Assignment 3
Sensor Characteristics

Luke Doman
Team 10

Introduction

The purpose of this assignment was to familiarize ourselves with utilizing sensors on our Squarebot platform to aid in the robots ability to interface with the environment. In previous labs we've used bumper sensors, light sensors, and ultrasonic sensors to accomplish several tasks, but with those instruments we were able to assume nominal performance according to the instrument's documentation. In this assignment we do sensor calibration ourselves to achieve desired results.

Active Directionality Sensing

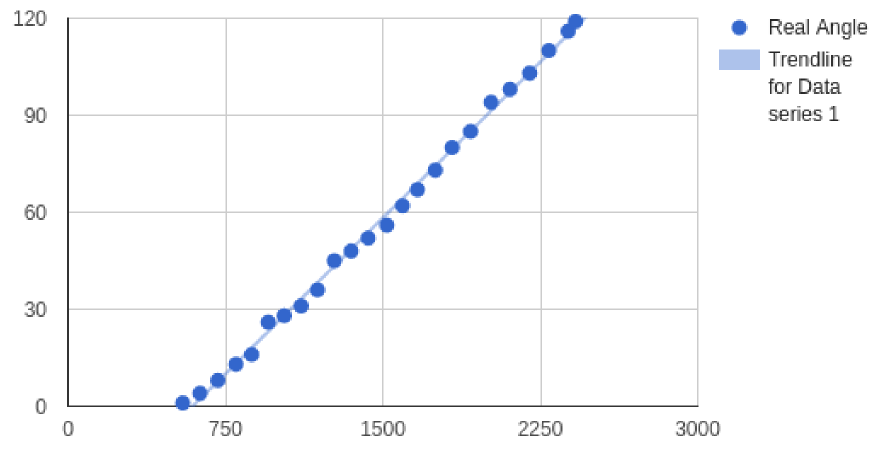
We had a few initial ideas to approach this problem, but ended up deciding on an architecture that iterates over every servo position and stores the light value at said position in an array. We used an array of size 254 to store the light values. The servo has available positions of -127 to 127, so the relationship of array index to servo position was: servo_position = array_index – 127. To scan for light we start by setting the servo position to -127 and then iterate over each servo position until 127, logging each light sensor value. At first we would wait about 200ms in between each iteration to allow for all these actions to take place, but after repeated tests we learned we could accomplish significantly faster performance without sacrificing accuracy and ended up running with 15ms between iterations.

We mounted our light sensor and servo to the front center of the Squarebot frame. While we considered mounting it to the side, we ruled it out due to foreseeing it may require additional calibration to get working correctly for Phase 3. After Squarebot performed a scan of all the light values at different servo positions we needed to be able to find the max and update the servo to point to it. This was easily accomplished with a for loop that compares the values of the array and returns the index with the max value. Once we had the index we just needed to set the servo to max_light_index – 127.

