**Remote Measurement and Monitoring Final Report**

Johns Hopkins University EN 525.743 / New Spin

Wesley Myers

**Abstract**

The following document lays out the documentation for the Remote Measuring and Monitoring tool created for New Spin.

Table of Contents

Project Goal and Description 4

High Level Requirements 4

System Components 7

Raspberry Pi 7

Raspberry Pi Camera Module 7

Laser Range Finder 8

Interface Board 8

Pan-Tilt System 9

Interface Board 9

Interface 10

Raspberry Pi and Arduino 10

Arduino and LRF 10

Web Interface 11

Target Operation 15

Risks and Challenges 16

Integration of Laser 16

Servo Operation 16

Identifying the Laser 16

Webserver 16

Algorithms 17

Calibration with OpenCV 17

Automatic Feature Detection 19

Project Schedule 20

October 13th 20

October 20th 20

October 27th 21

November 3rd 21

November 10th 21

November 17th 21

November 24th 21

December 1st 22

Equipment List 22

Materials Delivery 22

Future Work 23

Notification Saying Recording Has Started 23

Dropbox Interface 23

Object Measurement 23

Automatic Feature Detection Scale-Invariant Feature Transform 25

Assembly Instructions 26

Physical 26

Software 27

Formatting the SD Card 27

Installing the Operating System 27

Turning Off your Raspberry Pi 28

Getting Wifi Setup 28

Update Your Raspberry Pi 29

OpenCV Installation 30

Webserver Installation 30

MySQL Installation 31

Software Installation 32

Performance 33

Measurement of Targets 33

Analog Laser Limitations 34

Pan-Tilt Response Time 34

Over WiFi Network 34

On System 34

Accuracy of LRF 34

Analog Measurement 34

Serial Measurement 35

Terminal measurement 35

Appendix A 36

Arduino Code 36

Arduino LRF Test Serial Code 41

Appendix B 42

Serial Code 42

MySQL Code 44

Website Code 47

clean.php 47

targeting.php 50

stream.php 52

configuration.php 54

cmd\_func.php 56

script\_min.js 57

styles.css 61

OpenCV Code 62

Targeting Calibration 62

SIFT Algorithm 63

# Project Goal and Description

The goal of this project is to enable users to remotely measure noticeable change using a laser range finder. This fulfills requirements from New Spin[[1]](#footnote-1) customers who want the ability to remotely measure distance to track changes at electric substations that cannot otherwise be accomplished using image differentiation. In addition, the system could be used for absolute measurement for data to go into New Spin’s You Control[[2]](#footnote-2) system.



Figure - Example Electric Substation

As shown in Figure 1, electric substations are often remote and thereby vulnerable to nature or thieves. Thus having an automated system that can measure distances to targets provides the customer with an additional degree and the confidence that their assets are safe.

This project is being done in conjunction with a class[[3]](#footnote-3) at the Johns Hopkins University. As such, the requirements were set to meet a semester timeline.

## High Level Requirements

The system must have a web user interface to control it. The user must have the option to manually select targets for automatic measurement. Once the targets are selected, the system should then be able to measure those points. This system can be hard wired via Ethernet, or be deployed wirelessly for command and control.

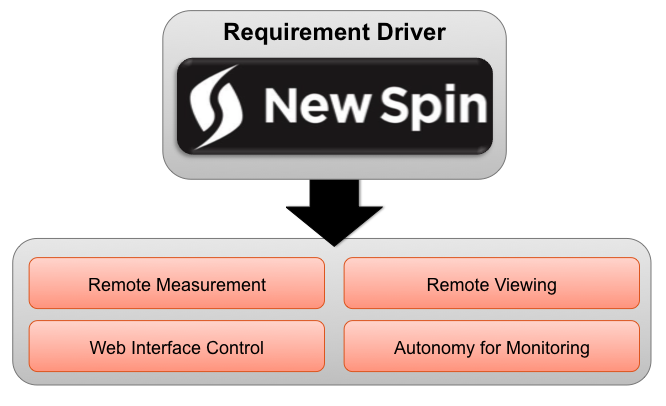


Figure - High Level Requirements

The final result is the system shown below. This system has a pan-tilt base that allows the user to aim the system. On the bottom platform are the Raspberry Pi and Arduino UNO. On the top platform are the LRF and camera.



Figure - New Spin Remote Measurement System

In order to fulfill this requirement, an embedded system was created to allow remote viewing and sensing. As a prototype, a Raspberry Pi and Arduino UNO was used for rapid development. This will support a camera, servos, and a range finder, which are all necessary for this project.

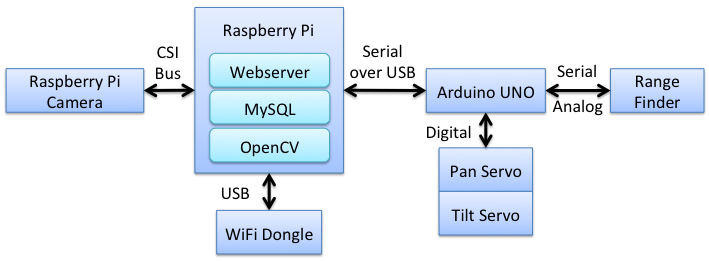


Figure - System Architecture

The system itself has some limitations due to the pan-tilt servo design of the system. This is well documented later in this paper, but essentially current draw from the servos causes poor LRF readings and it limits the accuracy of the aiming of the system. This means that at certain angles, the system will not return good results. At long ranges, the system may not be able to point at small targets. The servo strain can be attributed to the overall weight of the system. Using wood and metal in the frame caused the system to weigh about 2 pounds.

# System Components

## Raspberry Pi

This is a credit-card sized computer with peripherals that users are used to on a typical computer. This board allows for development with more horsepower than an Arduino. In this case, image processing was done utilizing OpenCV to automatically locate targets of interest. Further discussion into the OpenCV algorithms can be found in later sections.

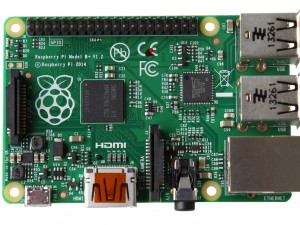


Figure - Raspberry Pi Model B+

## Raspberry Pi Camera Module

This camera is specifically designed to interface with the Raspberry Pi’s CSI bus. There are two cameras to choose from: standard RGB and Infrared. The IR camera was chosen since it was able to see the LRF’s laser.

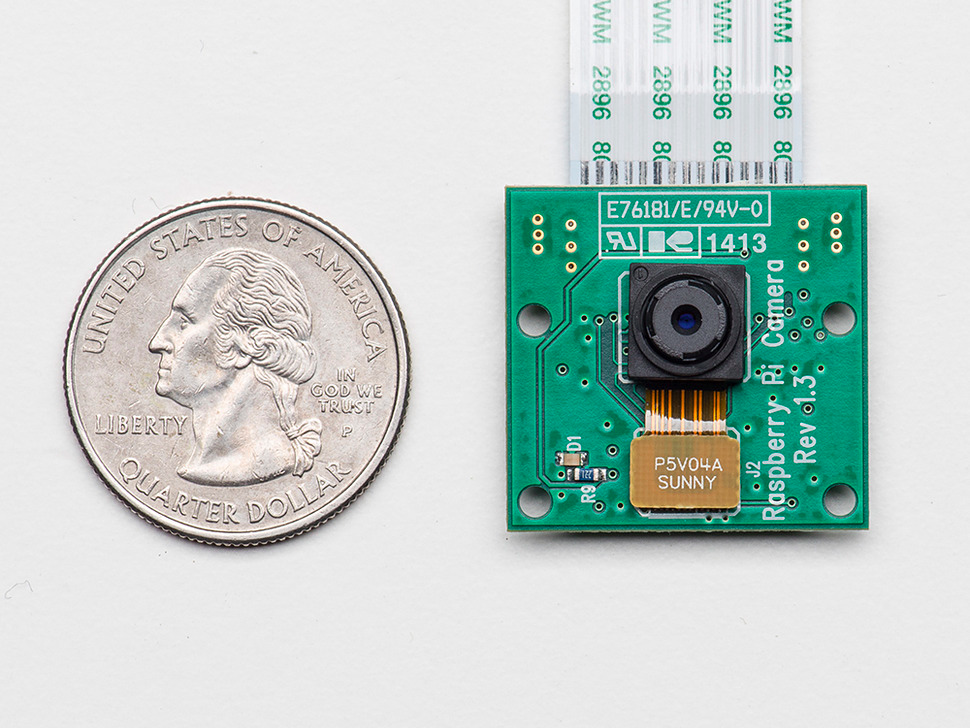


Figure - Camera Next to Quarter

## Laser Range Finder

The SF01 Laser Range Finder[[4]](#footnote-4) is a platform designed by LightWare Optoelectronics. It is a relatively low cost laser range finder that can measure targets at up to 60 meters with 1-centimeter accuracy. It also comes in a small package that is ideal for the system. For commercial grade range finders, this is the only one within the required specifications that is able to interface with a Raspberry Pi or Arduino platform.



