Fall | 14

Weekly Status Report

Wesley Myers

November 11th – 17th, 2014

wesley.y.myers@gmail.com

Table of Contents

Executive Summary

The project is behind on algorithm development, otherwise on track with requirements. The main hurdle was getting a functional website.

Website Development

- Button clicks work
- Alert pop up for laser distance
- Center System

Serial Interface

- Updated Python Script

Physical Infrastructure

- More secure laser mount

Website Development

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. 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, I learned a very handy trick[1]. If you ever screw up the sudeoers file, just run the following command

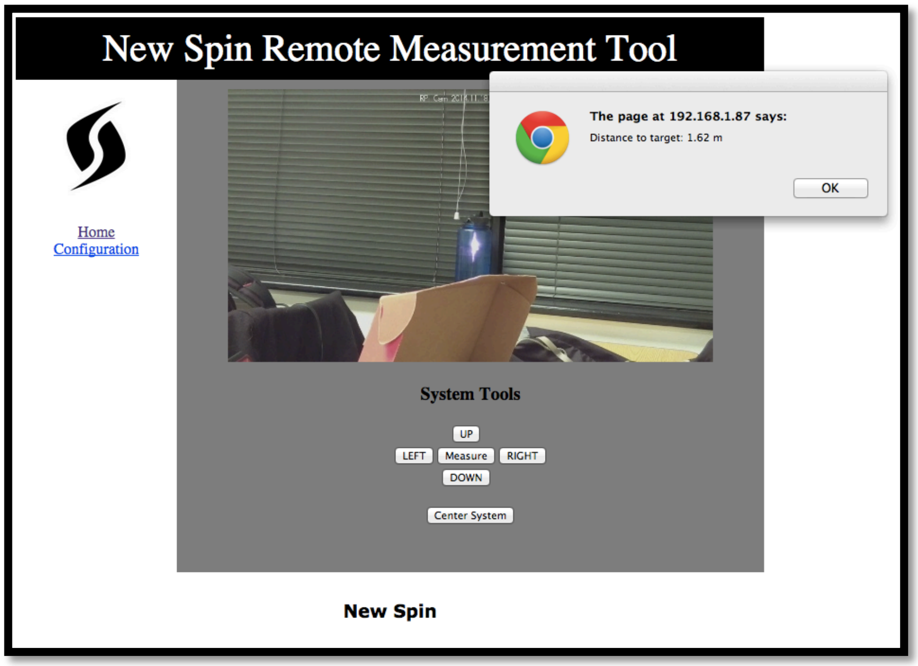
```
pkexec visudo -f /etc/sudoers.d/filename
```

Getting AJAX to work is a bit of magic for me. From my understanding, I send an AJAX command to what we may call as the back-end. There, I'm able to run system level commands to tell the Raspberry Pi to execute certain functions. In this case, I want the system to pan-tilt and read the laser. The tricky part now is trying to get data back to the user. In this case, the back-end returns data and the state of the

[1] http://askubuntu.com/questions/73864/how-to-modify-a-invalid-etc-sudoers-file-it-throws-out-an-error-and-not-allowi

AJAX command changes. Utilizing this, I can pull data back for the user to say what the laser measurement is.

In our system, the return comes back in the form of a pop up window on the system. The following figure illustrates this case.


New Spin Remote Measurement Tool

The page at 192.168.1.87 says:

Distance to target: 1.62 m

Figure 1 - New Website with Measurement

The visible changes to the website involve adding a button to center the system. Currently when one presses the left/right/up/down buttons, the system will move one-degree. This is slow and cumbersome to move it back to center. New possible additions might be to have a dropdown box to select laser steps/degrees.

Serial Development

This week I was unable to find time to debug the serial connection to the laser. I now have a copy of LightWare's serial terminal[2] up and running so that I can debug further.

The main focus of this week was to do some code clean up on the python scripts to command the laser. At the moment, all commands were separate scripts. In order to simplify this, I went ahead and consolidated into one file that took inputs. The code can be found in Appendix A.

[2] http://www.lightware.co.za/shop/en/content/8-software

Physical Infrastructure

I made some changes to the camera mount. The old mount simply had the camera taped on top of the laser. This was impractical since the camera moved around at 3 degrees-of-freedom and thus was off center on every new use.

The new mount is an L-bracket screwed onto the frame of the system. This now limits the camera to one degree-of-freedom and helps configure the system.

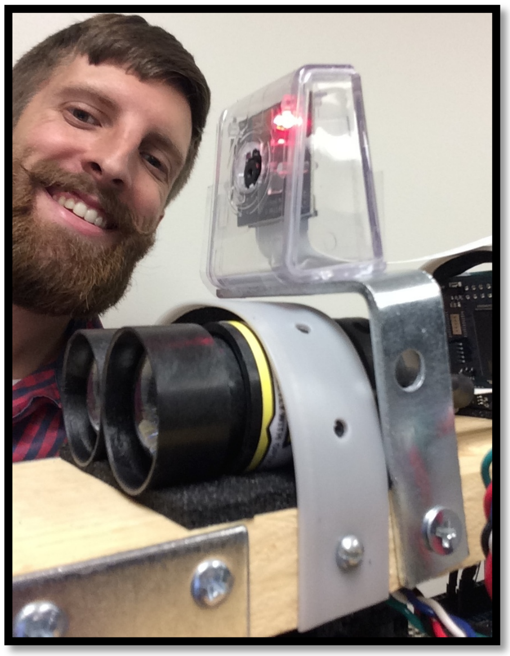


Figure 2 - New Mount (plus Chris)

Co-aligning the camera and laser is the desired end result. Thus as the distance of the laser changes, its position on the camera does not change. The problem with this is that the laser will not be located in the image in the center given the current system layout. With the new mount, we can aim the laser to be in the center of the frame at a certain distance. The laser will then start going high on the image at further distances, but lower on the image at closer distances.

Appendix A

```
import sys
import serial
import time

DEVICE = '/dev/ttyACM0'
BAUD = 9600
ser = serial.Serial(DEVICE, BAUD)

PAN_SERVO = 'p'
TILT_SERVO = 't'
LRF_READ = 'r'

def moveServo(servo, position) :
  ser.write(servo + str(position))
  ser.flush()
  time.sleep(1)
  return

def readLaser() :
  ser.flushInput()
  ser.write(LRF_READ)
  time.sleep(1)
  result = ser.readline()
  print float(result)      #print so that the ajax call can pick it up
  return float(result)

command = str(sys.argv[1])

if(command == LRF_READ) :
  readLaser()
elif(command == TILT_SERVO or command == PAN_SERVO) :
  pos = str(sys.argv[2])
  moveServo(command, pos)
```