soham**
09/11/16	Project Proposal	Project Proposal
09/18/16	Stepper motor set-up This involves figuring out what pins are needed and how everything is connected up. Finally we will write code that is sent to the motor driver	Camera and app set-up Create an android app that supports camera and OpenCV. Create Python project for webcam.
09/25/16	Laser set-up and distance finding - use the stuff above to calculate distances	
10/02/16	Bluetooth module set-up Connect the HC-06 module to the micro-controller and send and receive signals. Confirm using putty.	Setting up OpenGL Download and install OpenGL and its prerequisites. Learn how to render and how OpenGL deals with displaying objects on screen.
10/09/16	Bluetooth sync with app Create protocol so that when the stepper motor stops rotating, signal is sent to image processing to take picture and find distances. Once that is done, we send signal back to microcontroller to start moving again.	Creating array of distances Once we have calculated distances, we simply add the known distances into a 2D matrix that we can later plot. This will involve attaching the webcam to the stepper motor.
10/16/16	Precise stepper movement Make sure that the stepper motor is moving the given amount with a very low degree of error	Plotting point cloud Once we have the 2D matrix of distances, use it and OpenGL to create a 3D model of the distances.
10/23/16	Debugging and finding correct ranges Find correct red values under various conditions for laser so that we can track it in all environments	Memory Management Find ways to efficiently release captured images while keeping processing time up
10/30/16	Setting up stepper motor and webcam Create a base so that the stepper motor does not move and attach the webcam to the stepper motor so that the motor rotates the webcam without interference	Create 2D matrix of distance points After the webcam and the motor has been attached, iterate through the motors steps, calculate distances, and put them into a 2D array
11/06/16	OpenGL Rendered Use OpenGL to render triangles over set of vertices	Projection Transformation Transform the 3D points that are stored in the matrix to a format that can be presented on the computer screen
11/13/16	Debugging Fix any errors. Check for edge cases. Make sure the app does not crash	Optimization Try to get all the image preprocessing, point cloud transformation, and renderer to work faster.
11/20/16	Thanksgiving Break	
11/27/16	Final Presentation and Demons	
12/04/16	Final Presentation and Demons	

6 Ethics and Safety

6.1 Ethical Issues

Our project follows the IEEE code of ethics as following:

> 1. 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 - We deal with this in safety.
>
> 3. To be honest and realistic in stating claims or estimates based on available data - We will not make up data or exaggerate claims.
>
> 5. To improve the understanding of technology; its appropriate application, and potential consequences - We will actively try to improve the technology we are working on.
>
> 6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations - We will take the necessary training in order to further our project.
>
> 7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others - We will incorporate valuable criticism into our project, try to improve on the design as suggested by other, credit them.
>
> 10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics - We will help other build on similar projects and help them in any way we can.

6.2 Safety

The project uses low power overall. However, we will be using lasers and LiPo batteries that will be constantly charged and discharged. Laser safety is probably the most important factor here since even small amounts of laser light can cause permanent eye injuries. The Occupation Safety and Health Administration (OSHA) has classifications defined for laser safety as follows:

> Class IIIB: moderate power LASERs (cw: 5-500 mW, pulsed: 10 J/cm2 or the diffuse reflection limit, whichever is lower). In general Class IIIB LASERs will not be a fire hazard, nor are they generally capable of producing a hazardous diffuse reflection. Specific controls are recommended.

Our expected laser output is 10mW which will be classified as a CLASS IIIB laser. We will attach appropriate laser safety signs. As this class of lasers is not strong enough to bounce off of shiny or reflective surfaces to create hazardous diffuse reflections, this should not be a severe safety concern. However, objects with very high reflectivity should be removed when operating the laser.
As for the LiPo batteries there are various safety regulations that we will that are but not limited to:

- Do not charge batteries above their maximum safe voltage (+3.7V).
- Should not draw more than maximum current (600mA).
- Do not charge batteries above or below certain temperatures (usually 0-50 degrees C).
- Do not short circuit the battery.
- Do not immerse the battery in water or liquids. Keep or store it in a cool and dry place.