Figure - SF01/A LRF

### Interface Board

On the SF01/A LRF, there is an interface board for serial communication. The pins of interest to us are the UART pins and the Vout pin. With UART, we can serially communicate with the LRF at a baud rate of 19200. That baud rate is actually not documented in the ICD for the LRF. Serial communication is documented later in this report. The Vout pin actually gives us a voltage of the laser’s measurement. This can be used for rough estimation of distance. Arduino code to interact with the LRF can be found in Appendix B under Arduino Code.

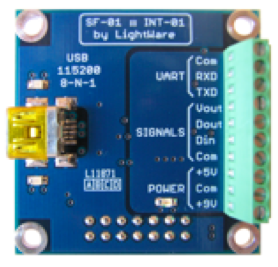


Figure - SF01 Interface

## Pan-Tilt System

To save time on design of this system, a pre-fabricated pan-tilt system was chosen. Servo City makes an assortment of pan-tilt platforms to choose from. The SPT200[[5]](#footnote-5) model in particular can hold up to 2-pounds, which should be adequate for the system’s purposes. To operate this system, two HS-5685MH[[6]](#footnote-6) servos will be used as recommended in the SPT200 description.



Figure - SPT200 Pan-Tilt Platform

## Interface Board

The interface board is constructed so that it connects the LRF, Arduino, pan-tilt, and Raspberry Pi for power and I/O. The interface board does not have an interface for the serial connection between the Raspberry Pi and Arduino. A USB cable between the two is being used instead. The board also has a barrel jack for power to be supplied to the Servos. This was to help prevent voltage drops in the system, which can cause poor measurements and accidental system resets.

## 

Figure - Bottom View of Interface Board

## 

Figure - Top View of Interface Board

## Interface

The following sections document communication between the major devices in play.

### Raspberry Pi and Arduino

The BAUD rate between the Raspberry Pi and Arduino is 9600.

|  |  |
| --- | --- |
| Command | Expected Response |
| ‘p <int>’ | Pan to <int>, return int of new pan position. |
| ‘p0’ | Return int of current pan position. |
| ‘p1’ | Pan Left by 1, return int of new pan position. |
| ‘p2’ | Pan Right by 1, return int of new pan position. |
| ‘t<int>’ | Tilt to <int>, return int of new tilt position. |
| ‘t0’ | Return int of current tilt position. |
| ‘t1’ | Tilt Down by 1, return int of new tilt position. |
| ‘t2’ | Tilt Up by 1, return int of new tilt position. |
| ‘r’ | Read the LRF, return float of distance in meters. |

### Arduino and LRF

The BAUD rate between the Arduino and LRF is 19200. This is not documented in the User Manual for the LRF.

|  |  |
| --- | --- |
| Command | Expected Response |
| ‘d’ | Distance in meters, return as a float. |
| ‘s’ | Signal strength as a percentage (int) |

# Web Interface

The following list is the original requirements as set forth at the beginning of the project. Of all of these, the only one that was not achieved was the autonomous target selection, which was removed as a requirement by New Spin. The website design/layout uses Bootstrap[[7]](#footnote-7) for rapid development.

* Tilting the system
* Panning the system
* Viewing the video feed
* Active measurement
* Autonomous target selection options
* Display measurement results

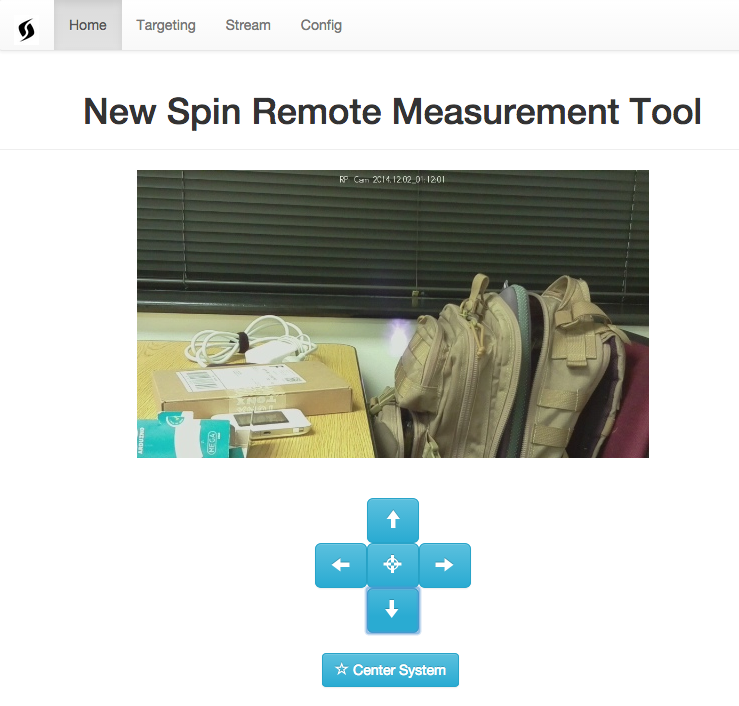


Figure - Home Page

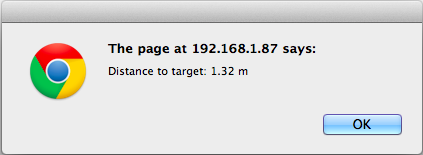


Figure - Laser Read Response

Target selection is an important part of this project. The user needs the ability to select targets and create a database for which the system can periodically check on. The targeting page allows the user to get a list of current targets in the database, clear out targets, and add a target (with a user specified name) to the database. Most importantly, the last option is to verify the targets currently in the database.

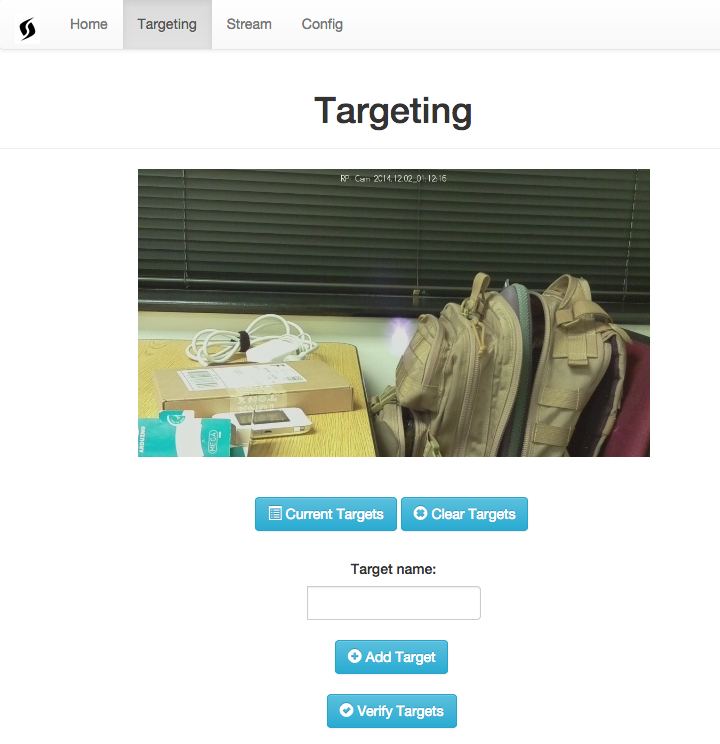


Figure - Targeting Page

The following is a list of pop-up responses from the Targeting page.

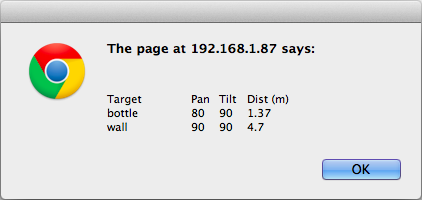


Figure - Current Target List

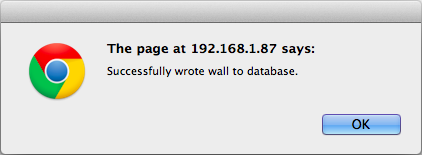


Figure - Adding Target to Database

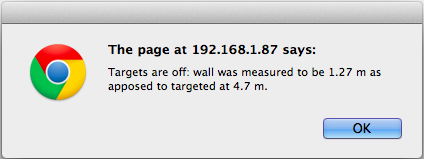


Figure - Bad Target(s) Response

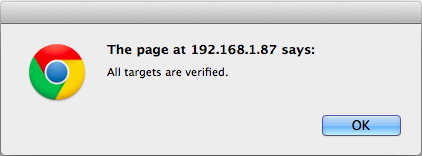


Figure - Good Targets Response

The streaming page cleanly displays the video stream to the user. This can be used solely for monitoring purposes.

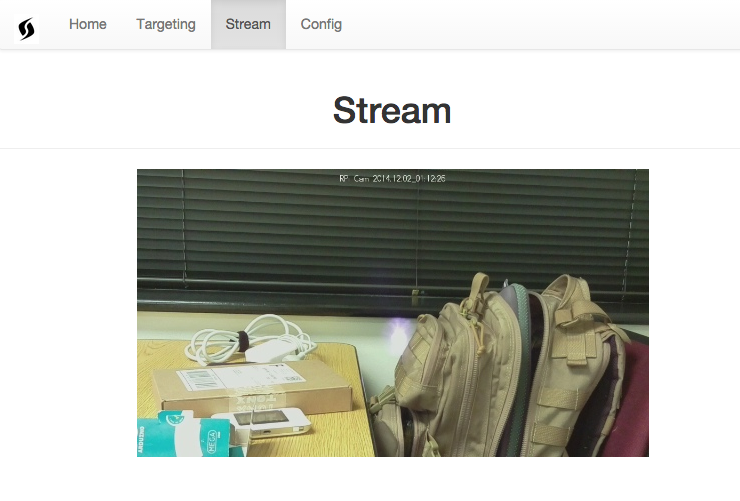


Figure - Stream Page

The configuration page allows the user to reboot and shutdown the system. The user will not always be in close access, so this is a much-needed capability. Note that shutting down the system will cause manual intervention to start it back up.

