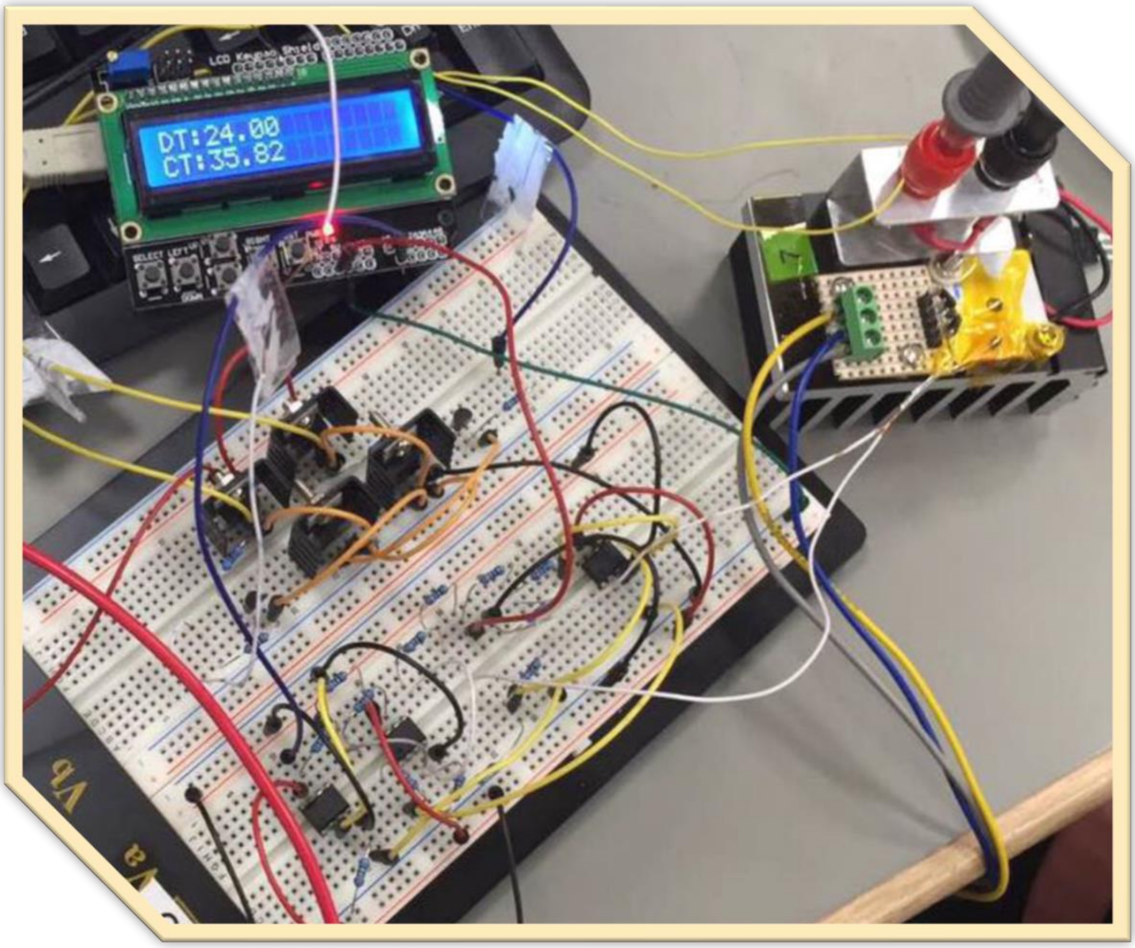
ELEN 30013 ESI REPORT

Project of temperature controller



Yue Chang 872301

Yi Jian 850509

Workshop time: Wednesday 3:15-6:15
Group NO.7

20.10.2017

EXECUTIVE SUMMARY

The project involves the design, construction, testing and documentation of circuits and microprocessor code to control and display the temperature of the Aluminium block to a user settable temperature in the range 5°C ≤ T ≤ 45°C. The temperature is controlled to within +/- 0.5°C of the set temperature.

In this system, the current temperature is monitored every 0.5s by a 10K NTC thermistor in the Wheatstone bridge together with the instrumental amplifier circuit. A LCD with keypads is used for input and display temperature. Thermo-electric cooler (TEC) is the device that can heat or cool due to the direction of current. H-bridge circuit is used to drive TEC and PWM output from Arduino is to control the heat up or cool down speed of the system. All the instructions are accomplished by using Arduino UNO.

The performance of the system can be listed as below.

- Range of temperature control: 0~45℃
- Temperature accuracy: +/- 0.5℃(When the difference between the desired temperature and the room temperature is large, the temperature accuracy may decrease)
- Temperature settling time: less than 2 mins
- Power consumption： Sensor circuit: 0.65w Driving circuit: 4.7w TEC: 1.5w
- Energy utilization: 28%

CONTENTS

1. Introduction

1.1 Design Concept

The temperature controller in the project uses NTC thermistor as the sensor to monitor the current temperature (CT) in real time, use TEC (Thermo-electric cooler) to cool down or heat up an Aluminium block to a user settable temperature in the range 5°C ≤ T ≤ 45°C. The temperature can be controlled to within +/- 0.5 °C. Arduino is used as a microprocessor to control all the circuits. The system also includes a LCD screen that can display the current and desired temperature (DT).