7 Appendix

7.1 Gaussian Filter

The Gaussian function is the normal distribution from statistics. In one dimension it is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

In two dimensions, it is the product of two such functions:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where x and y are distance from origin, σ is standard deviation. Values from this distribution are used to create a convolution matrix which is then applied to the original image. Each pixel value is set to the weighted average of that pixel's neighborhood (that is, if we apply a 5x5 convolution matrix, the pixel at the center will have a value equal to the weighted average after the convolution is done). For example, this is a Gaussian matrix or kernel of size 5 with $\sigma = 1.4$ convolved with the image to give the blurred image.

$$B = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A$$

The final equation looks something like

$$I^{filtered} = \sum_{x_i \in S} G_\sigma(||x_i - x||)$$

7.2 Bilateral Filter

The bilateral filter is similar to the Gaussian filter but instead of just blurring based on the distance from central pixel, the bilateral filter also incorporates color intensity and depth differences. The filter is defined as:

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \in S} I(x_i) G_{\sigma_s}(||x_i - x||) G_{\sigma_r}(||I(x_i) - I(x)||)$$

where the normalization term is

$$W_p = \sum_{x_i \in S} G_{\sigma_s}(||x_i - x||) G_{\sigma_r}(||I(x_i) - I(x)||)$$

In the equations above,

- $I^{filtered}$ and I are filtered and original images and x is the current working coordinate
- $I(x)$ gets the intensity value at that pixel location x
- S is a $n * m$ window centered around x
- G_{σ_s} is the spatial kernel
- G_{σ_s} is the range kernel

Out of the four items listed above, the range kernel is the only item that differs from a Gaussian filter. This kernel calculates smoothing differences in intensities while the spatial kernel smooths based on coordinates.

7.3 Sharpen Filter

The easiest way to sharpen an image is to perform a Gaussian smoothing filter on the original image and to subtract the smoothed image from the original image. Below is the C++ code using OpenCV to perform the smoothing operation.

```
#include "opencv2/opencv.hpp"
int main(int argc, char** argv){
    cv::VideoCapture cap;
    cv::Mat frame;
    cv::Mat image;
    for(;;){
        cap >> frame;
        if(frame.empty()) break;
        cv::GaussianBlur(frame, image, cv::Size(0, 0), 3);
        cv::addWeighted(frame, 1.5, image, -0.5, 0, image);
        if(waitKey(1) == 27) break;
    }
}
```

References

[1] Todd Danko
Webcam Based DIY Laser Rangefinder

[2] Christian E. Portugal-Zambrano, Jesus P. Mena-Chalco,
Robust Range Finder Through a Laser Pointer and a Webcam

[3] Yu-Shin Chou and Jing-Sin Liu,
A Robotic Indoor 3D Mapping System Using a 2D Laser Range Finder Mounted on a Rotating Four-Bar Linkage of a Mobile Platform

[4] Logitech C270 Webcam Specs,
`http://www.logitech.com/en-us/product/hd-webcam-c270`

[5] Stepper Motor Datasheet,
`http://www.omc-stepperonline.com/3d-printer-nema-17-stepper-motor-2a-45ncm64ozin-17hs162004s-html`

[6] Stepper Motor Driver Datasheet,
`https://cdn.sparkfun.com/datasheets/Robotics/A3967-Datasheet.pdf`

[7] Microcontroller Manual,
`https://www.pjrc.com/teensy/K20P64M72SF1RM.pdf`

[8] Microcontroller Datasheet,
`https://www.pjrc.com/teensy/K20P64M72SF1.pdf`

[9] Bluetooth Module Instruction Manual,
`https://www.rcscomponents.kiev.ua/datasheets/hc_hc-05-user-instructions-bluetooth.pdf`

[10] Laser
`https://www.amazon.com/gp/product/B012V3U3KK/`

[11] OpenGL Projection Matrix
`http://www.songho.ca/opengl/gl_projectionmatrix.html`

[12] Canny Edge Detection
`https://en.wikipedia.org/wiki/Canny_edge_detector`

[13] Canny Edge Detection using OpenCV
`http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html`