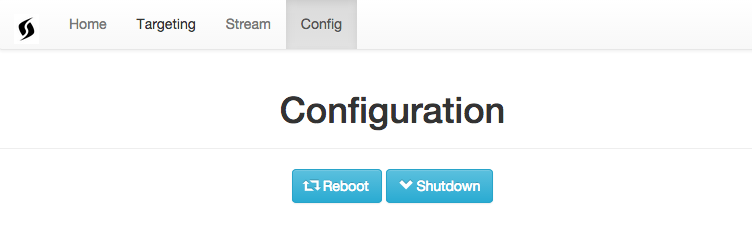


Figure - Configuration Page

# Target Operation

The algorithm for scanning over the targets is as follows. The system essentially reads in the database, which has the pan-tilt position, distance, and target name. The system will point to the pan-tilt position, read the laser, compare the distance to the distance in the database, and then decide if the target isn’t right. If it isn’t right, then the target name and distance offset is added to a string. The return string is returned as a pop up to the user, as shown in Figure 17.

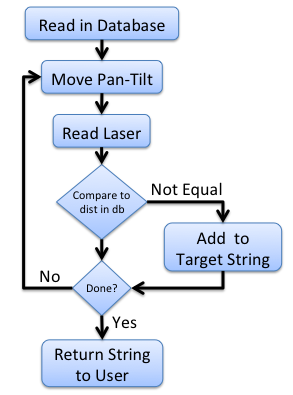


Figure - Verify Targets

In Manual Mode, the user will add targets that are of interest to them. As the user measures the distance to create a profile of the target, the information is saved for later comparison.

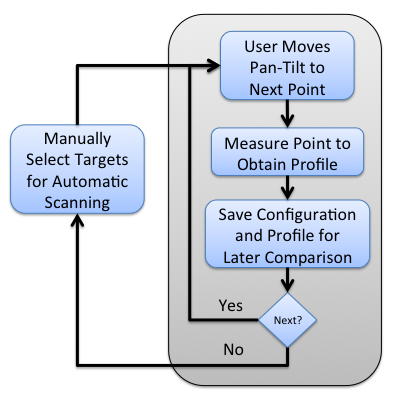


Figure - Manual Point Selection

# Risks and Challenges

## Integration of Laser

Though LightWare might advertise the SF01 as a sensor that is interface-able with Raspberry Pis and Adruinos, there is no documentation of anyone attempting to do so with a Raspberry Pi. The interface protocols for the LRF are well documented, which should’ve allow initial guidance with the laser integration. LightWare does not provide any source code for interfacing with a Raspberry Pi, however there is code[[8]](#footnote-8) for an Arduino. Thus the Arduino was added to help act as a go-between for the LRF and Raspberry Pi.

The Arduino Library did not however work for the LRF out of the box. After much research, it was determined this was due to a faulty baud rate that is undocumented in the ICD. Until the baud rate error was discovered, the analog output was used to determine distance. This proved to be accurate for the system’s classroom purposes.

One issue that has yet to be addressed is that when the servos are active and trying to support the system, it causes the laser to give bad readings. This is mainly due to the weight of the system causing stress on the servos.

## Servo Operation

Operating the servos was the first roadblock encountered. It was determined that the Raspberry Pi was not designed to operate the servos appropriately. The methods to do so was through bit-banging the PWM signal out of the GPIO pins. As this would require separate threads to run the servos, an Arduino was chosen to support the Raspberry Pi. The Raspberry Pi would send serial commands to the Arduino, which would in turn operate the servos.

## Identifying the Laser

One of the challenges was identifying the laser. Using the standard Raspberry Pi camera, this was not possible. The IR filter on the camera did indeed filter out the IR beam from the LRF. To detect the laser, the Raspberry Pi NoIR camera was chosen to support the system. As shown in later sections, we are able to visually identify the laser. In addition, we can use OpenCV to detect the laser’s position relative to the camera’s field of view. This is documented in the next section.

## Webserver

One of the risks up front was the design of the webserver for the system. This challenge was mitigated through a webserver package that can be found on the Installation section of this document.

# Algorithms

## Calibration with OpenCV

When the camera and LRF are pointed in the same direction, it is critical that the system is calibrated to verify they are aimed co-linearly. OpenCV allows us to verify what the camera sees and can tell the user if they need to tweak the system.

As is typical with many image-processing techniques we want as few pixels as possible and ideally have something binary to work with. So the basic strategy is to reduce the image size by looking in the region we care about (the center). Next threshold the image using the basic method of saying the laser is going to be bright. Next we need to detect the circle. There are a couple of techniques, but the common one used was using a Hough Transform[[9]](#footnote-9), which OpenCV provides[[10]](#footnote-10).



Figure - Original Image (1944x2529)

Image dimensions are 1944x2592. This is a lot of area to cover, so it’ll be easier to work with something a bit smaller. Masking over to just work with a 300x300 image will be much faster and easier.



Figure - Cropped Image (300x300)

Thresholding is a bit trickier. We want to only get the bright spot in the image. This is accomplished by using the thresholding[[11]](#footnote-11) libraries in OpenCV.

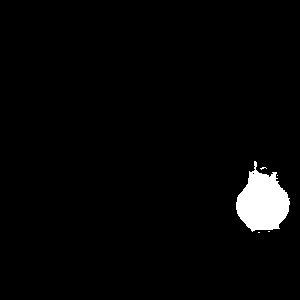


Figure – Threshold Image

The end result is a circle labeling the target. We now have a centroid for the laser in relation for the camera.



Figure – Blob/Centroid Detection

Given that we have a centroid, we can tell the user/operator how off center the laser is in relation to the field of view.

===== Calibration Program Results =====

Blob Information:

x1 = 269.5

y1 = 208.5

[[[ 269.5 208.5 23.37734032]]]

X distance off by: 119.5 pixels

Y distance off by: 58.5 pixels

Figure - OpenCV Calibration Output

## Automatic Feature Detection

After talking to New Spin leadership, they decided that they don’t think automatic feature detection is practical, nor useful. They felt that the user should go ahead and be the one who selects the points to operate the system. Though this was not done in this course, it is documented later for possible future work.

# Project Schedule

To meet class deadlines, the following timeline has been generated. On the following subsections, the executive summary from the Weekly Status Updates have been included.

|  |  |  |
| --- | --- | --- |
| Date | Milestone | Met |
| September 29th | Ordering and Receiving Equipment | Yes |
| October 6th | Pan Tilt Mechanism / CDR Due / TDY Tuesday – Friday | Yes |
| October 13th | Camera and Laser Integration | Yes |
| October 20th | Continue Integration / TDY Sunday - Wednesday | Yes |
| October 27th | OpenCV Development | Yes |
| November 3rd | UI and Webserver Design | Yes |
| November 10th | Algorithm Development / TDY Thursday - Sunday | Yes |
| November 17th | Algorithm Development / Vacation | Yes |
| November 24th | Testing and Analysis | Yes |
| December 1st | Testing and Analysis | Yes |
| December 8th | Project Documentation Due | Yes |
| December 13th | Final Presentation | n/a |

## October 13th

* Architecture Design Change
  + Addition of Arduino UNO
* Pan-Tilt Assembly and Testing
  + Servo range road block
  + Possible custom Servo Library for PWM
* Step 1 of Camera-Laser Integration
  + Analog output not good enough (Arduino ADC)
  + Serial communication roadblock
* Installation of Raspberry Pi

## October 20th

* Final Assembly of System
  + Added brackets and servo mounting plate
* Laser-Camera Integration Behind
  + Dead camera
  + Longer cable required
* Interface Board Development
  + Board layout drafted
  + Soldering to complete

## October 27th

* Serial Connection Work
  + Arduino Receive/Transmit Working
  + Raspberry Pi Receive/Transmit Working
  + Arduino to Laser Not Working
* Camera Integration and Testing
  + Laser Visible to Long Distance
  + OpenCV Detection of Laser

## November 3rd

* Interface Board
  + Narrowed Bug from Last Week
  + Board Revision for Power Issues
* Website Development
  + Basic Website Stood Up
* Arduino Serial Code Development
  + Reads Analog Pin and Sends to Pi
  + No luck on serial communication to LRF

## November 10th

* Interface Board
  + Completed
* Website Development
  + Arduino reset on Serial Begin solved
  + Buttons not working yet
* Serial Development
  + Minor tweaking done on Arduino end

## November 17th

* Website Development
  + Button clicks work
  + Alert pop up for laser distance
  + Center System
* Serial Interface
  + Updated Python Script
* Physical Infrastructure
  + More secure laser mount

## November 24th

Website Development

* MySQL database added
* Additional Serial Needs

## December 1st

* Website Development
  + New webpage design and layout
  + Error Checking
* Serial Communication
  + Bad Analog Reads
  + SoftwareSerial causing jitter
  + Arduino Mega getting strange serial reads
  + Good output on LightWare terminal

# Equipment List

The following are core items required to complete this project.

