Team 3 Documentation
Smart Lighting Control System

Table of Contents

Introduction

One of the major issues we've seen in college and in the real world was that the lights in buildings were always left on and never turned off, even though people would already have left the room. These countless hours of wasted electricity due to the lights being switched on is ultimately unsustainable, as a significant amount of energy is lost without anyone utilizing it. People typically forget to turn lights off, or they may simply be too lazy to switch them off.

In order to address this issue, our product is an automatic system that turns on the lights whenever a person enters a room, and will stay on as long as the user remains in the room. When they are no longer in the room, the lights will switch off by themselves. Additionally, the app allows users to configure the sensor system itself, so when they want to manually turn off the lights while they are in the room, they can use the application to switch the lights off.

The big difference between the infrared sensor we use and the motion sensor used in average public restrooms (which also turns on light automatically) is that motion sensor can only detect when a person enters a room, then turns off the lights after a set amount of time without any movement. In order to create a system where the lights are continuously powered on while there is a person in the proximity, the thermal sensor has to be installed such that the sensor can detect the presence of a human in any part of the room the sensor is installed in. Thus, the infrared sensor detects objects in an area instead of a single spot. e can accomplish the function only by deploying a couple of sensors over the whole ceiling.

Materials

House Components
- Acrylic Sheets
- Hot Glue

Electronic Components
- HiLetgo GY-906 Infrared Temperature Sensor
- DSD Tech HM-10 Bluetooth
- Arduino UNO
- Breadboard
- LED Lights
- 330 Ohm Resistor
- Resistors
- M/M + M/F Jumper Cables
- Micro USB Cable

Software
- Arduino IDE
- MIT App Inventor
- InkScape

Tools
- Computer
- Laser Cutter
- Hot Glue Gun
- Electrial Tape

Instructions

The product has some software components as well as some hardware components. A model house was built using laser-cut pieces of acrylic to simulate a domestic setting.

The software components involve setting up the thermo sensor to make the output useful to us in detecting the presence of humans and not other objects, so the program will switch on the lights only when humans are present.

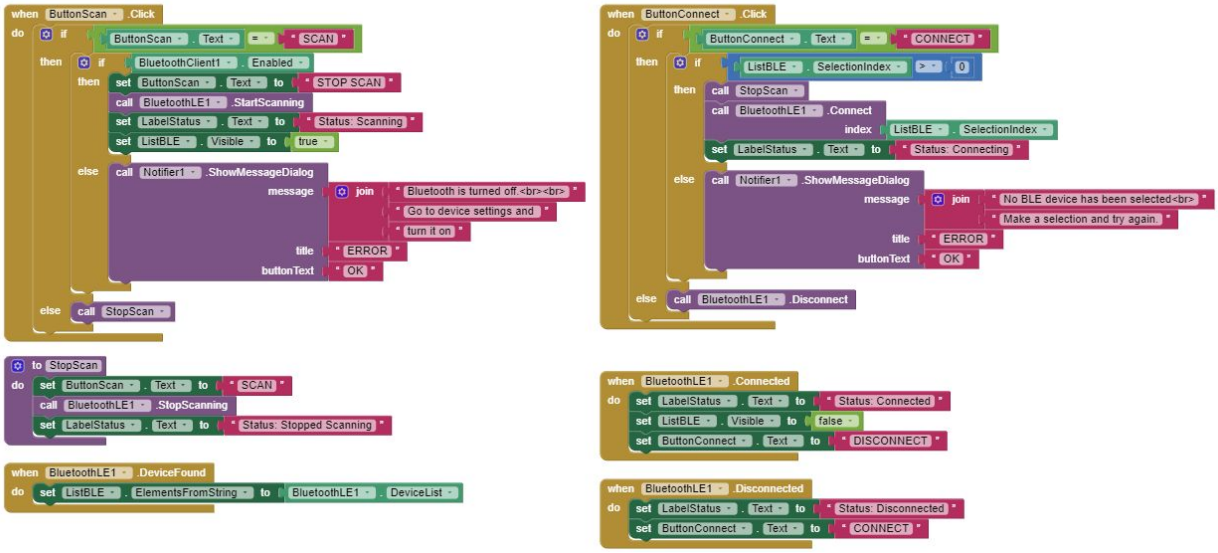
The hardware components mainly include the sensor, the bluetooth module to connect to phones, and the circuit setup, as well as the output, which are the LED lights.