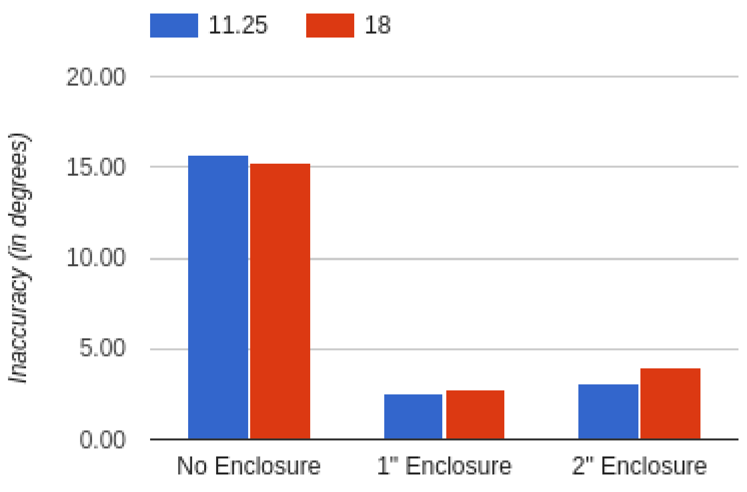
Accuracy Measurement

In our sensor mounting solution for Phase 1 we accounted for adding the potentiometer to our design so installing it did not require much effort. However, after starting Part B for this problem we encountered an issue that needed to be addressed. The potentiometer was on the same shaft as the light sensor, but was not secured down to anything. When we scanned for light we noticed that the potentiometer would not rotate in sync with the shaft it was on. We corrected this issue by screwing it down independently from the rest of the rest of the shaft.

Once we addressed our hardware issues we were ready to begin our calibration effort. We performed the calibration by having a loop iterate in increments of 10 and report the potentiometer reading while measuring the real angle on the protractor. With this data we were able to make a plot and derive an equation that defines the relationship of potentiometer readings and degrees. Our derived equation was: degrees = .064 * potentiometer_value – 38.157. Please see the plot below.



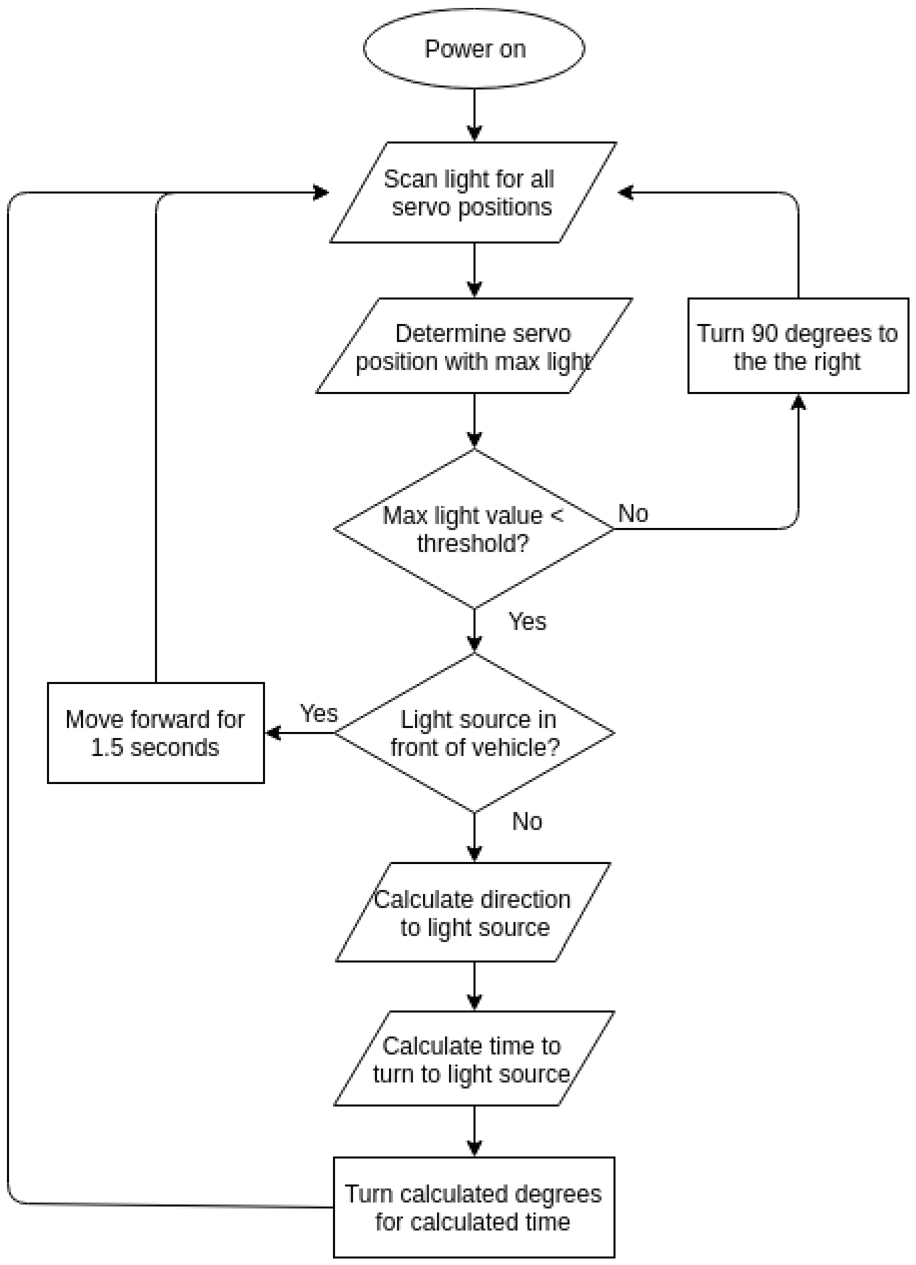
With our sensor calibrated we were able to move on to our data collection for parts b and c. As a light source we utilized one of our phones, and made a simple mounting device to hold it securely. The distances we used for calibration were 11.25 and 18 inches. Using the protractor, string, and a tape measure we taped off the various testing locations at the different angles and lengths for the light source on the desk so we didn't need to repeat work. For each testing location we collected 5 measurements and took the average to calculate the delta-angle. After recording all of the data in Google Sheets we were able to graph error the for each enclosure.