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost per |
| LRF | 1 | $500.00 |
| Raspberry Pi | 1 | $40.00 |
| Raspberry Pi Camera | 1 | $30.00 |
| Pan-Tilt System | 1 | $60.00 |
| Servos | 2 | $40.00 |
| Wireless Dongle | 1 | $12.00 |

## Materials Delivery

Most of the material can be ordered off of Amazon[[12]](#footnote-12). This includes the Raspberry Pi, camera module, and wireless dongle. These were delivered by October 3rd as they are all contained within the Amazon Warehouse.

The Laser Range Finder is located in South Africa, but arrived by October 3rd. This is by far the most expensive component of this project. In addition, if it breaks, then replacing it will take multiple weeks. The company has these lasers on back-order because they are in such high demand.

The Pan-Tilt system and servos can be ordered from Servo City[[13]](#footnote-13). These items ship from Kansas City and arrived by October 3rd.

# Future Work

Many of the requirements described in this document seem to revolve around security camera applications. A good security camera to model after is the Dropcam[[14]](#footnote-14). This system in particular has many good features that could be applied here. In this case we can have the system email/text the owner if there is a change. Attached would be an image showing the detected incident. In addition, if videoing capabilities are incorporated, then automatic downloading of recordings of incidents can be hooked up to a Dropbox[[15]](#footnote-15) account. In this instance, Dropcam wants the user to pay for storage, while this system would upload to a free account.

## Notification Saying Recording Has Started

Send an email or text if a target appears to be missing. An image should be included as an attachment so that the user can verify if in fact the target is missing. The user should then be able to provide feedback

## Dropbox Interface

As scans are periodically done, upload images to a specified Dropbox account. This enables the user to check past history, as well as verify that the system is detecting changes appropriately.

## Object Measurement

The goal is to measure the distance to various targets. The simplest method would be to simply measure a target using multiple samples and then run the information through a low pass filter. Currently, that is what the system does because of the pan-tilt servos. Once the system goes to using stepper motors, we can use the following described measurement technique.

As explained in the Risks and Challenges section, it is possible that we could miss this target if the system is off by a little bit. In Figure 28, we have a potentially scan-able area for the laser. In this case, we have four corners labeled as Top Left (TL), Top Right (TR), Bottom Left (BL), and Bottom Right (BR). As the laser moves 1 degree or position from corner-to-corner, it could do multiple scans per movement. So for example, between TL and TR, it could take multiple samples to cover an object in the middle of the scan.

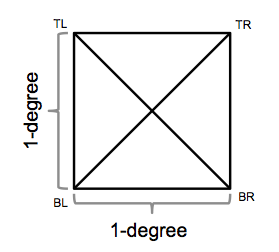


Figure - Range of Measurement

Figure 29’s gray line is an example of the power line. In this case, the line is a bit smaller. The red lines represent the laser’s movement across the wire.

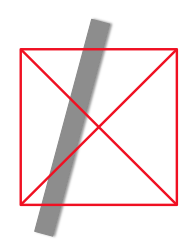


Figure - Simulated Measurement of Power Line

Figure 17 illustrates the results of the scan lines. In this case, we can see where the power line is detected as shown by the grey blocks. Note that in this case, we’re assuming continuous sampling while in implementation, we will have a few discrete samples to evaluate from.

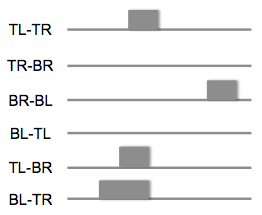


Figure - Results of Power Line Measurement

With this information, we can develop a profile of a specified target to remember what it looks like. Alternatively on a scan, we can figure out how far a target is away and provide the user with a probabilistic answer.

## Automatic Feature Detection Scale-Invariant Feature Transform

SIFT[[16]](#footnote-16) is an algorithm that can be used to detect and describe local features in an image. In order to keep the explanation light, this library allows us to make a call to find significant features within the image. Work was done on this, however New Spin decided to axe the requirement. The code for this can be found in Appendix B under OpenCV Code.

The Scale-Invariant aspect of this algorithm means that the results are invariant to object translation, scaling, and rotation. Meaning that we should have a similar set up results when we apply this algorithm to various substations. This algorithm should reference a database of images, allowing it to select better features on the specified image.

Figure 31 illustrates a possible perspective for the system, as well as the output of the SIFT algorithm. In this case, there are many possible features to choose from that might be of interest to the customer. What we get back from SIFT are a lot of features. In this figure, the features are not filtered. What we see are all of the possible features to choose from. As the algorithm has no perception of free space, one will notice that many centroids of the power structure are actually selected as features. Using this algorithm, we can filter out the insignificant features and offer to the user only the significant features that might be of interest.

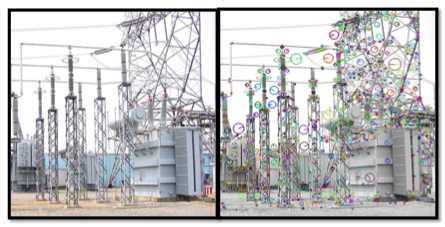


Figure - SIFT Algorithm Output

# Assembly Instructions

## Physical

The physical assembly is straightforward. The Arduino and Raspberry Pi can be mounted straight onto the wood via ¼-inch standoffs with wood screws. The LRF needs some padding to support it on the wood platform. Hanger straps can be used to support the laser on the platform. It is recommended that the foam pad be added to the hanger strap that supports the LRF PCB.

The wood is 7x2.5x.75 inches. The Mounting plates are 3.5x1.375 inches. The hanger straps are .75 inches wide. These can all be found at a hardware store.

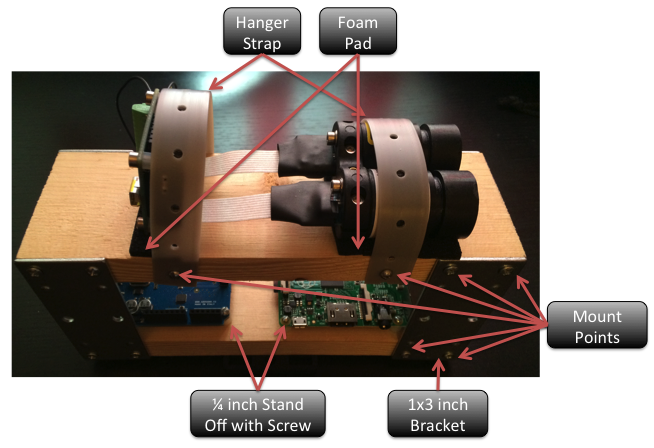


Figure - Assembly Diagram for Frame



Figure - Mounting Plate for Pan-Tilt Servo

# Software

Setting up the Raspberry Pi is a rather time intensive process. This required installation of the default operating system for the Raspberry Pi, Raspian. This is a version of Linux Debian that is made for the Raspberry Pi. The following is documentation on how the Pi was set up.

### Formatting the SD Card

First things first is getting your SD Card all set up for the Raspberry Pi. Below are three sets of instructions given whether you have a Windows, Mac, or Linux machine.

#### Windows

1. Download the SD Association’s Formatting Tool from <https://www.sdcard.org/downloads/formatter_4/eula_windows/>
2. Install and run the Formatting Tool on your machine
3. Set “FORMAT SIZE Adjustment” option to “ON” in the “Options” menu
4. Check that the SD card you inserted matches the one selected by the Tool
5. Click the “Format” button

#### Mac

1. Download the SD Association’s Formatting Tool from <https://www.sdcard.org/downloads/formatter_4/eula_mac/>
2. Install and run the Formatting Tool on your machine
3. Select “Overwrite Format”
4. Check that the SD card you inserted matches the one selected by the Tool
5. Click the “Format” button

#### Linux

1. Use gparted (or the command line version parted)
2. Format the entire disk as FAT

### Installing the Operating System

After your SD Card is formatted, extract the files from NOOBS[[17]](#footnote-17) and drop it on the SD Card. Then plug the card into your Raspberry Pi. Make sure you have your keyboard wireless dongle plugged in. Now turn it on! Give it a few seconds and your Pi should show the following.

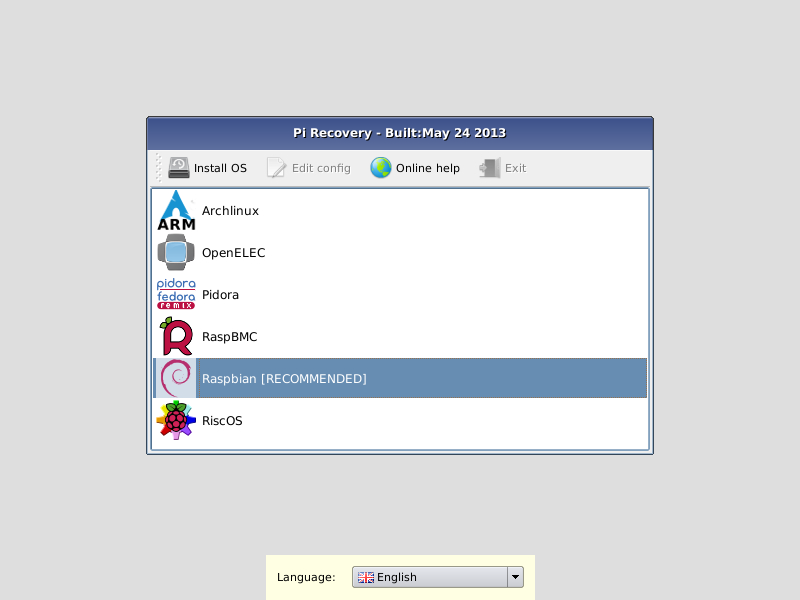


Figure - Raspbian Installation

Choose Raspbian as your OS and it will proceed to install the Raspbian OS. This might take a while. If it takes more than an hour, then you probably have a bad SD Card.