Software

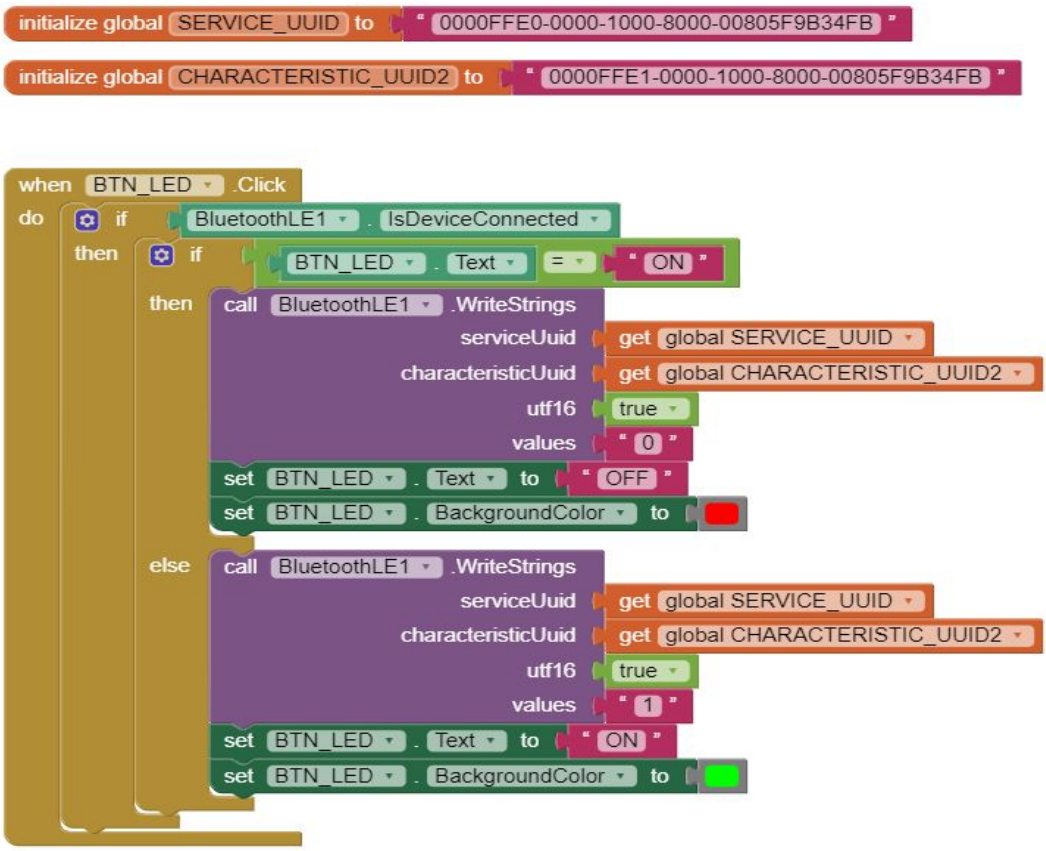
Creating MIT Mobile App (Android)

Since our project relies on the use of a switch to act as the controller for the thermal sensor, we used an app creator to develop a simple mobile app that can toggle the thermo sensor. However, it is important to understand that this mobile app is not a switch to turn the lights on or off. Rather, this app is programmed to turn the program that controls the thermal sensor on or off. Once the thermal sensor is turned off by the remote, the lights will continue to work like standard light switches. And we have to use a button to manually control the light just like the traditional switch on the wall.

Firstly, set up all the codes to scan Bluetooth on one's phone via the app.



