Honor Project Final Report

Project: Intelligent self-heat pocket

Yuheng Chang(yuhengc2), Yingtong Hu(yh15)

1. Introduction

a. Statement of Purpose

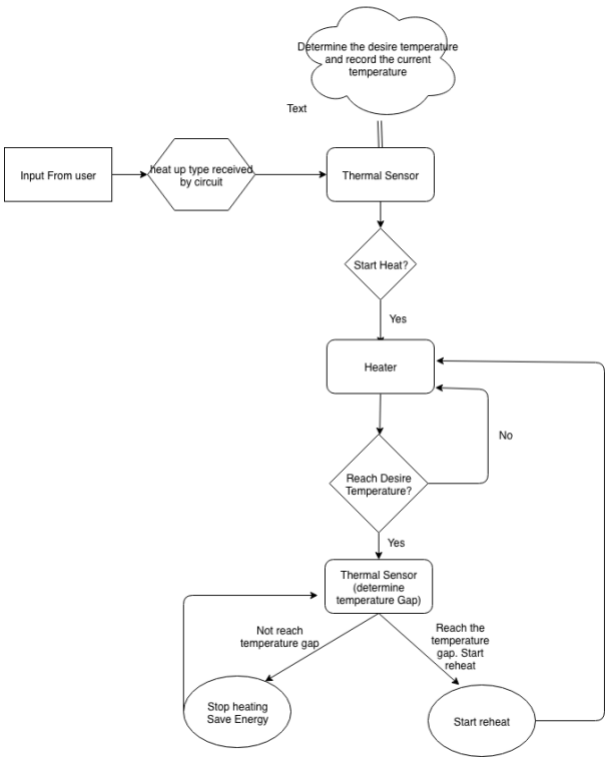
Our project is to build a self-heat pocket to keep hands on the desire temperature automatically or manually. The pocket automatically keep you hand warm by sensing the current temperature and based on the reading, the pocket makes the decision to begin to heat or not. The heating condition can also be changed manually by waving hand in front of the ultrasonic sensor to keep you pocket in your own desire temperature.

b. Features and Benefits

The main feature of our project is we give the pocket the ability to behavior differently based on users' personal desire by adding the manually changing option. In the manual mode, changing between heating or not heating can be easily done by waving hand in front of the sensor. Switching between automatically and manually is also relatively easy to achieve by turning the knob of the potentiometer. We try to give user more control of our project and we also add protection to user's hand if the temperature is too hot.

2. Design

a. System Overview



The manual mode is basically the same except changing state is all controlled by user.

b. *Design Details*

We use the ultrasonic sensor to read the input from the user by measuring the distance between the hand and the sensor. The ultrasonic sensor:

It has four pins and the ground pin and V_{cc} pin are used to power the sensor and the require power voltage 5V. For each $10\mu s$, the trigger pin is set to high and after the trigger pin change from high to low, the sensor will send out 8 sonic waves travelling at the speed of the sound. At the same time, the echo pin started to be on high stage. Once the sonic waves hit certain object and bounced back, the echo pin will change back to low stage. We can then measure the distance between the object (in our project, the human hand) by applying following formula.