As shown above, the 1 inch enclosure provided the lowest error. Due to the way we divided our work amongst the team we tackled Phase 3 before all the data from this phase was processed. Because of this we used the 2 inch enclosure for Phase 3, however it still performed nominally. We could improve the performance further by replacing the 2 inch enclosure with the 1 inch and running again.

Phototaxis

Utilizing the code from the previous two parts of this assignment we were ready to tackle Phase 3. Due to the way we wrote the code for Phase 1 we able to use it exactly as is and just add logic around it that utilized it. Our logic for control is defined in this flow chart below.


Power on
Scan light for all servo positions
Determine servo position with max light
Turn 90 degrees to the the right
Max light value < threshold?
No
Yes
Light source in front of vehicle?
Yes
Move forward for 1.5 seconds
No
Calculate direction to light source
Calculate time to turn to light source
Turn calculated degrees for calculated time

To aid in simplifying the implementation we made a “move” function that took arguments to determine direction and amount of time to move. The performance of this control logic proved to be highly effective. The only major thing that could use improvement is adding a bumper sensor and some logic that does something other than keep running into the nests when it finds the light. While we did not attempt Phase 4 for this assignment, I believe this logic would perform very efficiently since it moves slow and reassess its situation often. Please see the implementation in the Appendix below.

Conclusion

In the previous assignments we did very informal sensor calibration by testing essentially just min and max values the sensor reported compared to what the documentation said it would. This assignment taught us a great deal about how to be scientific and accurate when calibrating sensors and using them to achieve real world results. It was incredibly satisfying seeing our robot successfully find the light and I look forward to the upcoming work with having robots interface with the real world more.