Once this is all done, Raspberry Pi configuration will come up. Just skip to Finish at the bottom of the window.

### Turning Off your Raspberry Pi

DO NOT JUST UNPLUG THE POWER. This is very bad and you won’t do that to your own computer. Run the below shutdown command to properly shutdown the Pi and then remove the power. Not doing so could damage the file system and might be un-repairable!

sudo shutdown –h now

### Getting Wifi Setup

First make sure your Wifi dongle is plugged in. From the terminal, type in the below command

This should open up an editor that should show something similar to what is below. Edit this as necessary to match the below text.

sudo nano /etc/network/interfaces

auto lo

iface lo inet loopback

iface eth0 inet dhcp

allow-hotplug wlan0

auto wlan0

iface wlan0 inet dhcp

wpa-ssid "ssid"

wpa-psk "password"

Now reboot your Pi by typing the below command. You need to do this so that you can install packages.

sudo reboot

As your Pi is coming back up, you should be able to see some DHCP requests attempting to connect to the Wifi. If it can’t connect, then you should be able to see it attempt a couple of times and then say it couldn’t connect. If this is the case, then double check what was typed into that file. You can also set up the Pi to accept multiple networks[[18]](#footnote-18).

### Update Your Raspberry Pi

Type in the below commands to update your Raspberry Pi’s firmware. You will need Internet access to do this.

sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

sudo apt-get dist-upgrade

## OpenCV Installation

Install OpenCV by running the following command.

sudo apt-get install libopencv-dev python-opencv

## Webserver Installation

The package is called Raspberry Pi Webcam Interface[[19]](#footnote-19). All of the code can be sourced off of the referenced github account[[20]](#footnote-20). Download the software off of the github account and *scp* them over to */home/pi*. After you unzip them on the Raspberry Pi, then it is straightforward to install.

cd /home/pi/

unzip RPi\_Cam\_Web\_Interface.zip

cd RPi\_Cam\_Web\_Interface

./RPi\_Cam\_Web\_Interface\_Installer.sh install

./RPi\_Cam\_Web\_Interface.sh start

Now you should be up and running! Go to the website and you should be able to see an interface with many options.

192.168.1.110/index.php

All of your files are located in */var/www*. The files in that directory were copied over when the install was done. If your IP address is different, then you may want to do an *ifconfig* and double check.

In order to allow the website to execute commands on the Raspberry Pi, we need to allow access for the user of those files.

/etc/sudoers.d/RPI\_Cam\_Web\_Interface

All we need to do is add one line to the file. Note that this is very dangerous and should only be done for development purposes. Instead, the sudoers file should have a list of commands that are allowable for this system.

www-data ALL=(ALL) NOPASSWD: ALL

In the process of doing this, a very handy trick[[21]](#footnote-21) was learned. If you ever screw up the sudeoers file, just run the following command

pkexec visudo -f /etc/sudoers.d/filename

## MySQL Installation

Installing MySQL is fairly straightforward. Just execute the following.

sudo apt-get update

sudo apt-get install mysql-server

In order for MySQL to run more efficiently on the Raspberry Pi, we need to edit the configurations a bit from the standard distribution.

sudo mv /etc/mysql/my.cnf /etc/mysql/my.cnf.bk

sudo cp /usr/share/doc/mysql-server-5.5/examples/my-small.cnf /etc/mysql/my.cnf

Now in my.cnf, change the file to reflect

query\_cache\_size = 8M

And then restart mysql

sudo service mysql restart

Since most of the work already done uses python, I’ll be using python scripts to interact with the database. Apparently python only comes with MySQL drivers for SQLite, so thus drivers must be installed for python to interface with MySQL.

sudo apt-get install python-mysqldb

The first thing that needs to be done is create a new database. Change your working directory to */var/www*. Log into the system by executing the following.

mysql –-user=”root” –-password=”system”

You should now be inside of MySQL and should have a command prompt. Type in the following command

CREATE DATABASE nslaser\_db;

You should now have a new database to work with. Exit out of MySQL by typing “quit” into the prompt. Now we need to add tables. So go ahead and log back into MySQL, but include the database name.

mysql –-user=”root” –-password=”system” --database="nslaser\_db"

Now create 2 tables for the system to interface with. The first is a target list. This will store our target information, including the name. The second is the system information. This will enable us to keep track of the system’s pan-tilt position, plus the last laser read.

CREATE TABLE targets (name varchar(30), pan int, tilt int, distance float);

CREATE TABLE system\_info (pan int, tilt int, distance float);

## Software Installation

All code can be found in the appendices at the end of this document. All code will be eventually pushed to New Spin’s github[[22]](#footnote-22) account, but may not be publicly accessible.

All executable files, such as the serial commands, should have the owner:group set to www-data:www-data. In addition, I set the files to be wide open on permissions.

Appendix A, Arduino Code, Arduino code should be installed on the Arduino UNO.

Appendix B, Website Code, should be installed on the Raspberry Pi, under */var/www*.

Appendix B, Serial Code, should be installed on the Raspberry Pi, under */var/www/serial*.

Appendix B, MySQL Code, should be installed on the Rasbperry Pi, under */var/www*.

# Performance

## Measurement of Targets

The further the distance for measurement, the more precision in movement is needed for proper measurement. As the distance of the laser increases, the more “steps” will be required if the system uses stepper motors. This problem will persist as the system’s LRF is upgraded.

The use case example here is where a user wants to measure the distance to the power line. In this case, the power line is 2 centimeters thick and is 50 meters away. The following diagram illustrates this use case.



Figure - Case Study Example

Given that we have a .23-degree window to hit our target, we need a stepper motor that has about 3600 steps unless we want gearing to be applied to the system. That will be the only way to accurately point the laser, assuming it is lined up properly.

In testing, we saw a 1-degree change at 13 feet resulted in a ~2.875-inch deviation.



Figure - Measurement Moving Laser Left by 1-Degree

This means even at short distances, it is be hard to hit the target. Given the current system setup, there is no way to mitigate this.

### Analog Laser Limitations

The laser outputs a voltage between 0 and 3.3 V that represents the distance measured. So for our 5 V analog port, we’re already limiting ourselves a bit.

For our ADC, the Arduino UNO has a 10-bit ADC. So in this case we have 1024 different possible values.

So that means using purely the analog pin, we’re getting terrible resolution. This means that serial communication with the laser is the way to go.

## Pan-Tilt Response Time

The following test was performed over 7 days, taking 10 samples per day.

### Over WiFi Network

When communicating to the system over WiFi, the response time is dependent on the WiFi network. The response time was 3.41 seconds on average. The longest response time recorded was 12.61 seconds. Having a good network greatly affects the performance of the system.

### On System

When directly on the system, the response time is much faster. Testing over the same series of days, the average recorded time for a response was 2.13 seconds. This means that there is about 1.28 seconds of latency over the network.

## Accuracy of LRF

With the LRF, we have two ways to measure it in the system. We can do it over serial, which isn’t used in the final system due to limitations of the software, or over the analog output. In addition, we can run performance testing using LightWare’s terminal software.

### Analog Measurement

The accuracy of the LRF is greatly dependent on the servos at the moment. The current draw from the servos causes the LRF to give out bad readings. So if you can hear that the servos are straining, then the LRF will output bad data unfortunately. The following is a 10-sample set from the laser pointed at position 85, which is pointed down slightly. The average response time is 5.7 seconds. This test was conducted by hitting the measurement button on the home page.

1. 2.85 m
2. 1.27 m
3. 4.95 m
4. 4.46 m
5. 12.59 m
6. 4.46 m
7. 1.37 m
8. 7.30 m
9. 8.87 m
10. 4.36 m

The above list was given to illustrate how drastically the output changes. In the system, the average is running over 20 values. The update speed of the laser is .125 seconds, but a wait of .15 seconds is used. That means 3 seconds is being lost due to filtering.

### Serial Measurement

Though communicating over serial was unable to be done due to the limitations of SoftwareSerial[[23]](#footnote-23), testing was still done. The results were comparable to what was found in the last section. This confirms that the servos under strain cause poor measurements from the LRF.

### Terminal Measurement

The measurement from LightWare’s Terminal application was accurate. The measurements were up to specifications as listed in the LightWare SF01 User Manual.