- For the sensor circuit, we use the Wheatstone bridge to measure the thermistor resistance together with instrumental amplifier to amplifier the voltage to a proper range of 0-5v.
- For the driving circuit, we use h-bridge as the switch to control the current direction through TEC and use Arduino's PWM output to give TEC different size of current to achieve a more precise temperature control.
- For the display and input part, we use the LCD with keypads, which let the user to input their desired temperature on the keypads and the current temperature (CT) and desired temperature (DT) will show on the LCD screen.

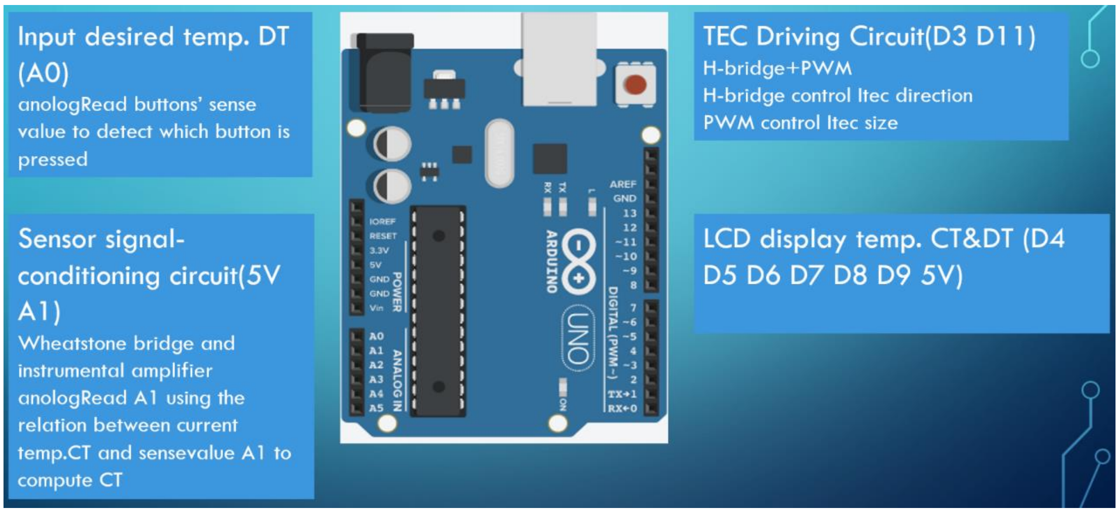

Input desired temp. DT (A0)
anologRead buttons' sense value to detect which button is pressed
Sensor signal-conditioning circuit(5V A1)
Wheatstone bridge and instrumental amplifier
anologRead A1 using the relation between current temp.CT and sensevalue A1 to compute CT
TEC Driving Circuit(D3 D11)
H-bridge+PWM
H-bridge control Itec direction
PWM control Itec size
LCD display temp. CT&DT (D4 D5 D6 D7 D8 D9 5V)
ARDUINO
UNO

figure 1-1 System Block Diagram

1.2 Components Count

table 1-1 Components

Name	Quantity
Power supply	1
AD590	1
10K NTC thermistor	1
TEC	1
Arduino	1
LCD1602 (purchased from Jaycar, ＄19.95)	1
Transistors	Tip41c×2 Tip42c×2 BC548B×2
LM741	3
Resistors	1Ω×1 270Ω×2 1kΩ×4 10kΩ×11 14.1kΩ×1 27kΩ×1
Breadboard	1
Wires	A few

2. Time management

Week	Free time	Workshop time
6	Determine project requirements r eview lecture contents(together)	Search for information(together)
7	Design the system overall frame work List time table(together)	Learning principles of sensor and driving circuit discuss and choose method(togerth er)
8	Work distribution(together)	Learn and start to purchase LCD and butt ons for display and input temp.(YI) Start simulation of H-bridge and Sensor circuit.(YUE)
9	Discuss and error check and mod ification of the simulation of h-bridge and sensor(together)	Build the physical circuit for h-bridge(YI) Error check and lab records.(YUE)
Non-teachi ng pe riod	1.Running simulation for LCD an d buttons(YI) 2.Arduino code for display temp. and input temp. parts(YI)	1.Modify simulation of h-bridge and sensor circuit(YUE) 2.Arduino code for sensor reading calibrat ion and h-bridge control(YUE)
10	Code and simulation errors check Experiments contents arrangement (together)	Open lab(Mon): Another physical circuit for h-bridge(YI),error check and lab records(YUE) Workshop: Physical circuits for h-bridge and sensor circuits(YI), error check and lab records(YUE) Open lab(Wen):Test Arduino code for h-bridge, change PWM(YUE) Record Itec and PWM for h-bridge(YI) Open lab(Fri):Modifications for Sensor circuits and test Vout(<5v) for different RT(YUE) Test LCD display and buttons for input temp.(YI)
11	Organize the code and prepare for project check(together)	Workshop: put all parts together and error check,calibration(together) Open lab(Fri):Physical test, calibration, improvements plans(together) Open lab(Mon):Test improvements plans, come to conclusion(together)

