Fall | 14

Weekly Status Report

Wesley Myers

November 4th – 10th, 2014

wesley.y.myers@gmail.com

Table of Contents

Executive Summary

Appear to be on track with requirements. This week's objective was to have a website up and running for the most part.

Interface Board

- Completed

Website Development

- Arduino reset on Serial Begin solved
- Buttons not working yet

Serial Development

- Minor tweaking done on Arduino end

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.

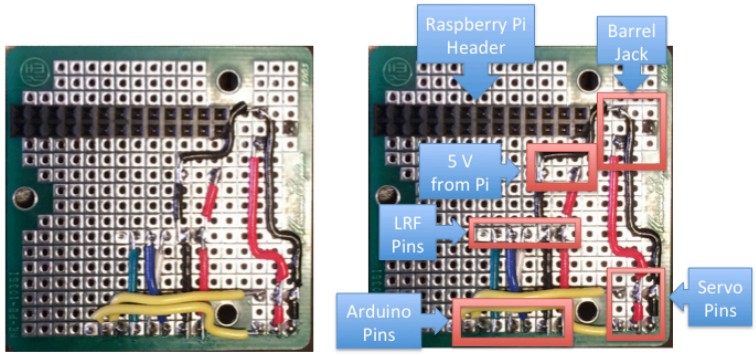

Figure 1 - Bottom of Interface Board

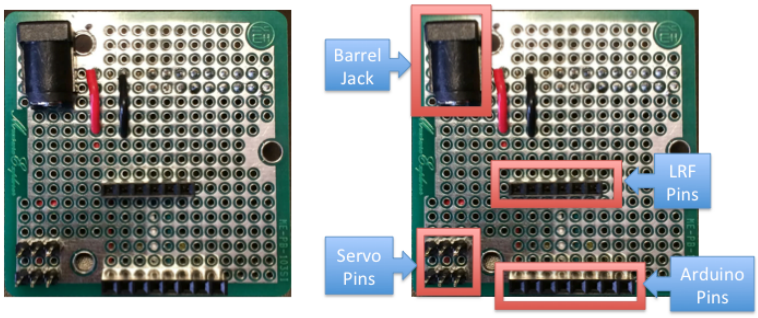

Figure 2 - Top of Interface Board

Website Development

So here is a fun problem: initiating serial causes a reset command to the Arduino. Shouldn't serial just be sending bits? How does initiating a serial connect reset the arduino? I may need to make the python script run persistently so that the reset command does not get sent. Either way, the voltage drop hypothesis from last week is invalid at this point. Regardless, the servos now have their own dedicated 5 V line.

This is a known problem with Arduinos and serial connections. There are multiple options with trying to tackle this problem. Many possible solutions are presented on an Arduino forum[1]. The one that works, on first try I might add, is to put a 10 uF capacitor on the RESET line connected to ground. This is a bit jenky of a solution as it is plugged straight into the Arduino. Version 2 (after the semester) of this system will have a more elegant solution. A side effect of this is that you cannot program the Arduino with the capacitor plugged in.

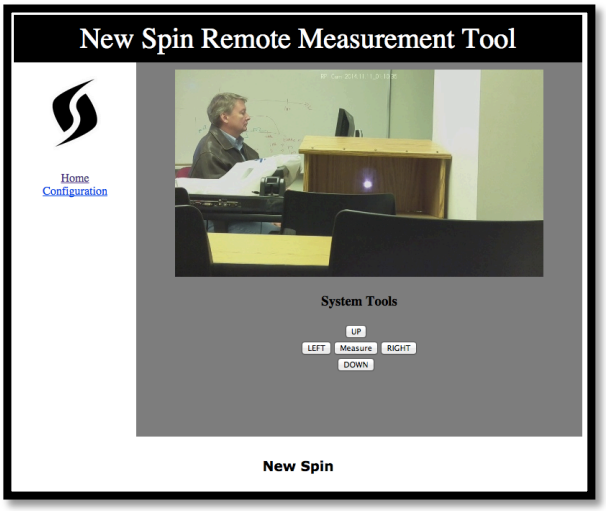

Figure 3 – Website

Currently the website's buttons do not work. I'm learning AJAX to try to figure this out. I tried using an example interface and it didn't work out of the box. The python commands for moving the system do work however now that the Arduino reset problem is solved.

[1] http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection

Serial Development

For commanding the servo position, I realized that the Arduino should maintain the state of the servos. Initially, I felt that the webserver should maintain this, but I realize that it doesn't make sense because the user will tell the servo to move a step to the left/right/up/down. Thus maintaining the state in the Arduino is the right choice. The code was changed to support this. The Arduino will now expect a command that could tell it to pan-tilt the system by one step instead of commanding it to go to a specific position. The code can be found in Appendix A.

The serial communication between the LRF and the Arduino will be addressed more extensively next week. There is nothing noteworthy on this effort for this week.

Appendix A

```
#include <Servo.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 0
#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;

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;

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);
}


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))
      {
        #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 == 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();
  }
  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))
    {
      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 == 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();
      }
      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 = 0; i < 5; i++)
      {
        // Read the ADC value of the analog input pin
        analog += analogRead(SF01_ANALOG);
        delay(150); //refresh time for laser is 125
      }

      //LPF the value
      analog = analog/5;

      // 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.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
    }
  }
}
```