# Appendix A

## Arduino Code

#include <Servo.h>

#include <SoftwareSerial.h>

//#define DEBUG

//tilt

#define TILT\_LEVEL 90

#define TILT\_MAX\_DOWN 75

#define UP 1

#define DOWN 2

//pan

#define PAN\_MIDDLE 90

#define PAN\_MAX\_LEFT 50

#define PAN\_MAX\_RIGHT 130

#define LEFT 1

#define RIGHT 2

// laser defines

#define SF01\_ANALOG 1

#define SF01\_0\_0V\_DISTANCE 0.0

#define SF01\_3\_3V\_DISTANCE 33.00

int analog;

float analog\_voltage;

float analog\_distance\_meters;

float slope = (SF01\_3\_3V\_DISTANCE - SF01\_0\_0V\_DISTANCE)/3.3;

//SoftwareSerial sf01\_serial(9, 10);

Servo panServo; // create servo object to control a servo

Servo tiltServo;

int pan\_pos; // variable to store the servo position

int tilt\_pos;

int new\_pos;

int val;

char command;

int i;

void setup()

{

panServo.attach(5); // attaches the servo on pin 9 to the servo object

panServo.write(PAN\_MIDDLE);

tiltServo.attach(6);

tiltServo.write(TILT\_LEVEL);

pan\_pos = panServo.read();

tilt\_pos = tiltServo.read();

Serial.begin(9600);

//sf01\_serial.begin(115200);

}

void loop()

{

if (Serial.available())

{

//get command (i.e. read laser, move pan-tilt)

command = Serial.read();

//if there is a number trailing, then get that too

new\_pos = Serial.parseInt();

if(command == 'p')

{

// we want to move the pan servo

if((new\_pos >= PAN\_MAX\_LEFT && new\_pos <= PAN\_MAX\_RIGHT) ||

(new\_pos == LEFT && (pan\_pos - 1) >= PAN\_MAX\_LEFT) ||

(new\_pos == RIGHT && (pan\_pos + 1) <= PAN\_MAX\_RIGHT) ||

(new\_pos == 0) ) //return current position

{

#ifdef DEBUG

Serial.print("Moving Pan Servo: ");

Serial.print(new\_pos);

Serial.print(" -> ");

Serial.println(pan\_pos);

#endif

if (new\_pos > pan\_pos || new\_pos == RIGHT)

{

if(new\_pos == RIGHT)

new\_pos = pan\_pos + 1;

while (pan\_pos < new\_pos)

{

pan\_pos++;

panServo.write(pan\_pos);

delay(50);

#ifdef DEBUG

Serial.println("right!");

#endif

}

}

else if((new\_pos < pan\_pos && new\_pos != 0) || new\_pos == LEFT)

{

if(new\_pos == LEFT)

new\_pos = pan\_pos - 1;

while (new\_pos < pan\_pos)

{

pan\_pos--;

panServo.write(pan\_pos);

delay(50);

#ifdef DEBUG

Serial.println("left!");

#endif

}

}

new\_pos = panServo.read();

Serial.flush();

Serial.println(new\_pos);

}

else

{

#ifdef DEBUG

Serial.println("===== Bad Pan Servo input =====");

Serial.print(" - Servo: ");

Serial.println(command);

Serial.print(" - Position: ");

Serial.println(new\_pos);

#endif

}

}

else if(command == 't')

{

// We want to move the tilt servo

if((new\_pos >= TILT\_MAX\_DOWN && new\_pos <= TILT\_LEVEL) ||

(new\_pos == UP && (tilt\_pos + 1) <= TILT\_LEVEL) ||

(new\_pos == DOWN && (tilt\_pos - 1) >= TILT\_MAX\_DOWN) ||

(new\_pos == 0) ) //return current position

{

if (new\_pos > tilt\_pos || new\_pos == UP)

{

if(new\_pos == UP)

new\_pos = tilt\_pos + 1;

while (tilt\_pos < new\_pos)

{

tilt\_pos++;

tiltServo.write(tilt\_pos);

delay(50);

#ifdef DEBUG

Serial.println("up!");

#endif

}

}

else if((new\_pos < tilt\_pos && new\_pos != 0) || new\_pos == DOWN)

{

if(new\_pos == DOWN)

new\_pos = tilt\_pos - 1;

while (new\_pos < tilt\_pos)

{

tilt\_pos--;

tiltServo.write(tilt\_pos);

delay(50);

#ifdef DEBUG

Serial.println("down!");

#endif

}

}

new\_pos = tiltServo.read();

Serial.flush();

Serial.println(new\_pos);

}

else

{

#ifdef DEBUG

Serial.println("===== Bad Tilt Servo input =====");

Serial.print(" - Servo: ");

Serial.println(command);

Serial.print(" - Position: ");

Serial.println(new\_pos);

#endif

}

}

else if(command == 'r')

{

// We want to read the LRF

analog = 0;

for(i = 1; i <= 20; i++)

{

// Read the ADC value of the analog input pin

val = analogRead(SF01\_ANALOG);

analog += val;

delay(150); //refresh time for laser is 125

}

//LPF the value

analog = analog/20;

// Convert this into a voltage

analog\_voltage = analog \* 0.0049;

// Convert the voltage into a distance using the SF01 settings

analog\_distance\_meters = analog\_voltage\*slope + SF01\_0\_0V\_DISTANCE;

//test response

Serial.flush();

Serial.println(analog\_distance\_meters);

#ifdef DEBUG

Serial.println("===== Read LRF =====");

#endif

}

else

{

#ifdef DEBUG

Serial.println("===== Bad Input =====");

Serial.print("Command: ");

Serial.println(command);

#endif

}

}

}

## Arduino LRF Test Serial Code

void setup()

{

Serial.begin(9600); //computer terminal

Serial3.begin(19200); //LRF

}

void loop()

{

Serial3.flush();

Serial3.println("d"); //command to read

while (!Serial3.available()) { Serial.println("Waiting"); }

Serial.println(Serial3.parseFloat());

delay(5000);

}

# Appendix B

The code contained in this appendix is for the Raspberry Pi.

## Serial Code

Note: Install this in */var/www/serial*

import sys

import serial

import time

import MySQLdb

DEVICE = '/dev/ttyACM0'

BAUD = 9600

ser = serial.Serial(DEVICE, BAUD)

#delay to get the serial port set up

#time.sleep(1)

db = MySQLdb.connect(host="localhost", user="root", passwd="system", db="nslaser\_db")

cur = db.cursor()

PAN\_SERVO = 'p'

TILT\_SERVO = 't'

LRF\_READ = 'r'

def moveServo(servo, position) :

ser.write(servo + str(position))

ser.flush()

time.sleep(1)

p = int(ser.readline())

if servo == 'p' :

cur.execute("UPDATE system\_info SET pan = " + str(p))

if servo == 't' :

cur.execute("UPDATE system\_info SET tilt = " + str(p))

return

def readLaser(printM) :

ser.write(LRF\_READ)

ser.flush()

time.sleep(1)

result = float(ser.readline())

if (int(printM) == 1) :

print str(result) + " m" #print so that the ajax call can pick it up

elif (int(printM) == 2) :

print str(result)

cur.execute("UPDATE system\_info SET distance = " + str(float(result)))

return float(result)

command = str(sys.argv[1])

if(command == LRF\_READ) :

readLaser(sys.argv[2])

elif(command == TILT\_SERVO or command == PAN\_SERVO) :

pos = str(sys.argv[2])

moveServo(command, pos)

ser.close()

db.commit()

cur.close()

db.close()

## MySQL Code

Note: Install this in */var/www*

import MySQLdb

import sys

import os

import subprocess

import time

CLEAR\_DB = 'c'

WRITE\_TO\_DB = 'w'

VERIFY = 'v'

DISP\_DB = 'd'

def clear\_database() :

cur.execute("DELETE FROM targets")

return;

def write\_target\_to\_database(target\_name) :

os.system("python serial/serialAll.py r 0") #update the distance value

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

t = row[1]

d = row[2]

#need to check values

while (p > 180 or p < 0) :

os.system("python serial/serialAll.py p 0") #get current pan position

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

while (t > 180 or t < 0) :

os.system("python serial/serialAll.py t 0") #get current tilt position

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

print "Successfully wrote " + str(target\_name) + " to database."

cur.execute("INSERT INTO targets (name, pan, tilt, distance) VALUES('" + str(target\_name) + "', " + str(p) + ", " + str(t) + ", " + str(d) + ")")

return;

def verify\_targets() :

cur.execute("SELECT \* FROM targets")

ret\_string = ""

count = 0

for row in cur.fetchall() :

name = row[0]

pan = row[1]

tilt = row[2]

distance = float(row[3])

os.system("python serial/serialAll.py p " + str(pan));

os.system("python serial/serialAll.py t " + str(tilt));

time.sleep(1)

curDistance = float(subprocess.check\_output("python serial/serialAll.py r 2", shell=True))

if abs(curDistance - distance) > .5 :

if count >= 1 :

ret\_string = ret\_string + ", "

ret\_string = ret\_string + str(name) + " was measured to be " + str(curDistance) + " m as apposed to targeted at " + str(distance) + " m"

count = count + 1

if count == 0 :

print "All targets are verified."

else :

print "Targets are off: " + ret\_string + "."

return

def display\_db() :

cur.execute("SELECT \* FROM targets")

count = 0

ret\_string = "\n\nTarget\t\tPan\tTilt\tDist (m)"

for row in cur.fetchall() :

name = row[0]

pan = row[1]

tilt = row[2]

distance = float(row[3])

ret\_string = ret\_string + "\n" + str(name)

if len(str(name)) >= 12 :

ret\_string = ret\_string + "\t"

elif len(str(name)) >= 6 :

ret\_string = ret\_string + "\t\t"

else :

ret\_string = ret\_string + "\t\t\t"

ret\_string = ret\_string + str(pan) + "\t" + str(tilt) + "\t" + str(distance)

count = count + 1

if count > 0 :

print ret\_string

else :

print "No targets in database."

command = str(sys.argv[1])

if (command == CLEAR\_DB) :

clear\_database()

elif (command == WRITE\_TO\_DB) :

write\_target\_to\_database(sys.argv[2])

elif (command == VERIFY) :

verify\_targets()

elif (command == DISP\_DB) :

display\_db()

db.commit()

cur.close()

db.close()

## Website Code

All code should be placed */var/www*.

