ME 451: Automated Window Blinds Management System

Final Report

Mitchell Brown

1 SENSING PLAN

Deciding when to close the blinds comes from two key factors; heat and light. When the window in question goes into full sunlight, it will get much more intense solar radiation, and if the ambient temperature is high enough, this will result in the room heating up to be much warmer than the outside. By monitoring these two factors, it becomes quite clear when the blinds need to be shut in order to mitigate heating from solar radiation. To do this, a photocell and thermistor were used.

1.1 Photocell Setup

The first sensor to evaluate was the photocell. This sensor was used to detect light levels in the window the setup would be in. To begin, it was important to determine the values the photocell would output compared to the amount of sunlight entering the room observed by the human eye. Figure 1 shows how light levels changed throughout a hot, partly cloudy day. Notice the fluctuations throughout peak sunlight hours, which are likely due to the sun moving behind clouds at some points. It is also important to note that in this plot, the light data is still arbitrary. The determining of how this light data equates to human observation will be discussed further in a later section.

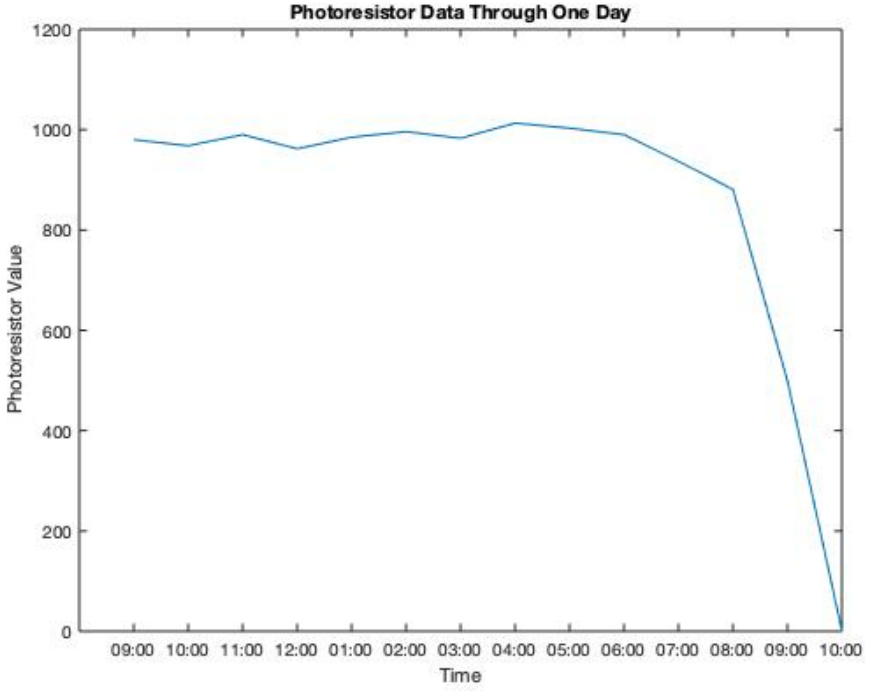

Figure 1: Light Level Readings in Corvallis Through One Day.

1.2 Thermistor Setup

The other sensor used for evaluation was a thermistor, which is used to detect the temperature in the room. This allows the system to determine if it is hot enough that the blinds should be closed, or if it will be a cool day and it should leave the blinds open even in direct sunlight. Figure 2 shows raw thermistor data taken throughout a hot day in Corvallis.