$$distance = \frac{t_{echo} \times V_{speed\ of\ sound}}{2}$$

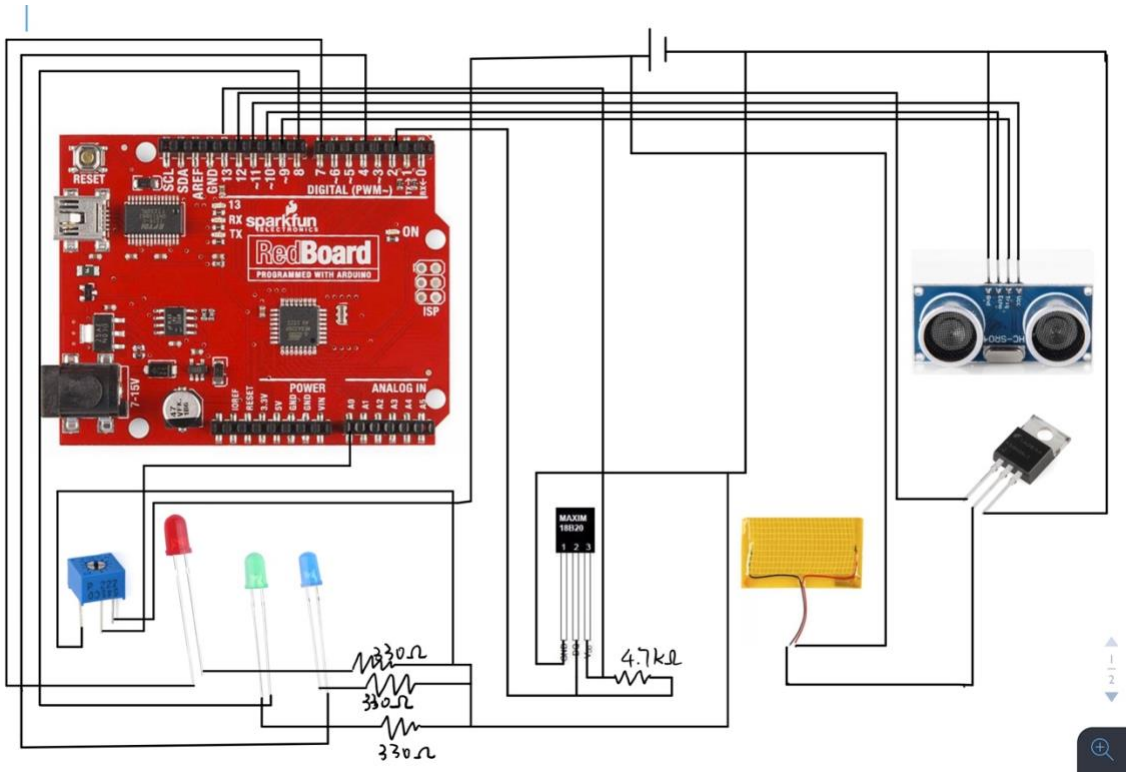
t_{echo} is the time duration the echo pin is at high stage. Since those sonic waves are travelling at speed of sound, the time duration multiply with speed of sound gives us the overall distance those sonic waves travel. Since those waves travel towards object and bounce back from the object, we need to divide the distance by two to get the one-way distance.

We also use a temperature sensor to keep track of current temperature of the environment. We use DS18B20. It changes the output voltage based on current temperature and we utilize the Arduino and external library (1-Wire Bus and Dallas Temperature) to convert the analog signal from the sensor to the digital value. We use the Dallas Temperature library to translate the digital temperature value into the Celsius value and based on the Celsius value, we change the states of our self-heat pocket accordingly.

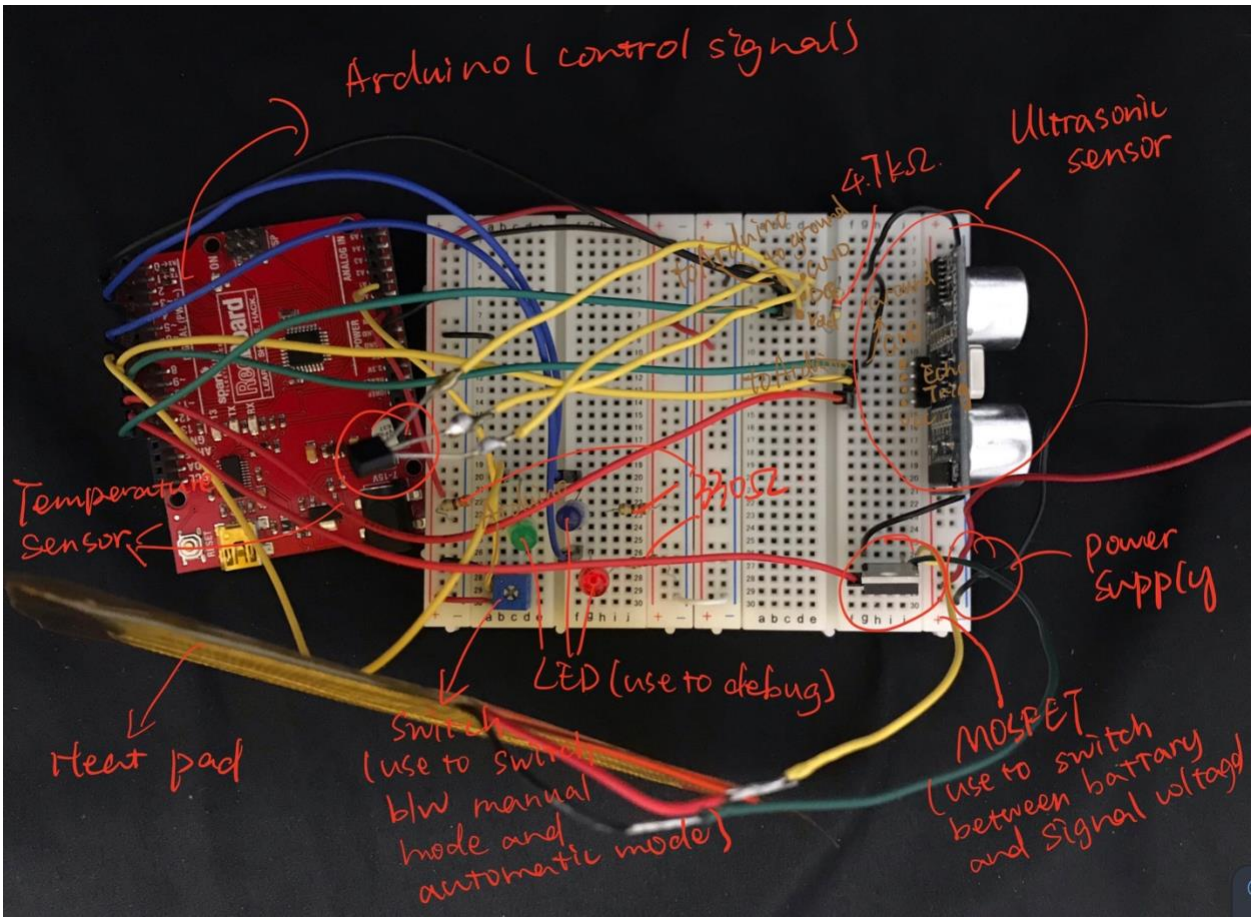
For the heating part, we use the heap pad from Sparkfun (COM-11288), which require $5V$ DC voltage and $750mA$ current to power on. However, we discover that to feel the temperature quickly and effectively, we need to power the heat pad with $9V$ DC voltage and the Arduino can only output $5V$. Thus, we use the N-Channel MOSFET combined with additional power supply. We use output from Arduino to control gate value of the MOSFET.

