Abby Starr
Light Switch Thingy
12/10/2018

Light Switch Thingy

When we were trying to come up with a project idea, one thing that interested in was using an RFID sensor to detect something. We then formulated the idea to make a door locking system, where when the card is scanned, the door will either lock or unlock. When the door is unlocked, the light will turn on and off according to how bright it is in the room. If there is enough light, there is no reason to keep the light on, even if you are in the room (signified by the door being unlocked). Secondly, we are also controlling the heater by sensing the temperature. If the temperature crosses the threshold of 69℉, then a servo will push the power button on the AC to turn it on or off. Originally, we were a team of four, so we had to split the project into two to form two teams of two. The half that we ended up with was everything except for the door locking and unlocking. So, we still have the RFID sensing, light sensing, and heat sensing, along with the corresponding actions besides the door locking. At the beginning of the class, we would have liked to set this up in one of our dorms, but we were not able to do this properly because we would have needed really long wires to reach the distance from the light switch to the AC window unit.

The first sensor we are using is the RFID sensor. This uses SPI to communicate with the Arduino, so there was little in the way of characterization we could do. We did find a library on GitHub by miguelbalboa to use for the project. All I had to do was to learn how to capture a specific ID number off of an RFID card, so that way only one card can lock/unlock the room, and the other one will have no effect. We have this demonstrated by red and green led, whether the door is locked or unlocked because that was not part of the project. The next sensor we had to

use was the photoresistor. To characterize this, I plugged it into the Arduino, put it in series with a 10K resistor to read the data from it. We had already characterized this in the regular 110 labs, so I knew a bit about how to work with it. Then, I printed the values to the serial monitor and discovered the average of the values of light and dark for the 110 lab room was about 690, so I make this my threshold value. Finally, I had to characterize the heat sensor. I knew that analog values on the Arduino came in with values from 0-1023, and according to the datasheet of the sensor, the range of values it could read was -55 to 150℉, so I wrote a method to convert the incoming data to Fahrenheit. I do not know how accurate this, as I do not have anything else that can read temperature, but the few times I tested it out, the values ranges from 67-70℉, which is about room temperature, so I ruled this ok and moved on. One thing that affected our design was the size of the RFID sensor. This was very large, which made organization very hard, as it took up much more space than everything else on the breadboard. I solved this by splitting up everything on to two breadboards, one with all of the analog inputs, and one with all of the things that needed to connect to the digital pins. This allowed me to put the digital breadboard on one side of the Arduino, and the analog breadboard on the other side, so I would only have to run one power line across the Arduino, as opposed to many signal lines without this system of organization. I also was not actually able to implement this in my dorm because I did not acquire long enough wires to span the room, nor did I have the organizational skills necessary to do this. In order to accomplish this, I would use two ESP32's to communicate with each other across the room, because they have a Bluetooth connection, so it is possible to use wireless communication with them.

In the construction of the project, I wanted to 3D print the parts to make this work, but sadly I did not get far enough on that quick enough to actually 3D print the parts. I do have CAD models of some of the parts that are included in the footnotes. Additionally, I have some schematics, the Arduino code, pictures of the product, and the block diagram. One of the challenges I had in doing this all was how little time there was. At the beginning of the class it felt like there was plenty of time, but nearing the end, it felt like time went by faster, and with only one person, it got caught up with me, and there were several things I did not finish, including the CAD models, and switching to the ESP32.

Along the way through this project, I was able to develop my skills in Arduino programming, and how to use sensors. One of the problems I had with the temperature sensor was that it jumps around a lot in values, so what I did to fix this was to take the past three values, and then average that number, to get the final temperature that I would actually check against the threshold. I also learned about communication skills with my lab partner, and how to manage a long-term project such as this one. In the beginning, it was hard to know all of the parts we would need, and I definitely ordered some of the wrong things and forgot to order some cables. I think that I assumed I would be able to find these things in the lab, but I should have just ordered everything I needed for the project so it could be one contained unit. I think that this project was a good prototype of what we wanted to do at the beginning of the class, but ultimately it is nothing more than a prototype. Ideally, it would be made out of plastic, and have soldered connections, but this worked for the time being.