After clicking the Button (when phone is connected to HM-10), we send 0 to the Arduino if it's an "off", and send 1 to Arduino if it's an "on", changing the color of the button at the same time. (Green is on and Red is off)



Set Up for Thermal Sensor Code

We choose the library from SparkFunMLX90614 and AltSoftSerial to apply commands to our Thermal Sensor and BlueTooth HM-10. In order for the program to work, these libraries must be included.

```
#include <Wire.h> // I2C library, required for MLX90614
#include <SparkFunMLX90614.h> // SparkFunMLX90614 Arduino library
#include <AltSoftSerial.h>

AltSoftSerial ASSserial; //Create an AltSoftSerial object to interact with
Bluetooth Module
IRTherm therm; // Create an IRTherm object to interact with throughout
```

The next thing we have to do is to set up all Pins or global Variables and assign them to the specific pins on the Arduino so that the Arduino knows where to direct the power to.

```
byte LED = 2;  // LED stands for the light in our room
int BUTTON = 4;  // Control the light manually with Button
char c=' ';  // store the text from the bluetooth into character c
int function = 0;  // detecting system is off if it's 0, and on if it's 1
```

Using function from the SparkFun library can set the temperature Units into Fahrenheit or Celsius

```
therm.begin(); // Initialize thermal IR sensor
therm.setUnit(TEMP_C); // Set the library's units to Celcius/Farenheit
```

In the main Loop, first we print out the current room temperature using the function ambien()

```
Serial.println("Room Temperature is " + String(therm.ambient()));
```

Check if the board receives any value from the bluetooth. If so, store the value (0 or 1) into character c. Print out c for testing. Set the function off if c is 0 and set it on if c is 1 (48 in ASCII stands for integer 0 and 49 stands for 1).

```
  if (ASSserial.available()){
    delay(10);
    c = ASSserial.read();
    Serial.println(c);

    if (c == 48) { function = 0;}
    if (c == 49) { function = 1;}
  }
```

Here's the most important part of the algorithm. If the function is on, then we are going to detect human body in the room automatically via the Thermal Sensor. The function read() returns 0 or 1 based on whether the board reads any object and ambient temperature from the sensor. Print "Unable to read" if there's no value. We set it to be 3 between the temperature of human body and non human body (could change it based on different scenarios). If the difference between the

temperature of the object right below the sensor and that of the ambient environment is greater than or equal to 3, we say there's a human body! Hence we turn on the light and print out the temperature in Celcius to the Serial Monitor. Otherwise, the difference is less than 3, so there's no human at all and turn the light off.

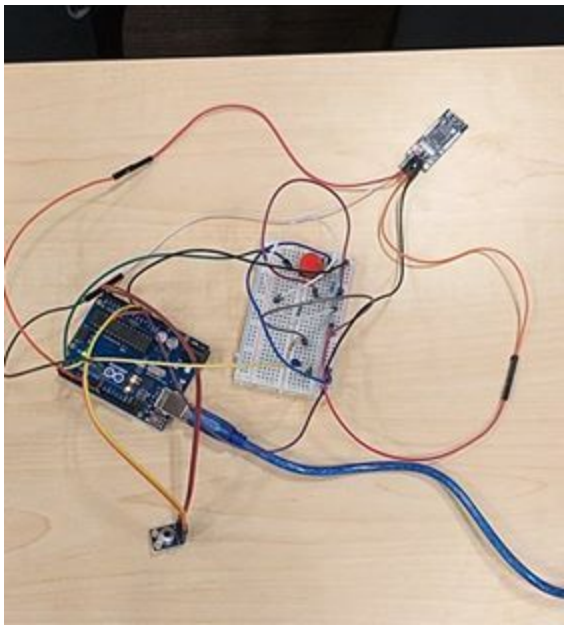
```
  if (function == 1){
    Serial.println("Now I am detecting the temperature...");
    if (therm.read())   // Call therm.read() to read object and ambient
temperatures from the sensor.On success, read() will return 1, on fail 0.
    {
      if (therm.object() - therm.ambient() >= 3){
        digitalWrite(LED,HIGH);
        Serial.print("Object: " + String(therm.object(), 2));
        Serial.write(0xC0 |(176 >> 6));
        Serial.write(176 & 0x3F | 0x80);
        Serial.println("C");
       }
       else {
        digitalWrite(LED,LOW);
        Serial.print("Object: " + String(therm.object(), 2));
        Serial.write(0xC0 |(176 >> 6));
        Serial.write(176 & 0x3F | 0x80);
        Serial.println("C");
       }
    }
    else
      Serial.println("Unable to read the thermal value.");
  }
```