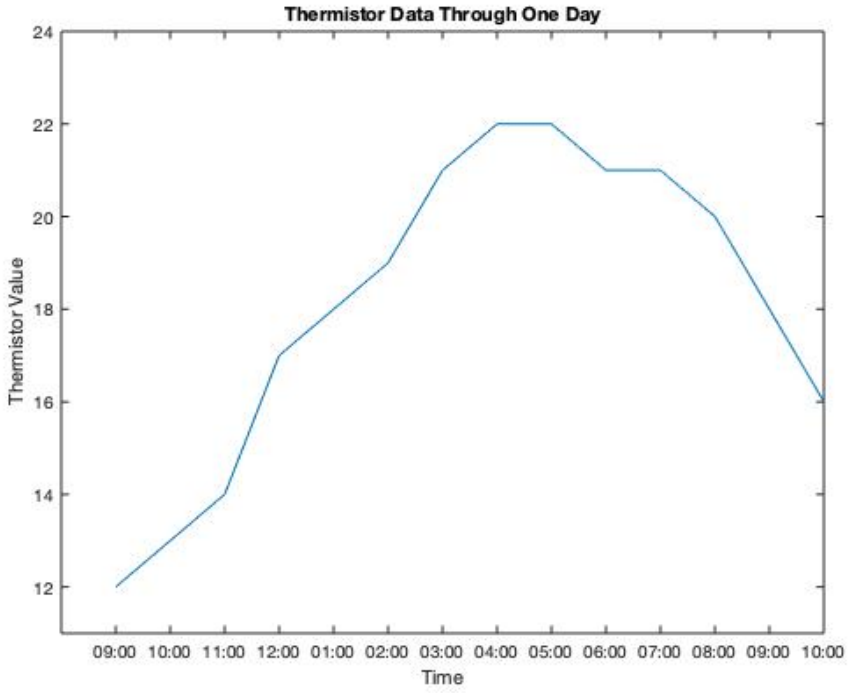

Figure 2: Temperature Readings in Corvallis Through One Day.

To achieve this data, a conversion equation was used. This equation uses a voltage divider to convert the thermistor values into real temperatures, which could then be used to more easily determine what values the blinds should be opened or closed. This equation is shown below.

$$Temperature = \frac{1}{(0.001129148 + 0.000234125 * log(10000) + 0.0000000876741 * log(10000)^3)} - 273.15 \quad (1)$$

1.3 Summary

Using the two sensors, three states of sensing can be achieved. The first state is when it is very bright out but not hot outside. A great example of this state is during the winter, when it is in the 40-degree range, even in direct sunlight, the blinds will not need to shut. The second state is cool and dim, which occurs whenever the window is not in direct sunlight and the temperature has gone below 23 Celsius, at which point the blinds will open. Finally, when it is hot and bright out, meaning it is above 24 degrees Celsius and the sensors are in direct sunlight, the blinds will close to block sunlight. Figure 3 shows a circuit daigram example for how these two components are setup with the Arduino.