Appendix

Arduino Code:

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

#define SS_PIN 10
#define RST_PIN 9
#define UNLOCKED_LED_PIN 4
#define LOCKED_LED_PIN 5
#define BAD_LED_PIN 2
#define LIGHT_SERVO_PIN 3
#define HEATER_SERVO_PIN 6
#define LIGHT_SENSOR_PIN A0
#define THERMO_SENSOR_PIN A1

MFRC522 rfid(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;

// Init array that will store new NUID
byte nuidPICC[4];

bool goodCardFound;
bool locked;

Servo lightSwitchServo;
Servo heaterServo;
int onLightAngle = 10;
int offLightAngle = 0;
int onHeatAngle = 30;
int offHeatAngle = 60;


float thresholdHeat = 65;
float thresholdLight = 480;
float lightValue = thresholdLight;
float heatValue = thresholdHeat;
float previousHeat = thresholdHeat;
float firstHeat = thresholdHeat;
float secondHeat = firstHeat;
float thirdHeat = secondHeat;
```

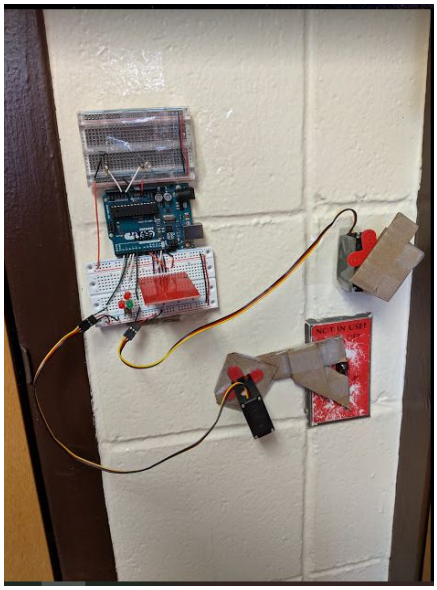
```
void setup() {
  pinMode(UNLOCKED_LED_PIN, OUTPUT);
  pinMode(LOCKED_LED_PIN, OUTPUT);
  pinMode(BAD_LED_PIN, OUTPUT);

  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }

  Serial.println(F("This code scan the MIFARE Classsic NUID."));
  Serial.print(F("Using the following key:"));
  printHex(key.keyByte, MFRC522::MF_KEY_SIZE);

  lightSwitchServo.attach(LIGHT_SERVO_PIN);
  heaterServo.attach(HEATER_SERVO_PIN);
}

void loop() {
  // before we do any sort of rfid thing, we need to do some data gathering
  lightValue = analogRead(LIGHT_SENSOR_PIN);
  previousHeat = heatValue;
  heatValue = updateHeat(scaleToFarenheit(analogRead(THERMO_SENSOR_PIN)));
  Serial.println(heatValue);
  //deal with the light value only if door is not locked
  if(!locked) {
   lightValue > thresholdLight ? lightSwitchServo.write(offLightAngle) :
lightSwitchServo.write(onLightAngle);
   (heatValue > thresholdHeat && previousHeat > thresholdHeat) ||
   (heatValue < thresholdHeat && previousHeat < thresholdHeat) ? delay(0):
pushButton(); //if the heat didnt cross the line, dont do anything
  }

  // Look for new cards
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
    return;

  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
```

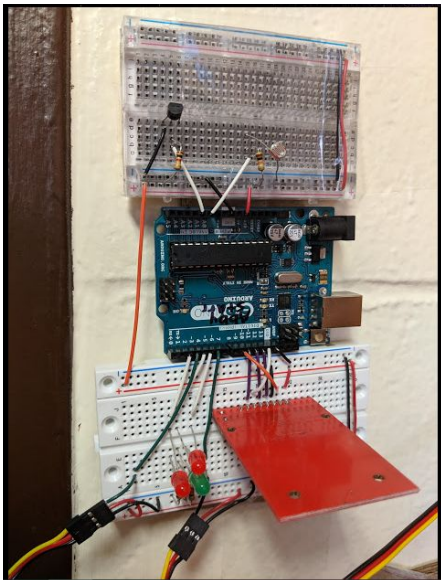
```
  Serial.println(rfid.PICC_GetTypeName(piccType));

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic."));
    return;
  }

  if (!goodCardFound) {
    goodCardFound = true;
    Serial.println(F("A new card has been detected."));

    // Store NUID into nuidPICC array
    storeUUID();

    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    printHex(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
    Serial.print(F("In dec: "));
    printDec(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();