### clean.php

import MySQLdb

import sys

import os

import subprocess

import time

CLEAR\_DB = 'c'

WRITE\_TO\_DB = 'w'

VERIFY = 'v'

DISP\_DB = 'd'

def clear\_database() :

cur.execute("DELETE FROM targets")

return;

def write\_target\_to\_database(target\_name) :

os.system("python serial/serialAll.py r 0") #update the distance value

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

t = row[1]

d = row[2]

#need to check values

while (p > 180 or p < 0) :

os.system("python serial/serialAll.py p 0") #get current pan position

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

while (t > 180 or t < 0) :

os.system("python serial/serialAll.py t 0") #get current tilt position

cur.execute("SELECT \* FROM system\_info")

for row in cur.fetchall() :

p = row[0]

print "Successfully wrote " + str(target\_name) + " to database."

cur.execute("INSERT INTO targets (name, pan, tilt, distance) VALUES('" + str(target\_name) + "', " + str(p) + ", " + str(t) + ", " + str(d) + ")")

return;

def verify\_targets() :

cur.execute("SELECT \* FROM targets")

ret\_string = ""

count = 0

for row in cur.fetchall() :

name = row[0]

pan = row[1]

tilt = row[2]

distance = float(row[3])

os.system("python serial/serialAll.py p " + str(pan));

os.system("python serial/serialAll.py t " + str(tilt));

time.sleep(1)

curDistance = float(subprocess.check\_output("python serial/serialAll.py r 2", shell=True))

if abs(curDistance - distance) > .5 :

if count >= 1 :

ret\_string = ret\_string + ", "

ret\_string = ret\_string + str(name) + " was measured to be " + str(curDistance) + " m as apposed to targeted at " + str(distance) + " m"

count = count + 1

if count == 0 :

print "All targets are verified."

else :

print "Targets are off: " + ret\_string + "."

return

def display\_db() :

cur.execute("SELECT \* FROM targets")

count = 0

ret\_string = "\n\nTarget\t\tPan\tTilt\tDist (m)"

for row in cur.fetchall() :

name = row[0]

pan = row[1]

tilt = row[2]

distance = float(row[3])

ret\_string = ret\_string + "\n" + str(name)

if len(str(name)) >= 12 :

ret\_string = ret\_string + "\t"

elif len(str(name)) >= 6 :

ret\_string = ret\_string + "\t\t"

else :

ret\_string = ret\_string + "\t\t\t"

ret\_string = ret\_string + str(pan) + "\t" + str(tilt) + "\t" + str(distance)

count = count + 1

if count > 0 :

print ret\_string

else :

print "No targets in database."

command = str(sys.argv[1])

if (command == CLEAR\_DB) :

clear\_database()

elif (command == WRITE\_TO\_DB) :

write\_target\_to\_database(sys.argv[2])

elif (command == VERIFY) :

verify\_targets()

elif (command == DISP\_DB) :

display\_db()

db.commit()

cur.close()

db.close()

### targeting.php

<!DOCTYPE html>

<html>

<head>

<title>New Spin RMS</title>

<script src="script\_min.js"></script>

<link rel="stylesheet" href="styles.css">

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>

</head>

<body background="nice\_snow\_@2x.png" onload="setTimeout('init();', 100);">

<nav class="navbar navbar-default" role="navigation">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="http://www.newspin.com">

<img src="ns\_logo.jpeg" width="25" height="30">

</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li><a href="clean.php">Home</a></li>

<li class="active"><a href="targeting.php">Targeting<span class="sr-only">(current)</span></a></li>

<li><a href="stream.php">Stream</a></li>

<li><a href="configuration.php">Config</a></li>

</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<div class="page-header">

<center> <h1>Targeting</h1> </center>

</div>

<div>

<center>

<img id="mjpeg\_dest" />

</center>

</div>

<div>

<center>

<br>

<br>

<button type='button' class="btn btn-info btn-log" id='current\_targets' onClick='display\_targets();'><span class="glyphicon glyphicon-list-alt" aria-hidden="true"></span> Current Targets</button>

<button type='button' class="btn btn-info btn-log" id='clear\_targets' onClick='clear\_targets();'><span class="glyphicon glyphicon-remove-sign" aria-hidden="true"></span> Clear Targets</button>

<br>

<br>

<div class="input-group">

<h5>Target name:</h5>

<input class="form-control" id="target\_name" type="text" name="target">

</div>

<br>

<button type='button' class="btn btn-info btn-log" id='select\_target' onClick='select\_target();'><span class="glyphicon glyphicon-plus-sign" aria-hidden="true"></span> Add Target</button>

<br>

<br>

<button type='button' class="btn btn-info btn-log" id='verify\_targets' onClick='verify\_targets();'><span class="glyphicon glyphicon-ok-sign" aria-hidden="true"></span> Verify Targets</button>

<br>

<br>

</center>

</div>

</body>

</html>

### stream.php

<!DOCTYPE html>

<html>

<head>

<title>New Spin RMS</title>

<script src="script\_min.js"></script>

<link rel="stylesheet" href="styles.css">

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>

</head>

<body background="nice\_snow\_@2x.png" onload="setTimeout('init();', 100);">

<nav class="navbar navbar-default" role="navigation">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="http://www.newspin.com">

<img src="ns\_logo.jpeg" width="25" height="30">

</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li><a href="clean.php">Home</a></li>

<li><a href="targeting.php">Targeting</a></li>

<li class="active"><a href="stream.php">Stream<span class="sr-only">(current)</span></a></li>

<li><a href="configuration.php">Config</a></li>

</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<div class="page-header">

<center> <h1>Stream</h1> </center>

</div>

<div>

<center>

<img id="mjpeg\_dest" />

</center>

</div>

</div>

</body>

</html>

### configuration.php

<!DOCTYPE html>

<html>

<head>

<title>New Spin RMS</title>

<script src="script\_min.js"></script>

<link rel="stylesheet" href="styles.css">

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>

</head>

<body background="nice\_snow\_@2x.png" onload="setTimeout('init();', 100);">

<nav class="navbar navbar-default" role="navigation">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="http://www.newspin.com">

<img src="ns\_logo.jpeg" width="25" height="30">

</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li><a href="clean.php">Home</a></li>

<li><a href="targeting.php">Targeting</a></li>

<li><a href="stream.php">Stream</a></li>

<li class="active"><a href="configuraiton.php">Config<span class="sr-only">(current)</span></a></li>

</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<div class="page-header">

<center> <h1>Configuration</h1> </center>

</div>

<div>

<center>

<button type='button' class="btn btn-info btn-log" id='reboot' onClick='reboot();'><span class="glyphicon glyphicon-retweet" aria-hidden="true"></span> Reboot</button>

<button type='button' class="btn btn-info btn-log" id='shutdown' onClick='shutdown();'><span class="glyphicon glyphicon-chevron-down" aria-hidden="true"></span> Shutdown</button>

</center>

</div>

</body>

</html>

### cmd\_func.php

<?php

function sys\_cmd($cmd, $name) {

$val = "none";

if(strncmp($cmd, "reboot", strlen("reboot")) == 0) {

shell\_exec('sudo shutdown -r now');

} else if(strncmp($cmd, "shutdown", strlen("shutdown")) == 0) {

shell\_exec('sudo shutdown -h now');

} else if(strncmp($cmd, "rlaser", strlen("rlaser")) == 0) {

$val = shell\_exec('sudo python serial/serialAll.py r 1');

} else if(strncmp($cmd, "pleft", strlen("pleft")) == 0) {

shell\_exec('sudo python serial/serialAll.py p 1');

} else if(strncmp($cmd, "pright", strlen("pright")) == 0) {

shell\_exec('sudo python serial/serialAll.py p 2');

} else if(strncmp($cmd, "tup", strlen("tup")) == 0) {

shell\_exec('sudo python serial/serialAll.py t 1');

} else if(strncmp($cmd, "tdown", strlen("tdown")) == 0) {

shell\_exec('sudo python serial/serialAll.py t 2');

} else if(strncmp($cmd, "center", strlen("center")) == 0) {

shell\_exec('sudo python serial/serialAll.py p 90; sudo python serial/serialAll.py t 90');

} else if(strncmp($cmd, "ct", strlen("ct")) == 0) {

shell\_exec('sudo python nslaser\_mysql.py c');