3. Methodology

3.1 Sensor Circuit

3.1.1 Wheatstone Bridge and Instrumental Amplifier

- Wheatstone Bridge: Usually been used as a more precise way to compute an unknown resistance like the resistance of the thermistor by measuring the voltage difference between V_A and V_b, for the reason that, noise from power supply V_0 is cancelled in A node and B node. (The schematic is as follows, where R_T is the thermistor)

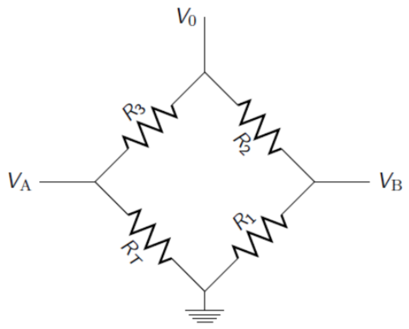


figure 3-1 Wheatstone Bridge

- Instrumental Amplifier： A suitable choice for amplifying small signal with high input impendence. Op-amp $A_1$and A_2 are buffers that can isolate input and output circuit, which gives it high level of noise injection. Resistance difference caused by the resistance tolerance can also be cancelled in the two inputs. We use it to amplifier V_A-V_b so that, V_{out} distributed in the range of 0-5V which can full utilize the Arduino analogRead range (0-1023) to enhance sensor accuracy. The schematic is as follows.

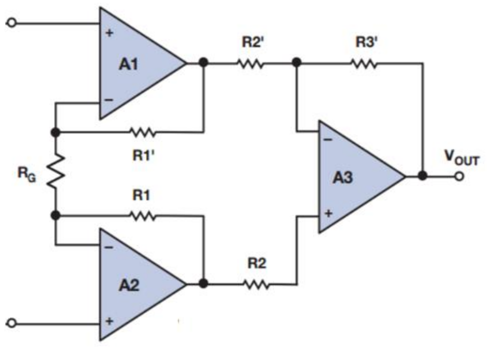


figure 3-2 Instrumental Amplifier

3.1.1 Simulation

The simulation circuit diagram of our sensor circuit is as figure 3-3.

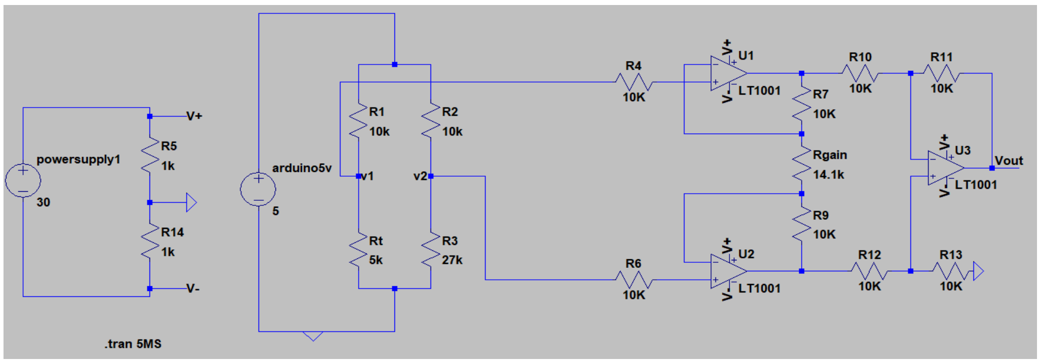


figure 3-3 Simulation of Wheatstone bridge and Instrumental Amplifier

Here we need to do some computation to figure out the resistance of R_3 and R_{gain}.

- R_3 : Because Arduino can only read positive voltage which means that V_{out} has to be positive all the time, so that, $V_1 > V_2$ that is $R_3 > R_{Tmax}$. In our case we choose R_3=27kΩ> R (5°C)=25.5kΩ
- R_{gain}: We want the voltage difference (V_2-V_1)< 5v after amplified Arduino to read. Using the relationship that Gain=1+20k/R_{gain} we can compute that R_{gain}=14.1kΩ, we choose 14kΩ in our physical circuit.

By simulation, we can get the V_{out} range is 0.19 ~ 4.79v, which is not only fully distributed in the range of 0~5v but also small enough for Arduino to read.

3.2 Driving Circuit

3.2.1 H Bridge

H-bridge is used as a switch to control TEC. We choose this type of the six transistors h-bridge to drive out TEC. The outer two NPN transistors are signal transistors, which can have high current gain for small current, the other inner two NPN and two PNP transistors are power transistors which can let larger current going through but gain is not that big. When apply an input at Heat, the current through the middle resistor can only flow from left to right. If we substitute the middle resistor with TEC, we can control the cool/heat status of it.

