Objectives:

Devise a test plan and develop your second prototype. Get customer feedback on your prototype

Instructions:

Teams will outline a prototyping test plan based on the template provided in “Lecture 11 – Prototyping Test Plan” and develop a prototype which will be used to achieve the objectives your team has set out in this plan (i.e. you need to answer the “why”, “what” and “when” of prototyping). Typical objectives include communicating and getting feedback for ideas, verifying feasibility, analysing critical subsystems or system integration or reducing risk and uncertainty. You must also define a stopping criteria which will allow you to end the test once you are satisfied that you have achieved your testing objectives. Be very clear about what you are trying to measure and define an acceptable fidelity based on the objectives of your prototype. Since this will be your team’s second prototype, your justifications and reasoning for this prototype should include a short explanation of your results from your previous prototype and how this second prototype continues the development of your solution. This second prototype should be of a critical (or the most critical) subsystem, in order to ensure that your design will work (keeping in mind the total course budget of $100). Get creative in order to improve your results. An analytical, numerical or experimental model should also be included. Finally, you must gather feedback and comments on your ideas and prototype from potential clients/users that you have sought out and identified on your own. Carefully document your prototyping test plan, modeling techniques and results (including detailed images of your prototype), as well as the feedback and comments you have gathered. Again, it is strongly recommended that you start early, as prototyping takes a significant amount of time. You only have one week to complete this prototype, but keep in mind that this prototype can integrate into a more comprehensive prototype later.

Lighting:

In our first prototype, we designed diagrams of how the lighting system was to be automated. For the second prototype, we bought some of our materials and put our diagrams to the test. We began by constructing the circuit for the indoor lighting, which would be powered directly by a 120V power supply, controlled by a simple 2-way light switch. Figure 1 shows the circuit with the switch off, while figure 2 shows the light on while the switch is on, meaning the circuit was successfully implemented. Figure 3 demonstrates the circuit with more details, to show exactly how it was wired.

Figure 1

Figure 2

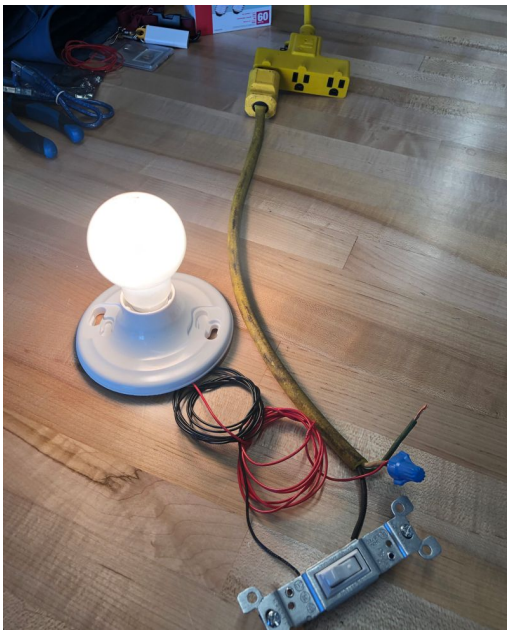
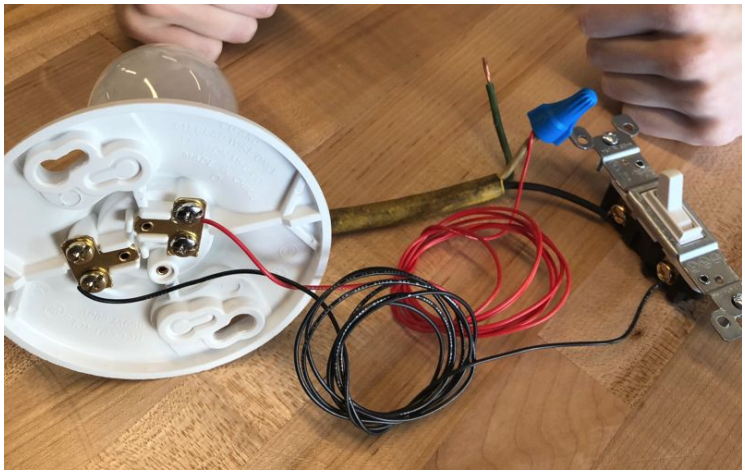


Figure 3



Since we are still waiting on the relay to implement the outdoor lighting circuit demonstrated in the diagram in the first prototype, we were unable to make the circuit. However, we have constructed the code that will be programmed into the arduino to power the automatic outdoor lighting. It can be found in the picture below.