  } else if(rfid.uid.uidByte[0] != nuidPICC[0] ||
    rfid.uid.uidByte[1] != nuidPICC[1] ||
    rfid.uid.uidByte[2] != nuidPICC[2] ||
    rfid.uid.uidByte[3] != nuidPICC[3]) { //wrong card scanned
    Serial.println(F("Card WRONG."));
    digitalWrite(BAD_LED_PIN, HIGH);
  } else {
    Serial.println(F("Correct Card"));
    locked ? lightSwitchServo.write(offLightAngle) :
lightSwitchServo.write(onLightAngle);
    locked = !locked;
    digitalWrite(BAD_LED_PIN, LOW);
    if(locked) {
      digitalWrite(UNLOCKED_LED_PIN, LOW);
      digitalWrite(LOCKED_LED_PIN, HIGH);
    } else {
      digitalWrite(UNLOCKED_LED_PIN, HIGH);
      digitalWrite(LOCKED_LED_PIN, LOW);
    }
  }
```

```
  // Halt PICC
  rfid.PICC_HaltA();

  // Stop encryption on PCD
  rfid.PCD_StopCrypto1();
}

/**
 * read a uuid to a byte array
 */
void storeUUID() {
  for (byte i = 0; i < 4; i++) {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }
}

/**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}

/**
 * Helper routine to dump a byte array as dec values to Serial.
 */
void printDec(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], DEC);
  }
}

float scaleToFarenheit(int rawValue) { //starts at 0-1024
  float temp = float(rawValue);
  temp *= 206;
  temp /= 1024;
  temp += -55; //goes from -55-155
  return temp;
}

void pushButton() {
 heaterServo.write(onHeatAngle);
```

```
 delay(500);
 heaterServo.write(offHeatAngle);
}

float updateHeat(float newValue) {
  //take the average of the last three inputs
  thirdHeat = secondHeat;
  secondHeat = firstHeat;
  firstHeat = newValue;