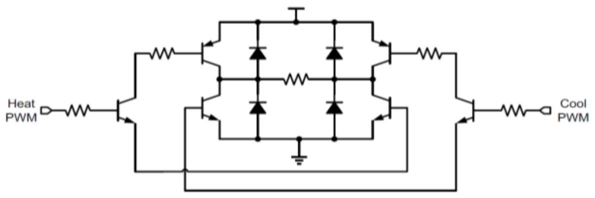


figure 3-4 H Bridge

3.2.2 Simulation

The simulation circuit diagram of our driving circuit is as figure 3-5.

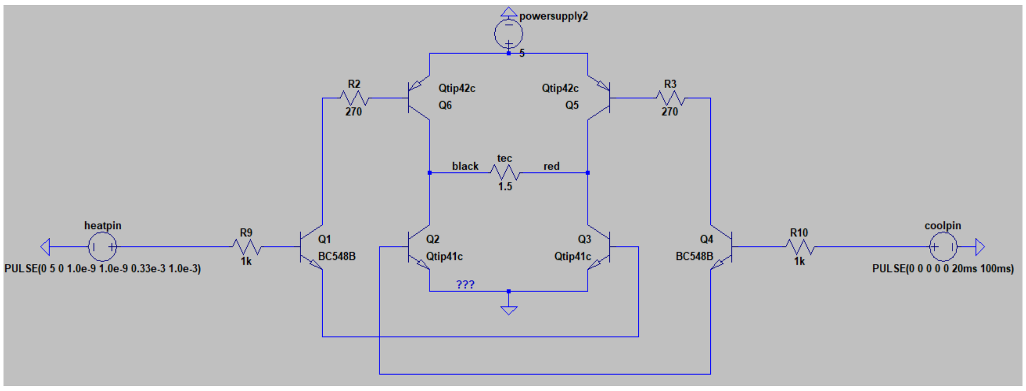


figure 3-5 Simulation of H Bridge

Because we want to use PWM output of Arduino to get different size of current to control TEC, we want one of our three transistors to work in linear region, through simulation, we get the resistance for$R_2$and$R_9$to be 270Ω and1kΩ respectively.

By measurement, the maximum simulated current through TEC is 932mA, which doesn't exceed the limited current through TEC.

The working status of all the transistors is as shown in the table below.

table 3-1 working status of all the transistors

Status	**ON**	**OFF**
Heat	Q6(Saturation)Q1(Saturation)Q3(Linear)	Q5 Q2 Q4
Cool	Q5(Saturation)Q2(Saturation)Q4(Linear)	Q6 Q1 Q3

3.3 Temperature Setting and Display

We use the LCD1602 with buttons for our temperature setting and display, comes with a built-in 16 character by 2 line LCD display with adjustable backlight, this six push button keypad allows us to create a user friendly interface for our project. The keypad consists of up, down, left, right, select and also a reset button. To initalise the shield for use with the Liquid Crystal Library use the Liquid Crystal LCD(8,9,4,5,6,7). When a button is pushed down, it will produce an analog voltage to Arduino Pin A0. We need to double check to make sure that no buttons have been accidentally touched. Then Arduino read the analog value from A0 to determine

table3-3 pins of LCD1602

which button is pushed down. In our project, when the LEFT button is pushed, the desired temperature will minus 10; when the RIGHT button is pushed, the desired temperature will plus 10; when the UP button is pushed, the desired temperature will plus 0.5; when the DOWN button is pushed, the desired temperature will minus 0.5.

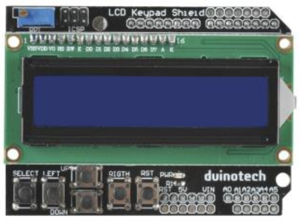


figure3-6 LCD1602 with buttons

table3-2 some specifications of LCD1602

Specifications	
Operating Voltage	5VDC
PushsButton Protocol	Analog

Pinout	
D9	E
D*	RS
D7	DB7
D6	DB6
D5	DB5
D4	DB4
A0	Buttons

3.4 Calibration

If we want to measure temperate, according to the sensor circuit design, we can get that CT→RT→Vout→analogRead A1→SenseValue. So to do calibration, we only need to find the relationship SenseValue and CT. We used AD590 as the reference of current temperature, by capturing the voltages of V_{out} and R_1 using Arduino we can compute the function representing CT using V_{out}. The senseValue range is 136~964 through measurements. The circuit used for calibration and the resulting curve fitting using MATLAB are as below.