} else if(strncmp($cmd, "st", strlen("st")) == 0) {

$val = shell\_exec('sudo python nslaser\_mysql.py w '.$name);

} else if(strncmp($cmd, "vt", strlen("vt")) == 0) {

$val = shell\_exec('sudo python nslaser\_mysql.py v');

} else if(strncmp($cmd, "dt", strlen("dt")) == 0) {

$val = shell\_exec('sudo python nslaser\_mysql.py d');

} else {

$val = "Unknown command";

}

return $val;

}

$retVal = "none-out";

if(isset($\_GET['cmd'])) {

$cmd=$\_GET['cmd'];

$name = "none";

if(isset($\_GET['name'])){

$name=$\_GET['name'];

}

$retVal = sys\_cmd($cmd, $name);

}

echo $retVal;

?>

### script\_min.js

//

// MJPEG

//

var mjpeg\_img;

function reload\_img () {

mjpeg\_img.src = "cam\_pic.php?time=" + new Date().getTime();

}

function error\_img () {

setTimeout("mjpeg\_img.src = 'cam\_pic.php?time=' + new Date().getTime();", 100);

}

//

// Ajax Status

//

function createAjaxCommand(){

var ajax;

if(window.XMLHttpRequest) {

ajax = new XMLHttpRequest();

} else {

ajax = new ActiveXObject("Microsoft.XMLHTTP");

}

return ajax;

}

function reboot() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=reboot", true);

ajaxCmd.send();

}

function shutdown() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=shutdown", true);

ajaxCmd.send();

}

/\*

In the onreadystatechange in the ajaxCmd is where you would change the webpage via the response text

\*/

function pan\_left() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=pleft", true);

ajaxCmd.send();

}

function pan\_right() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=pright", true);

ajaxCmd.send();

}

function tilt\_up() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=tup", true);

ajaxCmd.send();

}

function tilt\_down() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=tdown", true);

ajaxCmd.send();

}

function read\_laser() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

alert('Distance to target: ' + ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=rlaser", true);

ajaxCmd.send();

}

function center() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=center", true);

ajaxCmd.send();

}

/\*

MySQL commands

\*/

function verify\_targets() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=vt", true);

ajaxCmd.send();

}

function clear\_targets() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

//alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=ct", true);

ajaxCmd.send();

}

function display\_targets() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

alert(ajaxCmd.responseText);

}

};

ajaxCmd.open("GET", "cmd\_func.php?cmd=dt", true);

ajaxCmd.send();

}

function select\_target() {

var ajaxCmd = createAjaxCommand();

ajaxCmd.onreadystatechange = function() {

if(ajaxCmd.readyState == 4 && ajaxCmd.status == 200) {

alert(ajaxCmd.responseText);

}

};

var obj = document.getElementById('target\_name');

if (obj.value == "") {

alert('Target name field must be filled.');

}

else if (obj.value.length >= 16) {

alert('Target name must be less than 15 characters.');

}

else {

var call = "cmd\_func.php?cmd=st&name=";

ajaxCmd.open("GET", call.concat(obj.value), true);

ajaxCmd.send();

}

}

//

// Init

//

function init() {

// mjpeg

mjpeg\_img = document.getElementById("mjpeg\_dest");

mjpeg\_img.onload = reload\_img;

mjpeg\_img.onerror = error\_img;

reload\_img();

}

### styles.css

<style>

#dummy {

}

#nav {

width:150px;

height:500px;

float:left;

background-color:white;

padding:10px;

text-align:center;

}

#section {

text-align:center;

height:500px;

width:600px;

padding:10px;

background-color:gray;

display:table;

}

#footer {

color:black;

font-family:verdana;

padding:10px;

background-color:white;

text-align:center;

width:770px;

}

#header {

text-align:center;

background-color:black;

color:white;

padding:10px;

font-size:250%;

width:770px;

}

.myBtn {

padding-left: 10px;

padding-top: 10px;

padding-right: 10px;

padding-bottom: 10px;

}

</style>

## OpenCV Code

### Targeting Calibration

import os

import cv2

import math

import time

##Resize with resize command

def resizeImage(img):

dst = cv2.resize(img,None, fx=0.25, fy=0.25, interpolation = cv2.INTER\_LINEAR)

return dst

##Take image with Raspberry Pi camera

img\_name = str(int(time.time())) + 'image.jpg'

os.system('raspistill -o images/' + img\_name)

##Load image

img = cv2.imread("/home/pi/opencv/images/" + img\_name)

grey = cv2.imread("/home/pi/opencv/images/" + img\_name, 0) #0 for grayscale

height, width = img.shape[:2]

total\_radius = 300

w\_y1 = (height/2) - (total\_radius/2)

w\_y2 = (height/2) + (total\_radius/2)

w\_x1 = (width/2) - (total\_radius/2)

w\_x2 = (width/2) + (total\_radius/2)

#want to reduce the amount of the image we are working with

cropped = img[w\_y1:w\_y2, w\_x1:w\_x2]

cv2.imwrite('laser.jpg', cropped)

##Run Threshold on image to make it black and white

ret, thresh = cv2.threshold(grey[w\_y1:w\_y2, w\_x1:w\_x2], 200, 255, cv2.THRESH\_BINARY)

cv2.imwrite('thresh\_laser.jpg', thresh)

##Use houghcircles to determine center of circle

circles = cv2.HoughCircles(thresh, cv2.cv.CV\_HOUGH\_GRADIENT, 1, 75, param1=50, param2=13, minRadius=0, maxRadius=175)

for i in circles[0,:]:

#draw the outer circle

cv2.circle(cropped,(i[0], i[1]), i[2], (0, 255, 0), 2)

#draw the center of the circle

cv2.circle(cropped,(i[0], i[1]), 2, (0,0,255), 3)

##Determine coordinates for center of circle

x1 = circles[0][0][0]

y1 = circles[0][0][1]

##print information

print "===== Calibration Program Results ====="

print "Blob Information: "

print "x1 = ", x1

print "y1 = ", y1

print circles

if math.fabs((total\_radius/2) - x1) > 0:

print "X distance off by: ", math.fabs((total\_radius/2) - x1), " pixels"

if math.fabs((total\_radius/2) - y1) > 0:

print "Y distance off by: ", math.fabs((total\_radius/2) - y1), " pixels"

##Resize image

img = resizeImage(img)

thresh = resizeImage(thresh)

cropped = resizeImage(cropped)

##Show Images

cv2.imwrite("thresh.jpg",thresh)

cv2.imwrite("img.jpg",img)

cv2.imwrite("grey.jpg",grey)

cv2.imwrite("cropped.jpg", cropped)

cv2.waitKey(0)

### SIFT Algorithm

import cv2

import numpy as np

img = cv2.imread('power\_substation\_focus1.jpg')

gray= cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

sift = cv2.SIFT()

kp = sift.detect(gray,None)

#img=cv2.drawKeypoints(gray,kp)

img=cv2.drawKeypoints(gray,kp,flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

cv2.imwrite('sift\_keypoints.jpg',img)

1. [www.newspin.com](http://www.newspin.com) [↑](#footnote-ref-1)
2. <http://vimeopro.com/newspin360/youcontrol> [↑](#footnote-ref-2)
3. <http://apps.ep.jhu.edu/course-homepages/3088-525.743-embedded-systems-development-laboratory-houser> [↑](#footnote-ref-3)
4. <http://www.lightware.co.za/shop/en/lrf-modules/31-sf01a.html> [↑](#footnote-ref-4)
5. <https://www.servocity.com/html/spt200_pan___tilt_system.html> [↑](#footnote-ref-5)
6. <https://www.servocity.com/html/hs-5685mh_servo.html> [↑](#footnote-ref-6)
7. <http://getbootstrap.com/> [↑](#footnote-ref-7)
8. <http://www.lightware.co.za/shop/en/content/8-software> [↑](#footnote-ref-8)
9. <http://en.wikipedia.org/wiki/Hough_transform> [↑](#footnote-ref-9)
10. <http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html> [↑](#footnote-ref-10)
11. <http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html> [↑](#footnote-ref-11)
12. <http://www.amazon.com/> [↑](#footnote-ref-12)
13. <http://servocity.com/> [↑](#footnote-ref-13)
14. <https://www.dropcam.com/> [↑](#footnote-ref-14)
15. <https://www.dropbox.com/> [↑](#footnote-ref-15)
16. <http://en.wikipedia.org/wiki/Scale-invariant_feature_transform> [↑](#footnote-ref-16)
17. <http://downloads.raspberrypi.org/noobs> [↑](#footnote-ref-17)
18. <http://raspberrypi.stackexchange.com/questions/11631/wifi-setup-for-multiple-networks> [↑](#footnote-ref-18)
19. <http://elinux.org/RPi-Cam-Web-Interface> [↑](#footnote-ref-19)
20. <https://github.com/silvanmelchior/RPi_Cam_Web_Interface> [↑](#footnote-ref-20)
21. <http://askubuntu.com/questions/73864/how-to-modify-a-invalid-etc-sudoers-file-it-throws-out-an-error-and-not-allowi> [↑](#footnote-ref-21)
22. <https://github.com/NewSpin/> [↑](#footnote-ref-22)
23. <http://arduino.cc/en/Reference/softwareSerial> [↑](#footnote-ref-23)