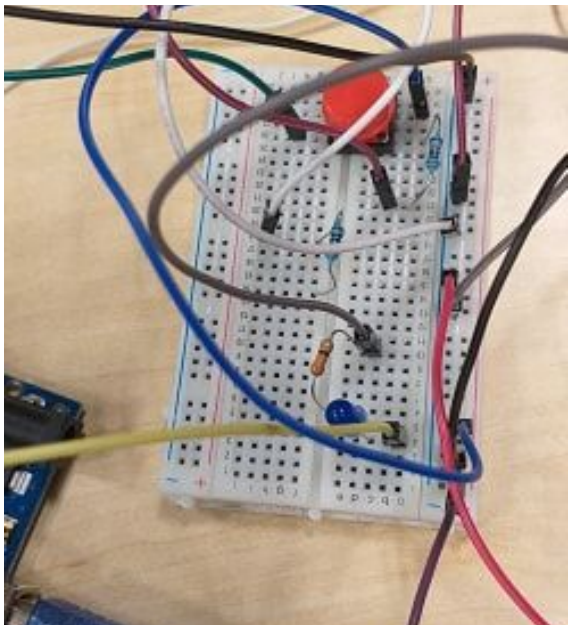
If the function is off, it enters a manual mode. In this manual mode, users can just use the button to control the light by switching it on or off as we traditionally do with current light switches. This function is for those who don't want the automatic detecting system on when they're sleeping or watching a movie while they are in the room with the sensors. This will make sure that the users can dictate whether or not they want the lights to be on at all times while they are not in proximity of the room.

```
  else if (function == 0){
    Serial.println("Manual Mode.");
    if (digitalRead(BUTTON)==HIGH && digitalRead(LED)==HIGH){
      digitalWrite(LED,LOW);
      delay(500);
    }
    else if (digitalRead(BUTTON)==HIGH && digitalRead(LED)==LOW){
      digitalWrite(LED,HIGH);
      delay(500);
    }
  }
```