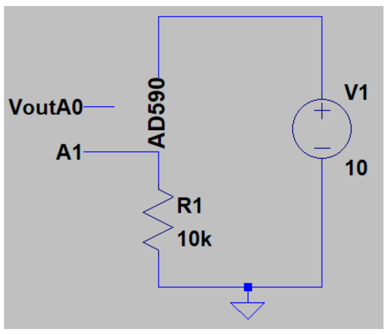


Figure3-7 circuit for calibration

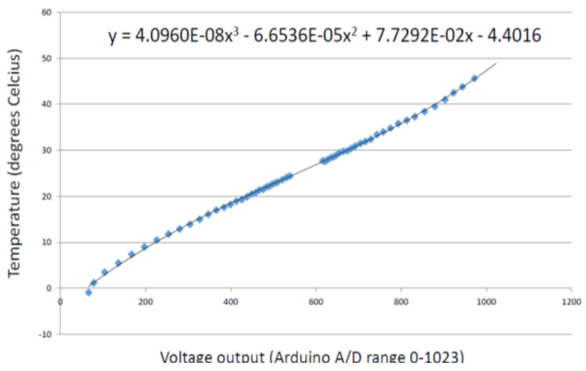


Figure3-8 calibration curve

3.5 Arduino Code Design

3.5.1 Sensor signal conditioning

Use Arduino A1 pin to read V_{out} of the sensor circuit then use the relationship of CT and senseValue from calibration to monitor the current temperature at intervals of 500ms.

3.5.2 H-bridge control

We use Arduino's PWM output to control H-bridge. There are two factors we need to take into consideration.

①DT-CT

For large temperature difference, larger current (big PWM) needed to speed up the heating or cooling process.

For small temperature difference, smaller current (small PWM) needed to keep steady.

②DT-25

Here, we assume room temp=25°, so that we can compute the absolute value.

When DT>>room temperature, we need larger current (bigger PWM) to reach DT or to say maintain a high/low temperature.

When DT is around room temperature, we need a small current (smaller PWM) to avoid over heat or under cool.

Through several experimental experience, function between DT-25 and PWM output is estimated as: F(x) = -0.243*x*x+10.31*x+11.260504201680671

The flow chart for H-bridge control is as below FIGURE.

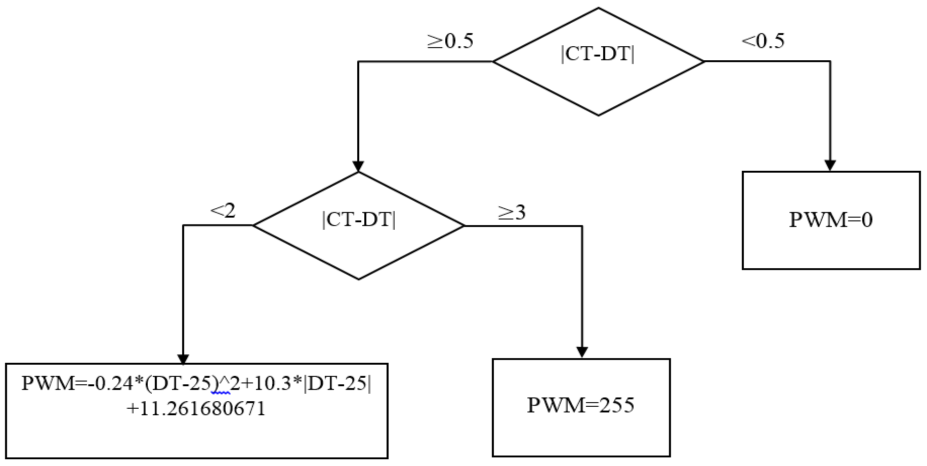


figure 3-9 Flow chart for H bridge control

3.5.3 Temperature Setting and Display

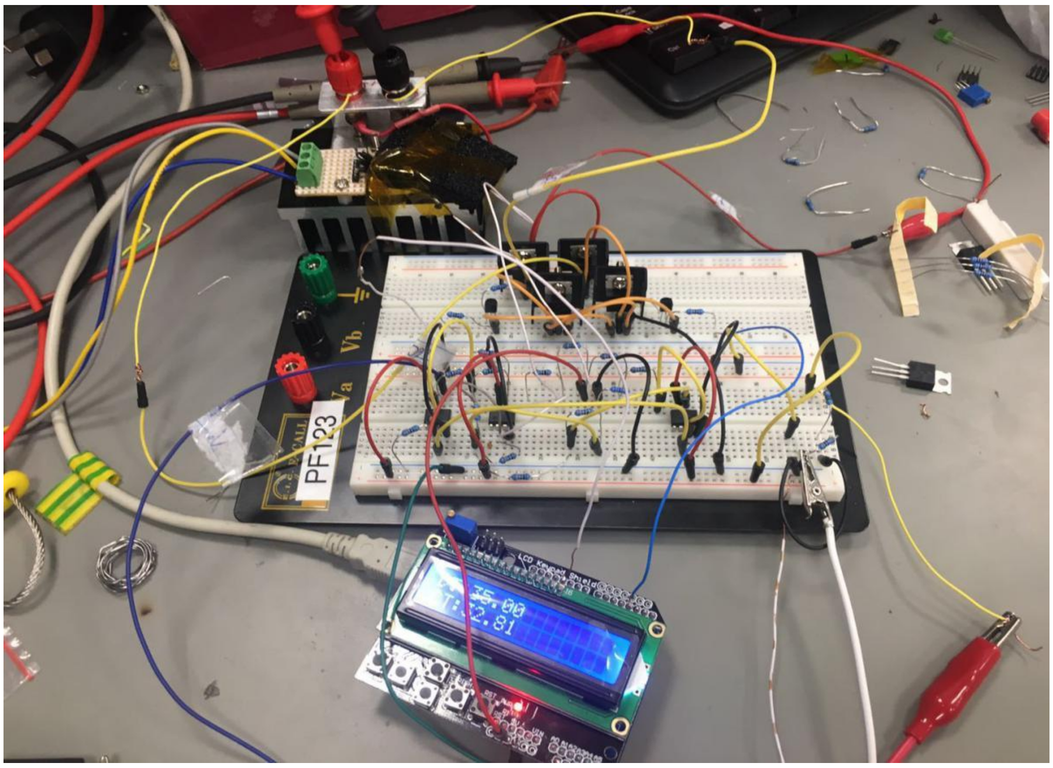
The code for temperature setting and display is based on the LCD1602, including two sub-routine (read and do). In the main program, first calling subroutine-read, to determine which key is pressed down and whether the button is pressed once or being held pressed. Then calling subroutine-do to make DT change. If the button is still held pressed, repeat the change. Finally, display contents on the LCD, the first row is DT and the second row is CT, show two digits after the decimal point.