Appendix

```
#pragma config(Sensor, in1,    lightSensor,    sensorReflection)
#pragma config(Motor,  port2,           leftMotor,     tmotorVex393, openLoop)
#pragma config(Motor,  port3,           rightMotor,    tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port4,           frontServo,    tmotorServoStandard, openLoop)
#pragma config(Sensor, in2, POT, sensorPotentiometer)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

// Constants
int light_target_threshold = 500;
int forward_servo_pos = 65;
int move_time = 1500;
int dir_left = -1;
int dir_forward = 0;
int dir_right = 1;
int turn_time_multiplier = 4;

// Variables
int light_values[254];
int max_light_index;
int max_light_pos;
int turn_dir;
int turn_time;
int pos_diff;

// Move servo from min to max position recording light values for each index
void scanLight()
{
   // Set servo to starting position
   motor[frontServo] = -127;

   // Give servo time to get in position
   wait1Msec(1000);

   // Iterate over each servo position, updating array for each index
   int light_val = -1;
   for(int i = 0; i < 254; i++)
   {
      motor[frontServo] = i-127;                 // Update servo position
      wait1Msec(15);                             // Wait for it to finish rotation
      light_val = SensorValue(lightSensor);      // Get new light sensor value
      light_values[i] = light_val;               // Store light value for this position
   }
}
```

```
// Find servo position with max recorded light value
int getMaxLightPosition()
{
   // Declare required variables
   int minValue = 1040;
   int minIndex = -1;

   // Iterate over array to find max light value
   for(int i = 0; i < 254; i++)
   {
      if(light_values[i] < minValue)
      {
         minValue = light_values[i];
         minIndex = i;
      }
   }

   // Return min index
   return minIndex;
}

// Move for a given amount of time in direction left (-1), forward (0), or right (1)
void move(int dir, int time)
{
   int speed = 30;

   // Turn left
   if (dir == -1)
   {
      motor[leftMotor] = speed;
      motor[rightMotor] = -speed;
   }
   // Move forward
   else if(dir == 0)
   {
      motor[leftMotor] = speed;
      motor[rightMotor] = speed;
   }
   //Turn right
   else
   {
      motor[leftMotor] = -speed;
      motor[rightMotor] = speed;
   }

   // Wait for passed time then stop
   wait1Msec(time);
   motor[leftMotor] = 0;
   motor[rightMotor] = 0;
}

// Assignment 3.3
task main()
{
   wait1Msec(2000); // give stuff time to turn on

   // Main control loop
   while(true)
   {
      // Stop and scan
      move(0,0);
      scanLight();
      max_light_index = getMaxLightPosition();
      max_light_pos = max_light_index - 127;

      // If the max reported light is brighter than the threshold
      // pursue light source
      if (light_values[max_light_index] < light_target_threshold)
      {
         // If we are pointing at light source move forward until next loop
         if (max_light_pos >= forward_servo_pos - 15 && max_light_pos <= forward_servo_pos + 15)
         {
            move(dir_forward, move_time);
            continue;
         }

         // Set turn direction
         turn_dir = (max_light_pos <= forward_servo_pos) ? dir_right : dir_left;

         // Calculate time to turn
         pos_diff = forward_servo_pos - max_light_pos;
         turn_time = pos_diff * turn_time_multiplier + 100; // Need the + 100 for small values of
pos_diff

         move(turn_dir, turn_time);
      }
      // Else the detected light was negligible and we need to turn and
      // search for a brighter source
      else
      {
         move(dir_right, 1000);
      }
   }
}
```

```
    ////// Part 2a - Calibration

    //// Start left
    //motor[frontServo] = -127;
    //wait1Msec(500);

    //// Go all the way right
    //while(motor[frontServo] < 127)
    //{
    //    // Increment by ten
    //    int temp = motor[frontServo];
    //    motor[frontServo] = temp + 10;
            // wait1Msec(1000);

            // // Get pot value at new increment
            // float pot1 = SensorValue(POT);
            // writeDebugStreamLine("Pot1: %.6f", pot1);

            // wait1Msec(5000);
    //}


///////// Part 2b - Sensor data collection

    //scanLight();
    //max_light_index = getMaxLightPosition();
    //writeDebugStreamLine("Max light: %d", max_light_index-127);
    //while(true)
    //{
            // motor[frontServo] = max_light_index-127;
            // float pot = SensorValue(POT);
            // writeDebugStreamLine("Degree: %.6f", (pot/15.52));
    //    writeDebugStreamLine("Delta: %.6f", (pot/15.52) - test3_degree);
            // //wait1Msec(1000);
    //}
    //////
```