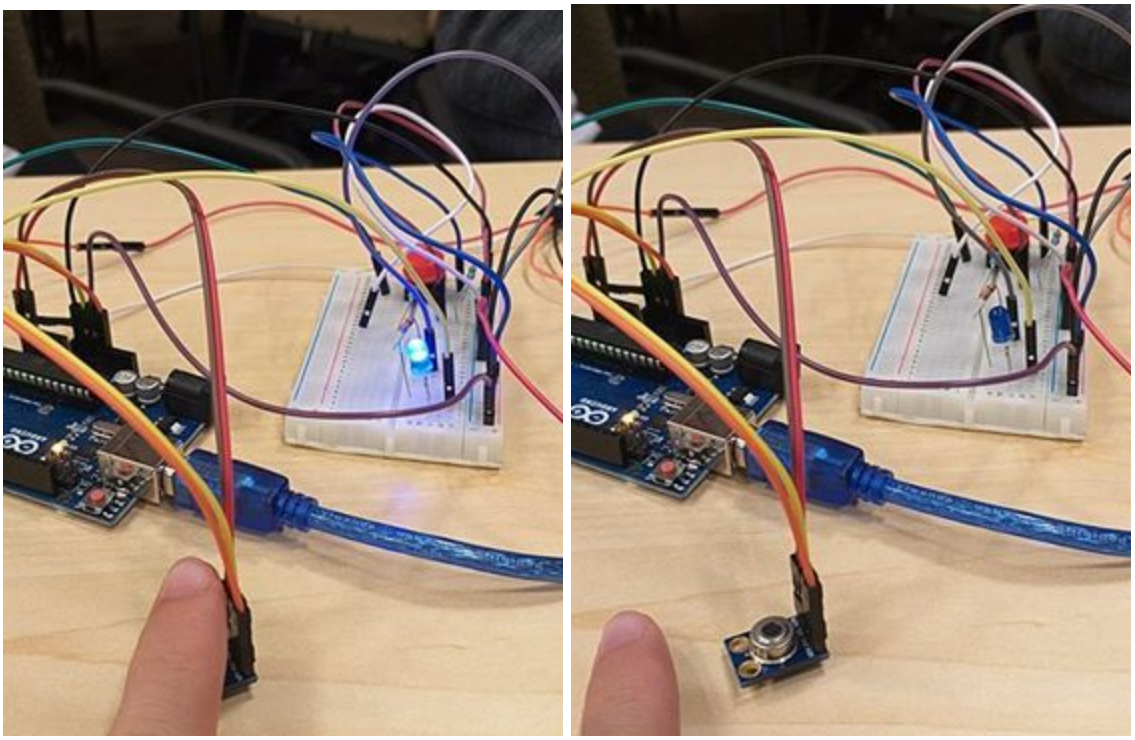
Hardware

In order to incorporate our project into a more domestic setting, we integrated our circuit onto a scale model of a house to demonstrate how the system would work in practice. Since our sensors aren't powerful enough to cover wide area, we had to create a smaller scaled model that would allow us to be in range for the thermal sensor to detect a significant amount of change in the temperature. Any source of substantial heat, such as a human walking into the proximity of the sensors, will trigger the program and turn the lights on. Additionally, our program does not detect the movement of objects that do not emit any heat. So if we were to move a piece of paper over the sensor, the light will not turn on due to there being a lack of sufficient heat emission. Our product is run on an Arduino UNO. We incorporated the bluetooth device and thermal sensor into the same circuit. The bluetooth device will receive signals from the phone application, which will then relay the input data to the Arduino. The Arduino will then utilize the bluetooth inputs to turn the function of the thermal sensor on or off, where the off position will allow for the use of the switch to manually turn the lights on or off. When the function is turned on, the button will be disabled, and the thermal sensor will be the device responsible for operating the LED. We needed to use the breadboard to connect the button, the LED lights, the bluetooth module, the heat sensor module, as well as including various resistors in between.