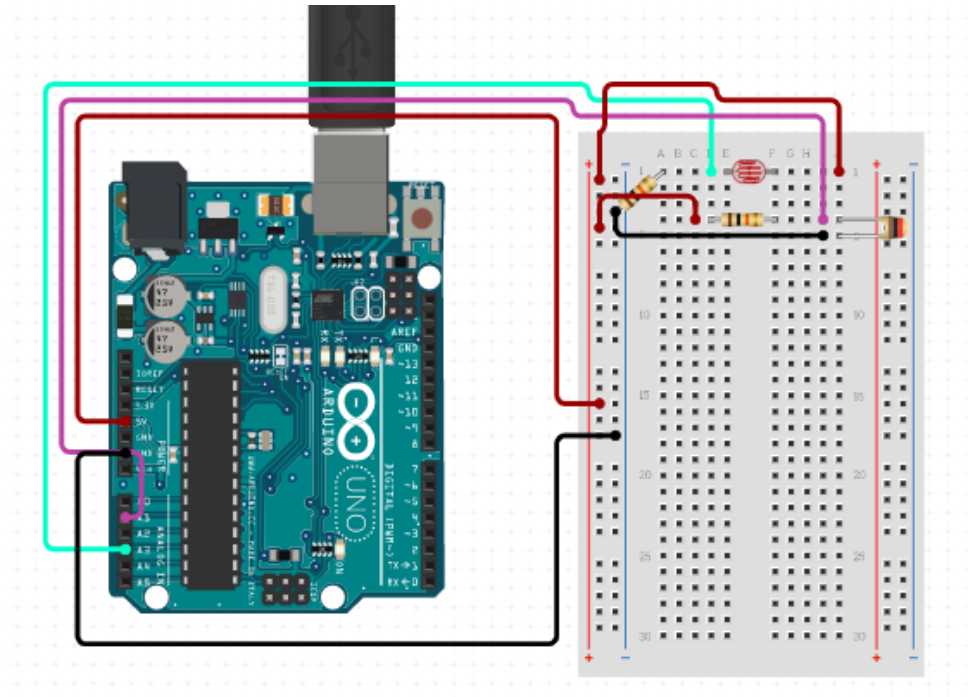

Figure 3: The Circuit Diagram for the Sensors.

2 Actuation Plan (20%)

For the actuation of closing the blinds, a stepper motor was used. This stepper motor had a wheel attachment, which was connected to a string loop. When the stepper motor rotated, it would pull the string in one direction, pulling the blinds open or closed depending on the direction of rotation. A circuit diagram for the stepper motor can be seen in Figure 4.

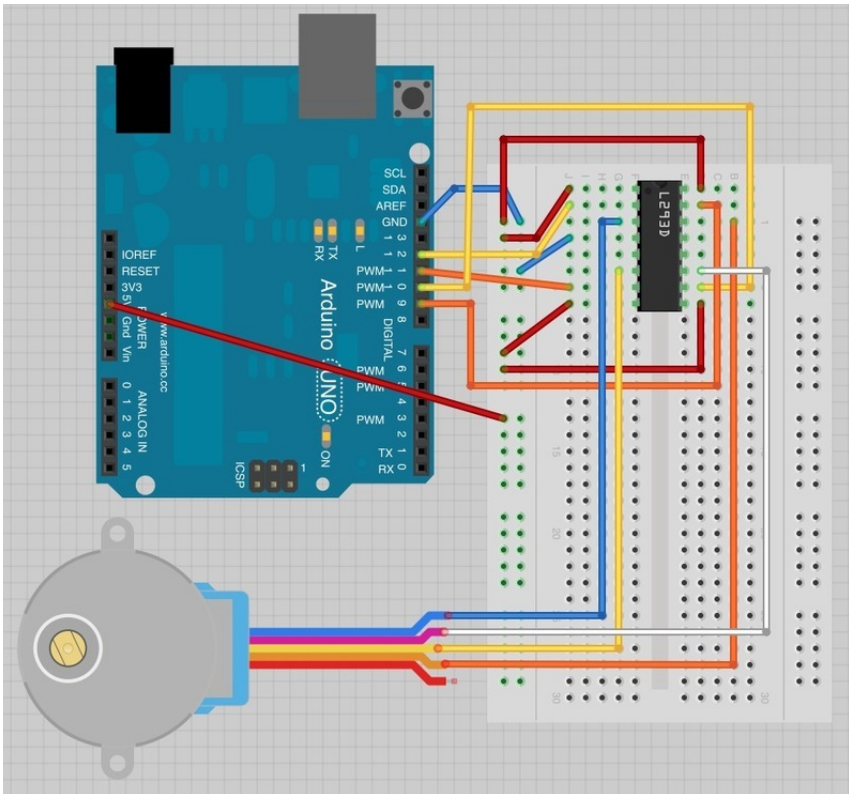

Figure 4: The Circuit Diagram for the Stepper Motor.

There are three behaviors of the actuator. The first is rotating clockwise, which would pull the blinds shut. The second state is rotating counter-clockwise, which would pull the blinds open. The third was remaining stationary, both in the open and closed states, which would allows the system to sense the surroundings until it determined when the blinds needed to be shut.

The information which led me to determine when the three states needed to occur came from both sensor data as well as previous knowledge of what makes my room get hot and need the blinds shut. For one, when the sun is pointing directly into the room and the outside temperature is already hot, the room tends to warm up to the point of being 10 to 20 degrees warmer than the outside, which means the blinds should have been shut during that time. However, there are many days throughout the year which never reach a high enough temperature, so it was important to recognize that the blinds could be left open even during direct sunlight.