4. Result and Conclusion

4.1 System performance

- Temperature settling time: less than 2 mins
- Temperature accuracy: +/- 0.5℃(When the difference between the desired temperature and the room temperature is large, the temperature accuracy may decrease)
- Power consumption： Sensor circuit: 0.65w Driving circuit: 4.7w TEC: 1.5w
- Energy utilization: 28%

Physical Circuit Picture:



4.2 Limitations

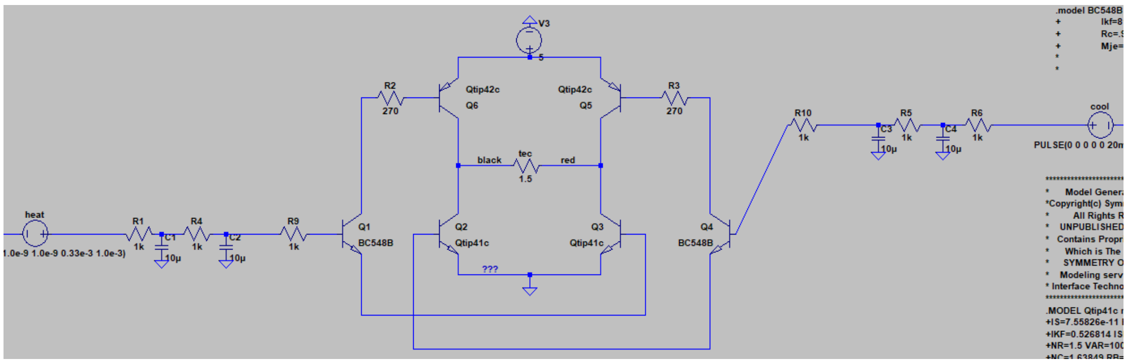
- The transistors of H bridge can easily to get hot because of high power consumption.

- When the difference between the set temperature and the room temperature is large, the temperature accuracy may decrease, due to it's hard using TEC to keep high or low temperature against room temperature.
- The circuit may not work because of poor wire contact.

5. Improvements

- Add low pass filter in driving circuit to get a fixed output for H-bridge

Because we use PWM outputs which are a bunch of pulse waves with different duty-cycle for H-bridge control. If we add low pass filter at the output pins from Arduino, we can get a fixed output for Arduino which can smooth the heating or cooling process for TEC. The simulated H-bridge with 2nd order low pass filter is as figure 5-1.



```
.model BC548B
+         Ikf=8
+         Rc=.
+         Mje=
*
*
```

```
**********************
*    Model Genera
*Copyright(c) Symm
*       All Rights R
*   UNPUBLISHED
*   Contains Propr
*     Which is The
*    SYMMETRY O
*    Modeling serv
* Interface Techno
**********************
.MODEL Qtip41c
+IS=7.55826e-11
+IKF=0.526814 IS
+NR=1.5 VAR=100
+NC=1.63849 RB=
```

figure 5-1 H bridge with 2nd order low pass filter

- Make all transistors in H-bridge to work in saturation region

The transistors in H-bridge can easily get very hot even we add heat sinks. We can mitigate this situation by making all the transistors to work in the saturation region. Moreover, transistors power consumption will be lower in this workspace. However, the drawback of this H-bridge design is that we can't get different size of current for TEC. By changing resistance and power supply voltage we get circuit diagram as figure 5-2.