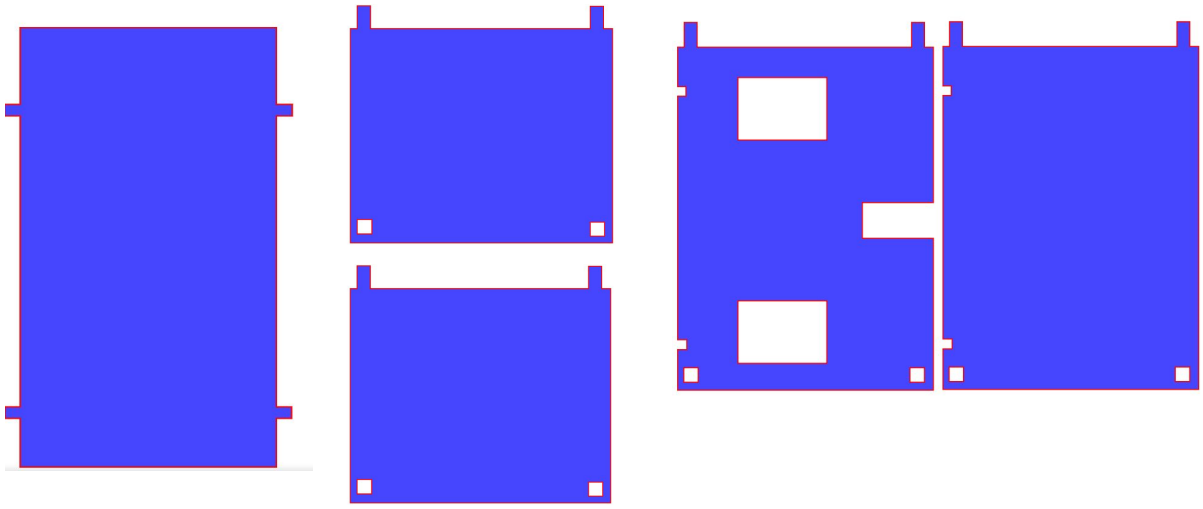


In this circuit, the LED is placed in series with the resistor. In our actual model, we placed a resistor in parallel with 2 LEDs to make the lights brighter and to provide light over a larger area in the scale modeled house. The left 2 pins of the thermal sensor must be connected to the sda and scl pins of the Arduino UNO, with the remaining two pins connected to a 5V pin and ground. The switch must also connect to a pin on the Arduino, and in our case we dedicated the left pin of the button to pin 4 on the Arduino, with the other right sided of the button connected to a resistor and then over to ground for the bottom right button pin. The top right end of the switch connects to the positive rail of the breadboard.

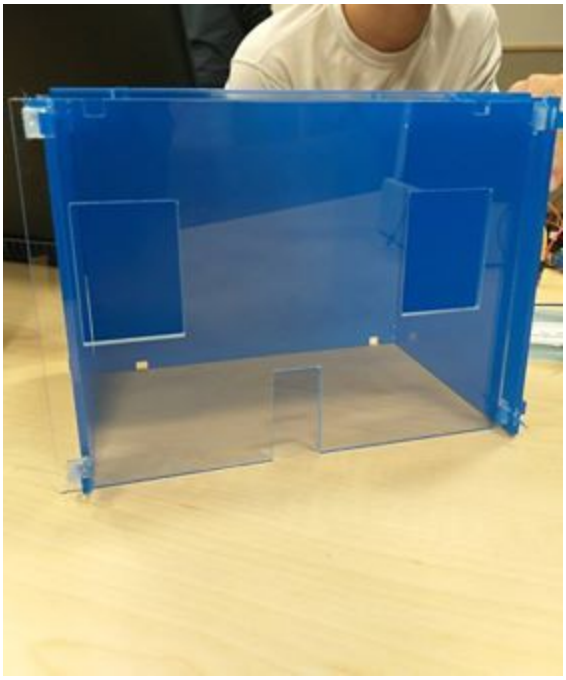


Testing with the thermal sensor can be done by placing a finger over the thermal sensor to turn on the LED. The LED should also turn off once the heat source, which in this case is the finger, is out of range of the thermal sensor. The sensor can be calibrated further by waving a piece of paper over the sensor or any other object over the sensor, and the light should not turn on.

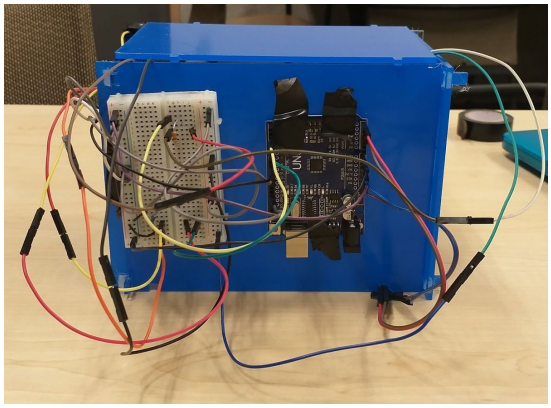
In order to create the scale model of the house to implement the sensors and lights into, we utilized a laser cutter to make each piece of the house. The design is imperfect, as there are more efficient ways to create these pieces such that they can fit more nicely together. However, for our small scale model, the design serves its purpose.

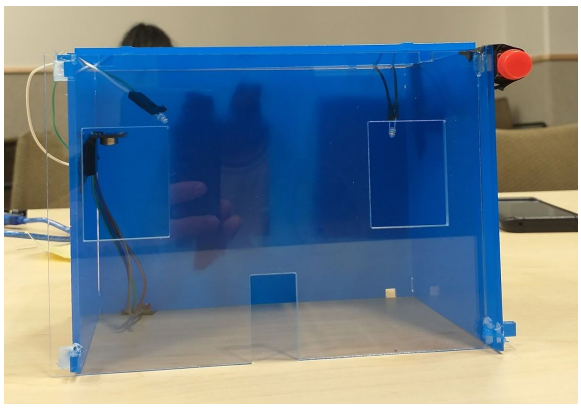


The final model of the house can be assembled by glueing the pieces of the parts together, with the protrusions of some pieces going through the square holes of the other pieces, and so on.