3 SOFTWARE PLAN (10%)

For the software, there were a few key components used to ensure the system worked smoothly. The two main components were in a moving average filter and a boolean operator used to check if the windows are currently opened or closed.

The implementation of the moving average filter was key for ensuring the system would not bounce between open and closed if the light levels were fluctuating a lot. Often times in Corvallis, cloud covers move in front of the sun for a brief period, which causes the light levels to move up and down a lot in a short time span. By using the filter, it ensures the system will wait until it is absolutely positive that the light levels have remained high or low enough to need some sort of actuation. Figure 5 shows how the moving average filter can be used on the light sensor data.

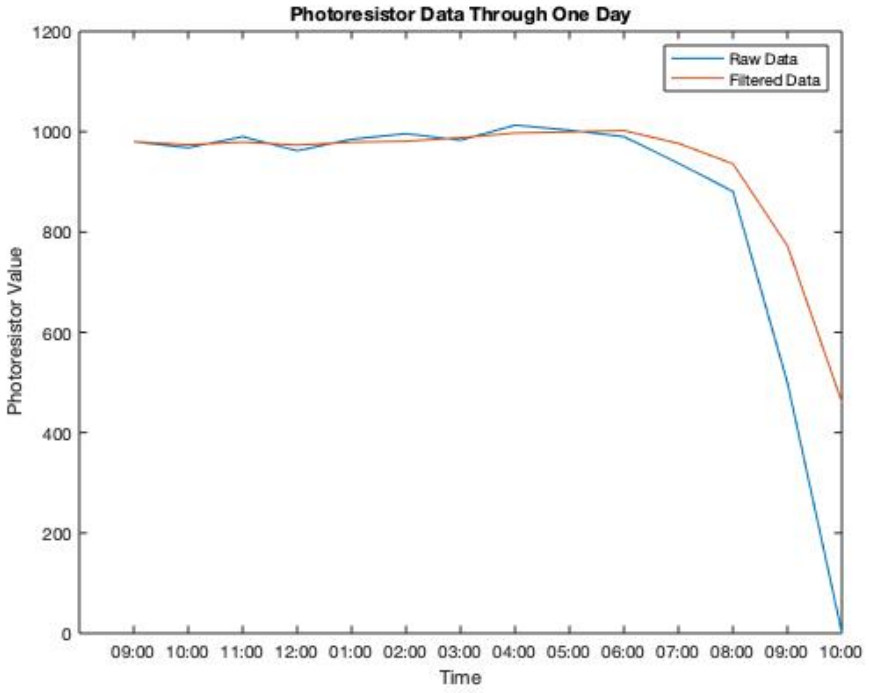

Figure 5: Raw Versus Filtered Light Data.

The boolean operator was utilized to ensure the system did not actuate the same sequence repetitively. What this means is if it is hot out and the blinds need closed, it will not try to close the blinds over and over, which would cause the system to jam and likely break.

4 DEBUGGING EXPERIENCE (20%)

One issue I encountered during implementation was the value needed to parse to the stepper to close and open the blinds to the correct distance. Figure 6 shows the wheel and string loop where they attached to the stepper motor.

Figure 6: The Stepper Wheel and String Loop Setup.

To determine the value parsed to the stepper, I first found the wheel circumference, and distance from fully open to fully closed on the blinds. The wheel circumference turned out to be 100.5 mm, and the distance from open to closed was 820 mm. These values were then used to determine how many rotations were necessary, and was found to be 8.16 rotations. Next, the number of ticks parsed to the stepper per rotation was found, which was 2048. With this and the needed rotations, it was found that the value parsed would be around 16,710. A summary of these calculations is shown below.

$$(820\ mm) * (\frac{1\ rotation}{100.5\ mm}) * (\frac{2048\ ticks}{1\ rotation}) = 16,710\ ticks \tag{2}$$

This value provided a good starting point for the value to be parsed to the stepper motor. However, there is a lot of inaccuracy in the calculation, as it is missing the slack in the string and any loss to friction, among other factors, so physical testing was necessary to determine a final value. After some testing, the value of 17,000 was chosen.