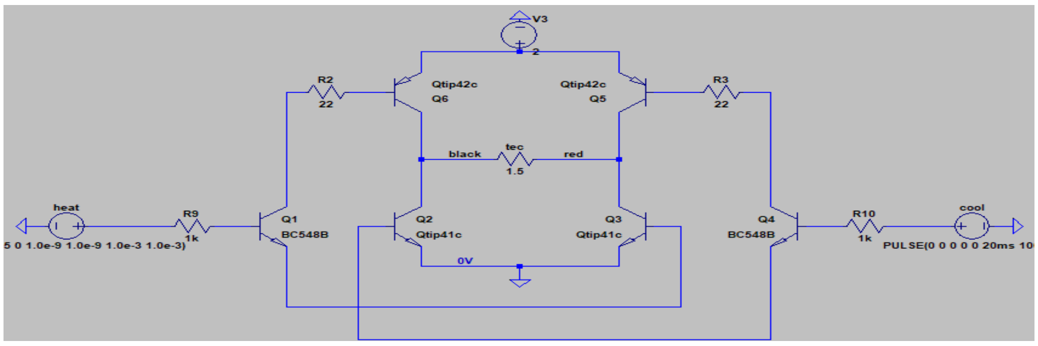


figure 5-2 Improved H bridge

- Use PID controller for signal processing

When the set temperature and room temperature difference is large, we need to provide high current to maintain the temperature of TEC, but also it is likely to cause greater error. The schematic diagram for PID control is as figure 5-3.