The final product can be assembled by attaching the Arduino and the breadboard to the backside of the scale model. The LEDs can be attached to longer male to female wires to wrap from the back of the scale model to the inside of the house. The switch is placed on the top right for convenience, with the thermal sensor inside the house model within range for our fingers to reach in and activate the lights hanging from the top.





Conclusion

Our product ended up working, with the option of having the lights controlled by the thermo sensor, and also having the option of having the lights controlled by the button, and this can be toggled using the phone app. We were happy that it worked out in the end, given some of the issues we had faced while making the product. Though our model house was fairly simple and the heat source for the sensor to detect was a human finger, it should be enough to demonstrate our concept of automated lights.

If we had more time to work on this project and more resources to utilize, the major change that we would make is to use a more advanced thermal sensor that can already distinguish the difference between human body heat and other sources of infrared radiation. Our thermal sensor at first worked more like a motion detector, where it would record inputs from any object regardless if the object radiated heat. So we had to spend some time modifying a public code to only recognize heat from humans. The thermal sensor also only had a limited range, so we had to make sure that our scale model was small enough to fit within the range of our sensor. If we were to implement our project in a real world situation rather than a scale model, then we would need a more powerful sensor that is capable of taking in inputs from a much larger proximity and range to accomodate an actual sized living room.

For some reason when we were attempting to implement a button to turn the LED on and off in the case that the thermal sensor malfunctions, or if we want a manual override, the button initially did not manage to turn the LED on. We had also tried to connect it to the bluetooth module, following a tutorial we found online that was based on an Arduino Nano model. We set it up accordingly and modified it to suit our Arduino UNO. After many tests, where we didn't manage to get the button to work, we checked our circuits and made adjustments. We found out that the bluetooth button eventually worked when we adjusted the pin positions, since the online instructions to setting up the circuit were based on an Arduino Nano when we had an Arduino UNO. Our proudest moment was being able to finally utilize our phone application to wirelessly turn on the LED light.

Moreover, while in the process of simplifying the circuit to reduce the number of wires used, we had misplaced an LED while testing which led to a short circuit. Initially, we were not sure what the issue was, whether it was the thermo sensor that malfunctioned, our circuit setup was incorrect, or if the whole Arduino was broken. Being quite late into the project, we panicked a bit, knowing we could not easily replace the thermo sensor. After testing every component, we realised it was the Arduino board that had malfunctioned, so we had to get that replaced, and it worked in the end.

To further extend the project, we can develop the mobile app further to notify users the amount of electricity they use compared with the conservation standards, or allows the user to switch the function in different rooms during different periods; for example, users might want the system off during their sleep but switches on when they get up and hurries going to work.

Other possible applications for thermo sensors and automated outputs may include automatic switches for air conditioning, fans, heaters, automation of blinds or curtains, etc. These would be switched on during appropriate times of the day when required, and can be configured to have their efficiency maximised while being used, and they can be switched off automatically when not required or when the users leave the room.

References

Martyn Currey who provided the codes for setting up the HM-10 bluetooth device
http://www.martyncurrey.com/arduino-hm-10-and-app-inventor-2/?fbclid=IwAR2988ztGqkjLbclOsE1UEV-LpBQmqrnMzFJvBCVDVRCzzUb78dQ8u_bBCo

MIT App inventor that allowed us to create our own mobile phone application for our project
http://appinventor.mit.edu/explore/index-2.html

AltSoftSerial Library that is needed to run the program
https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html

Instructions on how to set-up GY-906 thermal sensor
https://learn.sparkfun.com/tutorials/mlx90614-ir-thermometer-hookup-guide/all