Besides, we use a potentiometer to switch between manual mode and automatic mode of our self-heat pocket. The potentiometer is basically a voltage divider and we read the voltage value from the resistor value of R_1, the left resistor of the potentiometer, by Arduino. If the value is greater than 512, it's in manual mode and if its value is less than 512, it's in automatic mode.

c. Circuit Schematics



d. Circuit Configuration



3. **Results**

We use two sensor temperature sensor and ultrasonic sensor. For the temperature sensor, the reading value after translation is very accurate compare the reading from the real thermometer. For the ultrasonic sensor, it's very sensitive to surrounding and can react pretty quickly with my hand movement.

4. **Problems and Challenges**

One of the challenges we confront is that the temperature is so sensitive that each time I put my hand in front of it, it will read multiple value below the threshold we set. So, I will change several states at a time rather than just change one state. To fix this, I add a buffer towards ultrasonic sensor in our code to make it only read value per 0.5s. It can approximately buffer the time of hand movement and provide relatively desired outputs.

Another challenge is that this project requires several states interact with each other (two state in automatic mode and four state in manual mode). We need very careful design about how to change from one state to another without interfere others and try to minimize the corner cases which might cause unexpected output. We put many "if else" statements in our Arduino code and add several buffers to smooth the interchange between different states.

5. **Future Plans**

Our current project has very limited portability since all the circuit is on the breadboard and the overall size is large for this wearable project. First thing we want to optimize is make our circuit more concise and solder the

wire in a PCB to minimize the size and make the circuit less vulnerable. We also need to weave this circuit into a hoodie to create an ultimate integrated product.

Another optimization we want to make in the future is that we want to add more protection in the manual mode. Sometimes, the users might forget to change the state so that the heat pad is constantly heating, and the temperature might get too high to harm the user. We might add several LED or warning tone to remind the user change the state or we change the state automatically.

6. References

Nedelkovski, D. (2015). *Ultrasonic Sensor HC-SR04 and Arduino Tutorial*. [online] HowToMechatronics. Available at: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ [Accessed 14 Dec. 2018]

Konstantin Dimitrov, K. (2016). *DS18B20 (digital temperature sensor) and Arduino*. [online] Arduino Project Hub. Available at: https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-9cc806#toc-step-3--libraries-2 [Accessed 14 Dec. 2018].