  return (firstHeat + secondHeat + thirdHeat) / 3;
}
```

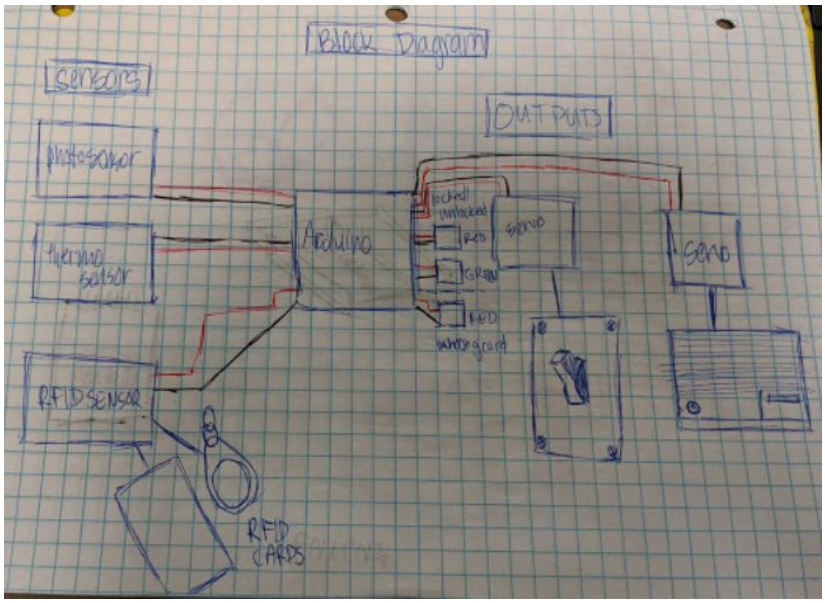
Pictures of actual product:





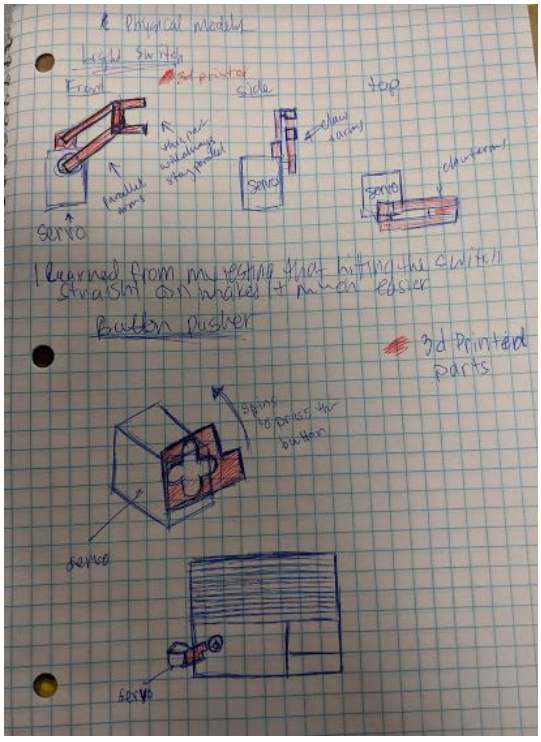
Pictured on the left is the whole system attached to the wall. I did not make the servo mounts out of cardboard, so that is why the servo is backward. One thing that is wrong about the cardboard diagram is that the part that actually flips is shaped like a y, with one part slanty. The slanty part does less work on the light switch because not all of the force is directed onto the switch. In the actual one, I would like to have two arms with a U-shape switcher, so that way the force is equal on the way up and down.

Block Diagram:



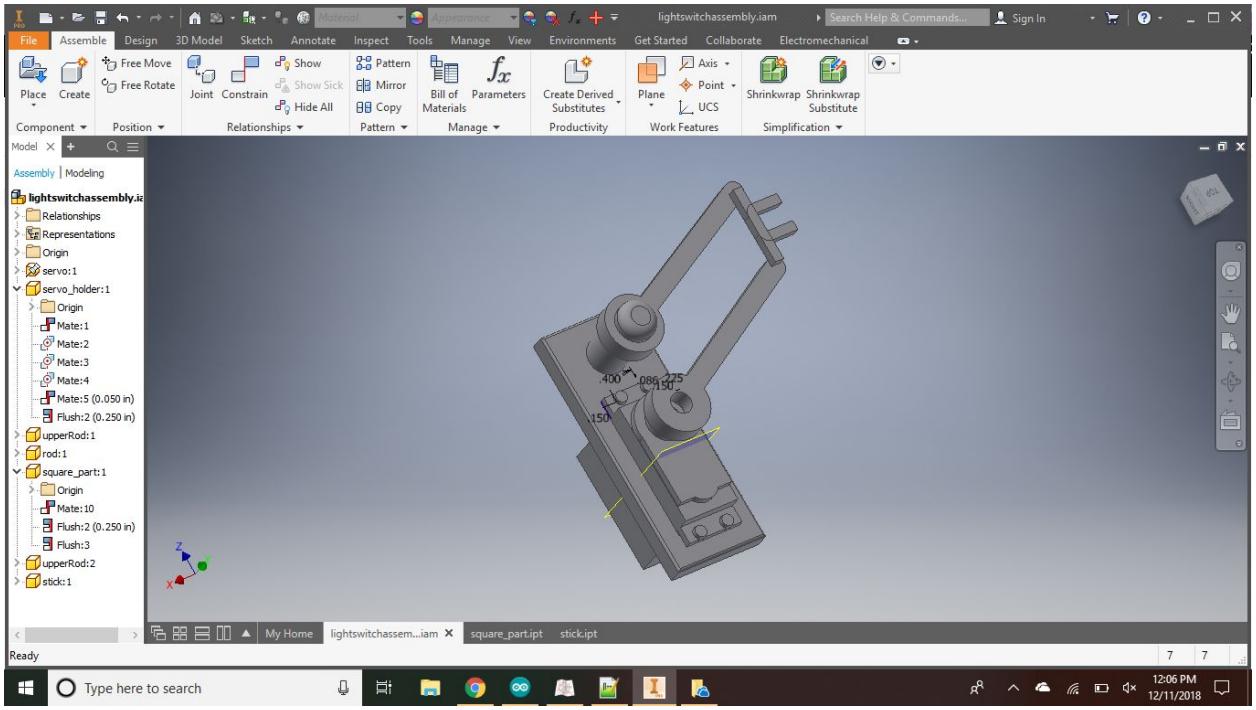
This is the block diagram. As you can see, all of the inputs and outputs are connected to the Arduino. Included in the block diagram is also how each of the parts is connected to the real world, represented by the blue line instead of the red and black lines.

Model Sketches:



On the top is the light switch servo, which I have mostly done in CAD, but not to the point where it is able to be 3D printed. On the bottom is the AC button pusher, which I just have made out of cardboard.

Cad Model:



The dimensions are not quite right, but this is approximately what I would like to do. For the button pusher, I would use the same servo mount but attach something different to the head.