sketch_feb28a §

```
int relay_outside = 8;
int relay_inside = 9;

volatile byte relay_outsideState = LOW;
volatile byte relay_insideState = LOW;

int sensor_outside = 2;
int sensor_inside = 3;

long lastDebounceTime1 = 0;
long debounceDelay1 = 10000;
long lastDebounceTime2 = 0;
long debounceDelay2 = 10000;

void setup() {
  pinMode(relay_outside, OUTPUT);
  digitalWrite(relay_outside, HIGH);
  pinMode(relay_inside, OUTPUT);
  digitalWrite(relay_inside, HIGH);
  attachInterrupt(digitalPinToInterrupt(sensor_outside), detectMotion1, RISING);
  attachInterrupt(digitalPinToInterrupt(sensor_inside), detectMotion2, RISING);
  Serial.begin(9600);

}

void loop() {

  if((millis() - lastDebounceTime1) > debounceDelay1 && relay_outsideState == HIGH)
  {
    digitalWrite(relay_outside, HIGH);
    relay_outsideState = LOW;
    Serial.println("OFF Outside\n");
  }

  if((millis() - lastDebounceTime2) > debounceDelay2 && relay_insideState == HIGH)
  {
    digitalWrite(relay_inside, HIGH);
```

```
sketch_feb28a §
  if((millis() - lastDebounceTime1) > debounceDelay1 && relay_outsideState == HIGH)
  {
    digitalWrite(relay_outside, HIGH);
    relay_outsideState = LOW;
    Serial.println("OFF Outside\n");
  }

  if((millis() - lastDebounceTime2) > debounceDelay2 && relay_insideState == HIGH)
  {
    digitalWrite(relay_inside, HIGH);
    relay_insideState = LOW;
    Serial.println("OFF Inside\n");
  }
  delay(50);
}

void detectMotion1() {
  Serial.println("Motion Outside\n");
  if(relay_outsideState == LOW)
  {
    digitalWrite(relay_outside, LOW);
  }
  relay_outsideState = HIGH;
  Serial.println("ON Outside\n");
  lastDebounceTime1 = millis();
}

void detectMotion2() {
  Serial.println("Motion Inside\n");
  if(relay_insideState == LOW)
  {
    digitalWrite(relay_inside, LOW);
  }
  relay_insideState = HIGH;
  Serial.println("ON Inside\n");
  lastDebounceTime2 = millis();
}
```

The customers feedback when presenting all of the progress we made to the client was positive. The client was impressed that everything was done on schedule and that the prototype was looking good. However there were concerns that the client expressed during the meeting. One concern the client had was how the light bulbs were going to get the electricity to run from the battery. We already have done all the research necessary before the meeting in order to be well informed about this topic which in result provided us with an answer for the client right away. The answer is that the Solar team is providing standards outlets throughout the house which is too run on 120 Volts of alternating current. This is helpful for our scenario because a battery outputs direct current which light bulbs can use; however they will not run as efficiently as they can. Instead we are

planning on running our electricity to the converter that the solar team is providing so that the light bulbs will run on alternating current which is much more efficient. Alternating current is more efficient because the light bulbs are not constantly on the entire time, they are oscillating from turning off and on at a very high frequency so that the human eye cannot detect that the light is being turned off and on repeatedly. Therefore, this will further provide more efficiency and battery saving to the house which is a very important priority. In conclusion, the client was impressed with the progress that was made over this time period and was relieved when we answered all of her concerns and questions.

Heathing:

The heating system is comprised of four critical subsystems.The first is the temperature monitor, the second is the lcd, the third is the potentiometer, and the fourth is the relay. In this prototype three of the four subsystems have been integrated. The first system integrated is the temperature sensor. The temperature sensor has three pins, the first is voltage, the second is output, and the third is ground. In order to get a proper reading out of the sensor it is important to have a dedicated ground for it. Once the wiring is setup you next need to convert the 10-bit analog output into celsius. Once this was finished the sensor was properly outputting the rooms temperature into the serial monitor.

The second system is the LCD. The lcd being used for the project will be using the IC2 standard. The IC2 standard works well for what we need as it only needs two analog ports from the arduino. Integrating this component was fairly simple as there are only two wires that needed to be attached to the arduino and then two more connected to power and ground. Once the code was complete, we tested the lcd by printing the current temperature to it (as seen in figure 2). The third system is the potentiometer. To fully integrate the potentiometer into the system there are three parts that will need to be completed. The first part is to attaching it to the arduino, second is to relate the output of the component to the desired room temperature, and third will be to print the desired room temperature onto the LCD.

In terms of customer feedback, the main concern that was brought up was about the output of the LCD screen that was talked about during the last prototype. In particular, the client focused on the casing surrounding the screen, and the possibilities for having alternate ways to notify the user about the temperature.