5 PROJECT CODE (30%)

Code used in the project can be seen below. Code in sections 5.1 and 5.2 were used for conditioning data from the two sensors, while code in section 5.3 was the final code used for the system.

5.1 Photocell Test Code

```
const int PCell = A0; // Set Photoresistor pin to Analog 0

int light; // Stores photoresistor values (0-1023)

void setup() {
  Serial.begin(9600); // Set up serial monitoring for recording data
  pinMode(PCell, INPUT); // Set the Photoresistor pin as an input

}

void loop() {
  light = analogRead(PCell); // Read light values
  Serial.println(light); // Print values for recording

  delay(1000); // 1 Second gap between recordings

}
```

5.2 Thermistor Test Code

```
// Initialize time variable as well as analogue and digital pin locations
unsigned long TIME_IN_MILLIS;
float A0_IN_VOLTS;
int analoguePin = A1;

int R2 = 10000;
float Vi = 5.00;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

}

void loop() {
  // record time of program
  TIME_IN_MILLIS = millis();

  // read in analogue pin value
  int A0_VALUE = analogRead(analoguePin);
  A0_IN_VOLTS = A0_VALUE * 0.0049; // Volt per unit conversion

  int R1 = R2*(Vi/A0_IN_VOLTS - 1);

  float tempC = (1/ (0.001129148 + 0.000234125 * log(R1) + 0.0000000876741 * pow(log(R1), 3)
```

```
  // print out the values in order(TIME_IN_MILLIS, A0_VALUE, 2_VALUE)
  Serial.print(TIME_IN_MILLIS);
  Serial.print(",");
  Serial.println(tempC);
  delay(10);
}
```

5.3 Final System Code

```
#include <Stepper.h>

// Stepper motor communication pins
int in1Pin = 12;
int in2Pin = 11;
int in3Pin = 10;
int in4Pin = 9;

// Rolling average values
float tempC;     // current value
float prev1 = 0; // previous value
float prev2 = 0; // 2 values prior
float avg = 0;   // average of the three values

// Value for tracking if the blinds are open or closed
bool closed = false;

// Initiate Stepper motor
Stepper motor(512, in1Pin, in2Pin, in3Pin, in4Pin);

const int PCell = A0; // Set Photoresistor pin to Analog 0
const int Therm = A1; //Set thermistor pin to Analog 1

float ThermVal; // Voltage value of thermistor reading

int light; // Stores photoresistor values (0-1023)
int R2 = 10000; // Thermistor resistor value (ohms)
float Vi = 5.00; // Input voltage to thermistor

void setup() {

  // Set up stepper pins
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);

  motor.setSpeed(50);

  pinMode(PCell, INPUT); // Set the Photoresistor pin as an input

}

void loop() {
  light = analogRead(PCell); // Read light values
```

```
  ThermVal = analogRead(Therm) * 0.0049; // Thermistor reading converted to Volts
  int R1 = R2*(Vi/ThermVal - 1);

  // Convert Thermistor Value to Celsius
  tempC =  (1/ (0.001129148 + 0.000234125 * log(R1) + 0.0000000876741 * pow(log(R1), 3))) -

  avg = (tempC + prev1 + prev2) / 3;

  // Close blinds when it gets too bright and hot
  if (avg >= 24 && light > 700 && closed == false) {
    motor.step(17000);
    closed = true;
  }

  // Open the blinds when it is cool and light is dim enough
  else if (avg < 23 && light < 600 && closed == true) {
    motor.step(-17000);
    closed = false;
  }


  // Set values for the moving average filter
  prev2 = prev1;
  prev1 = tempC;

  delay(1000); // 1 Second gap between recordings

}
```