7. Appendix

Arduino Code:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const int triggerPin = 9;
const int echoPin = 10;
const int ultrasonicSource = 11;
const int temperaturePin = 12;
const int switchReadPin = A0;
const int LEDGreenPin = 8;
const int LEDRedPin = 7;
const int temperaturePower = 13;
const int LEDBluePin = 4;


long duration;
int distance;
float switchVal;
bool startHeating;
bool startKeepWarm;
bool sleepState;
bool warmInitial;
unsigned long time;
unsigned long lastTime;
unsigned long sleepTime;
unsigned long warmTime;
```

```
unsigned long sleepTrackTime;
unsigned long currentTemperature;

void setup() {
    time = 0;
    lastTime = 0;
    sleepTime = 0;
    warmTime = 0;
    switchVal = 0.0f;
    currentTemperature = 0;
    startHeating = false;
    warmInitial = true;

    pinMode(switchReadPin, INPUT);
    pinMode(echoPin, INPUT);
    pinMode(triggerPin, OUTPUT);
    pinMode(ultrasonicSource, OUTPUT);
    pinMode(temperaturePin, OUTPUT);
    pinMode(LEDGreenPin, OUTPUT);
    pinMode(LEDRedPin, OUTPUT);
    pinMode(LEDBluePin, OUTPUT);
    pinMode(temperaturePower, OUTPUT);

    Serial.begin(9600);
    sensors.begin();
}
```

```
void loop() {

    switchVal = analogRead(switchReadPin);
    Serial.print("switchVal: "); Serial.println(switchVal);

    digitalWrite(temperaturePower, HIGH);
    sensors.requestTemperatures();
    currentTemperature = sensors.getTempCByIndex(0);
    Serial.print("current temperature: ");Serial.println(currentTemperature);

    time = millis();
    digitalWrite(ultrasonicSource, HIGH);
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance= duration*0.034/2;
    //Serial.print("distance: ");Serial.println(distance);

    if (switchVal > 512.0) {
        sleepTrackTime = 0.0;


       if (distance < 8 && time - lastTime > 800) {
           lastTime = time;
           if (!startHeating && !startKeepWarm && !sleepState) {
               startHeating = true;
           } else if (startHeating && !startKeepWarm && !sleepState) {
               startHeating = false;
               startKeepWarm = true;
               warmTime = time;
               warmInitial = true;
           } else if (!startHeating && startKeepWarm && !sleepState) {
               startKeepWarm = false;
               sleepTime = time;
               sleepState = true;
           } else if (!startHeating && !startKeepWarm && sleepState && time - sleepTime > 5000) {
               sleepState = false;
               startHeating = true;
           } else {
               sleepState = false;
           }
       }

    } else {
         if (startKeepWarm && sleepTrackTime == 0.0) {
             sleepTrackTime = millis();
         }
```

```
            if (currentTemperature < 27.0) {
                startHeating = true;
                sleepTrackTime = 0.0;
                startKeepWarm = false;
                sleepState = false;
            }
            if (currentTemperature >= 27.0 && currentTemperature <= 34.0) {
                startHeating = false;
                startKeepWarm = true;
                sleepState = false;
            }
//            if (millis() - sleepTrackTime >= 5000.0 && sleepTrackTime != 0.0) {
//                Serial.print("sleepTrackTime: ");Serial.println(sleepTrackTime);
//                startHeating = false;
//                startKeepWarm = false;
//                sleepState = true;
//            }
        }

        if (!startHeating && !startKeepWarm && !sleepState) {
           // Serial.println("I am shutted down");
            digitalWrite(LEDGreenPin, LOW);
            digitalWrite(LEDBluePin, LOW);
            digitalWrite(LEDRedPin, LOW);
        }

         if (startHeating) {
            //Serial.println("in start heating stage");
            digitalWrite(temperaturePin, HIGH);
            digitalWrite(LEDGreenPin, HIGH);
            digitalWrite(LEDBluePin, LOW);
            digitalWrite(LEDRedPin, LOW);
         }

         if (startKeepWarm) {
            //Serial.println("in keep warm stage");
//            if (time - warmTime <= 5000) {
//                warmInitial = true;
//                digitalWrite(temperaturePin, HIGH);
//            } else {
//                if (warmInitial) {
//                    warmTime = time;
//                    warmTime = false;
//                }
//                digitalWrite(temperaturePin, LOW);
//            }
            if (currentTemperature < 20.0) {
                startHeating = true;
                //sleepTrackTime = 0.0;
                startKeepWarm = false;
                sleepState = false;
            }
            digitalWrite(temperaturePin, LOW);
            digitalWrite(LEDGreenPin, LOW);
            digitalWrite(LEDBluePin, HIGH);
            digitalWrite(LEDRedPin, LOW);
        }

        if (sleepState) {
            digitalWrite(LEDGreenPin, LOW);
            digitalWrite(LEDBluePin, LOW);
            digitalWrite(LEDRedPin, HIGH);
            digitalWrite(temperaturePin, LOW);
        }
}
```