Concerning the former, the client was fairly pleased with the design of the casing (as seen in figure 1), and stated that the only thing to watch out for was the size in order to make sure it is not too large. When discussing the alternatives to notify the user, the client brought up the idea of how an LED could be attached to the system, and could light up if the temperature being read was appropriate or not. After talking about it with the client however, it was to be settled on that instead of an LED, the backlight of the LCD screen would turn on/off instead, notifying the user about the general condition of the temperature from a distance away.

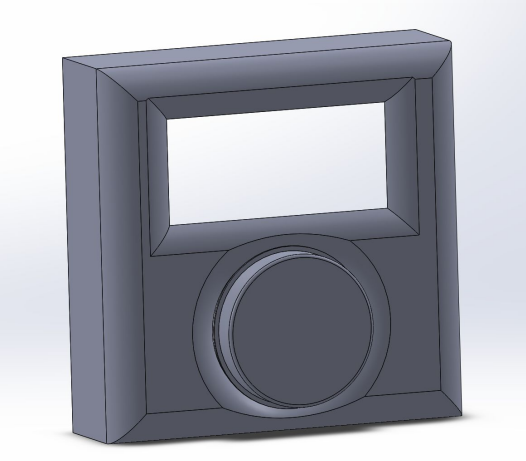


Figure 1 - thermostat casing

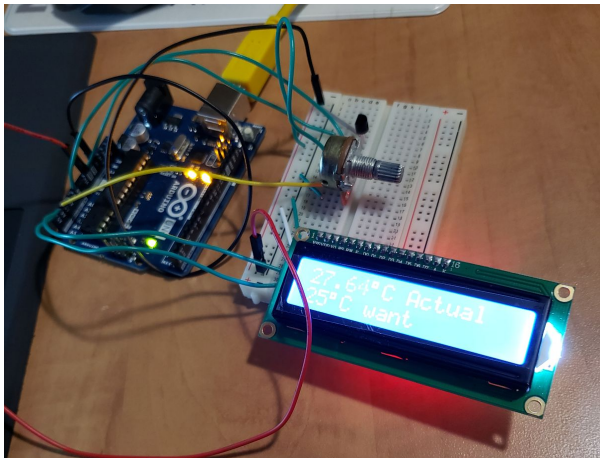


Figure 2 - current circuit

Heating Arduino code

```
 #include<Wire.h>
 #include<LiquidCrystal_I2C.h>
 LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup() {
 lcd.begin(16,2);
 lcd.backlight();
 Serial.begin(9600);

}

void loop() {
 heatingloop();

}
void heatingloop(){
 int sensorVal = analogRead(A0);
 int dialval = analogRead(A3);
 float temperature = calcTemp(sensorVal);  //Translates the data from temperature sensor into celsius
 int dtemp = settemp(dialval);  //translates the potentiometer output to desired temperature
 heater(temperature,dtemp); //turns on heater if required
 lcdHeat(temperature,dtemp);  // prints the the temperature on the lcd
}
float calcTemp(int sensorVal) {
 float voltage = ((sensorVal/1024.0)*5.0);
 float temperature = (voltage-0.5)*100;
 return temperature;
}
```

```
void lcdHeat(float temperature,int dtemp) {
  lcd.setCursor(0,0);
  lcd.print(temperature);
  lcd.print(char(223));
  lcd.print("C Actual");
  lcd.setCursor(0,1);
  lcd.print(dtemp);
  lcd.print(char(223));
  lcd.print("C want");
}
float settemp(float dialval){
  float dtemp = (dialval/40.0);
  return dtemp;
}
void heater(int temp,int dtemp){
  int relay1;
  if (dtemp > temp){
    relay1 = 1;
  }
  else{
    relay1 = 0;
  }
}
```

Water:

```
int solenoidPin = 4; // this is the output pin on the arduino we are using
int ultraPin = 10;  // input for ultrasonic

void setup() {
  // put your setup code here, to run once:
  pinMode(solenoidPin, OUTPUT);           //Sets the solenoidPin as an output
  pinMode(ultraPin, INPUT);                //Sets the ultraPin as the input

}

void loop() {
  // put your main code here, to run repeatedly:
  if digitalRead(ultraPin, HIGH){
  digitalWrite(solenoidPin, HIGH);  //switch solenoid ON
  }
  else{
  digitalWrite(solenoidPin, LOW);     //switch solenoid OFF
  }


}
```

For the second prototype for water, the main difference from the first prototype was the fact that we actually implemented a code to allow the user to turn the solenoid on and off. Actual construction of the sink has not begun due to the fact that we just ordered our materials. In the following 2 weeks we will have a test prototype and a final product for water ready to be used for design day.

Regarding the client, there was not much to show her just yet but she was pleased with our overall layout/ idea for how the water will be supplied. We will be emailing the client pictures of our first design of the sink on Wednesday, before moving onto the final stage of the project.