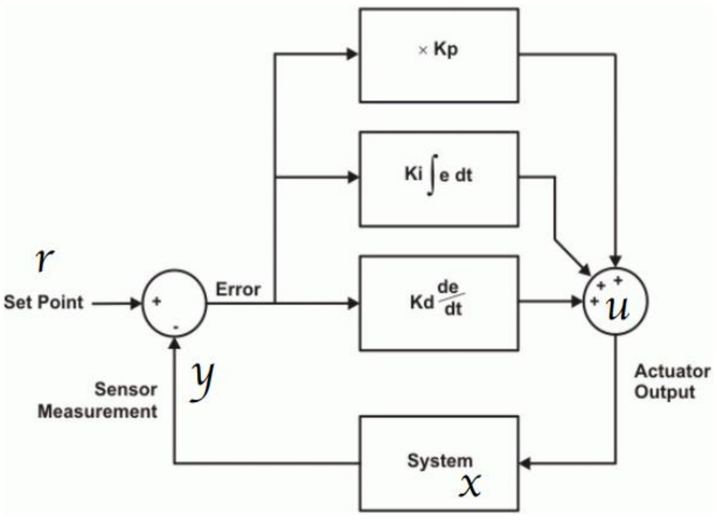


figure 5-3 PID controller

Appendices

Program code

```
#include <LiquidCrystal.h> //include LCD1602 library
#include<math.h>//include math library
double sensorValue; //analogRead value of sensor circuit pin
double CT; //current temperature
double DT; //desired temperature
int A; //temperature difference between CT and DT
int B;// temperature difference between DT and CT
int C;// temperature difference between DT and 25° C
int D; // temperature difference between DT and 25° C
int E; //pwm output for h-bridge due to (DT-25)
int F; //pwm output for h-bridge due to (DT-25)
int lastkey = 0;    //which key was pressed last cycle
unsigned long t0; //how long has key been held for
#define KEYPIN A0//pin for buttons
//button constants
#define btnRIGHT  6
#define btnUP     5
#define btnDOWN   4
#define btnLEFT   3
#define btnSELECT 2
#define btnNONE   (-1)
LiquidCrystal lcd1602(8, 9, 4, 5, 6, 7);//initialize LCD pins
void setup() {
    Serial.begin(9600);
    pinMode(3, OUTPUT);//cool pin D3
    pinMode(11, OUTPUT);//heat pin D11
    lcd1602.begin(16, 2);//initialize LCD screen
}

//function for read buttons analog values
int read_LCD_buttons() {
    int adc_key_in = 0;
    adc_key_in = analogRead(KEYPIN);      // read the value from the sensor
    delay(5); //switch debounce delay. Increase this delay if incorrect switch
selections are returned.
    int k = (analogRead(KEYPIN) - adc_key_in); //gives the button a slight range to
allow for a little contact resistance noise
    if (5 < abs(k)) return btnNONE;  // double checks the keypress. If the two readings
are not equal +/-k value after debounce delay, it tries again.
// my buttons when read are centered at these valies: 0, 144, 329, 504, 741
// we add approx 50 to those values and check to see if we are close
```

```
    if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed
reasons since it will be the most likely result
    if (adc_key_in <50)  return btnRIGHT;
    if (adc_key_in <195)  return btnUP;
    if (adc_key_in <380)  return btnDOWN;
    if (adc_key_in <555)  return btnLEFT;
    if (adc_key_in <790)  return btnSELECT;
    return btnRIGHT;

}
//function for buttons to perform different actions
void dobuttons() {    //updates variables. debounces by only sampling at intervals
    int key;
    key = read_LCD_buttons();
    switch (key) {
    case btnLEFT:
        DT = DT - 10; //press left to -10
        delay(500);
        break;
    case btnRIGHT:
        DT = DT + 10; //press right to +10
        delay(500);
        break;
    case btnUP:
        DT = DT + 0.5; //press up to +0.5
        delay(500);
        break;
    case btnDOWN:
        DT = DT - 0.5; //press down to -0.5
        delay(500);
        break;
    case btnSELECT:
        DT = DT + 10;
        delay(500);
        break;
    case btnNONE:
        DT = DT + 10;
        delay(500);
        break;

    }
}
void loop() {
    //1.buttons to input DT
    int key = read_LCD_buttons();//read keys
    lastkey = key;            //save for next
    unsigned int t;
    if ((key>0) && (lastkey == 0)) {   //if key pressed
        t0 = millis();              //record when button pressed
        dobuttons();
        //act once straight away

    }
    if ((key>0) && (lastkey == key)) {  //if button is still held pressed
        if (millis() - t0>400) {
            t0 = t0 + 200;
            dobuttons();
        }

    };//repeat after delay if held down

      //2.LCD to display DT
    lcd1602.clear();
    lcd1602.setCursor(0, 0);
    lcd1602.print("DT:");//content for first line
    lcd1602.print(DT);//print out DT in first line
      //3.read temperature
    sensorValue = analogRead(A1); //get temperature by capture voltage value from A1
pin
    CT = (4.096E-08)*pow(sensorValue, 3) - (6.6536E-05)*pow(sensorValue, 2) +
0.077292*sensorValue - 6.4016;//relationship between sensorvalueA1 and current
temperature
    /*Serial.print("CT = ");
    Serial.println(CT, 4);//print CT on serial monitor

    //4.LCD display CT
    lcd1602.setCursor(0, 1);//put cursor on second line
    lcd1602.print("CT:");//display CT on second line
    lcd1602.print(CT, 2);



    //5.H-Bridge control
    if ((-0.5)<(CT - DT) && (CT - DT)<0.5) {


        digitalWrite(3, LOW);


        digitalWrite(11, LOW);
        Serial.print("don't need to cool or heat");//h-bridge off
    }
    if (CT>0.5 + DT)//need cooling
    {
        A = (int)(CT - DT + 0.5); //rounded up temperature between DT and CT
        C = (int)(fabs(DT - 25));//compute absolute value between DT and 25
        D = (int)((-0.244)*pow(C, 2) + 10.311*C + 11.261);//relationship between pwm
output and(DT- 25)
Serial.print("cooling");
    }
    switch (A) {//case for different DT-CT
    case 1:
        Serial.print("1c");
        analogWrite(3, D); //pwm output when DT-CT is small
        delay(500);
        break;
    case 2:
    case 3:
    case 4:
    case 5:
        Serial.print("2-5c");
        analogWrite(3, D + 20); //pwm output when DT-CT 2-5
        delay(500);
        break;

    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
        Serial.print("6-10c");
        analogWrite(3, D + 60);
        delay(500);
        break;

    case 11:
    case 12:
    case 13:
    case 14:
    case 15:
        Serial.print("11-15c");
        analogWrite(3, D + 80);
        delay(500);
        break;
```

```
case 16:
case 17:
case 18:
case 19:
case 20:
    Serial.print("16-20c");
    analogWrite(3, D + 100);
    delay(500);
    break;


case 21:
case 22:
case 23:
case 24:
case 25:
    analogWrite(3, D + 100);
    Serial.print("21-25c");
    delay(500);
    break;

case 26:
case 27:
case 28:
case 29:
case 30:
    analogWrite(3, 220); //when DT-CT large use full speed to cool down quickly
    Serial.print("26-30c");
    delay(500);
    break;

case 31:
case 32:
case 33:
case 34:
case 35:
    analogWrite(3, 220);
    Serial.print("31-35c");
    delay(500);
    break;

case 36:
case 37:
case 38:
    case 39:
    case 40:
        analogWrite(3, 220);
        Serial.print("36-40c");
        delay(500);
        break;
    }
    if (DT>0.5 + CT)//need heating
    {
        B = (int)(DT - CT + 0.5); //rounded up temperature between DT and CT
        E = (int)(fabs(DT - 25)); //compute absolute value between DT and 25
        F = (int)((-0.244)*pow(E, 2) + 10.311*E + 11.261); //relationship between pwm
output and(DT- 25)
        Serial.print("heating");
    }
    switch (B) {//case for different DT-CT
    case 1:
        Serial.print("1h");
        analogWrite(11, F); //pwm output when DT-CT is small
        delay(500);
        break;

    case 2:
    case 3:
    case 4:
    case 5:
        Serial.print("2-5h");
        analogWrite(11, F + 20);
        delay(500);
        break;

    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
        Serial.print("6-10h");
        analogWrite(11, F + 60);
        delay(500);
        break;

    case 11:
    case 12:
    case 13:
```

```
    case 14:
    case 15:
        Serial.print("11-15h");
        analogWrite(11, F + 80);
        delay(500);
        break;

    case 16:
    case 17:
    case 18:
    case 19:
    case 20:
        Serial.print("16-20h");
        analogWrite(11, F + 100);
        delay(500);
        break;



    case 21:
    case 22:
    case 23:
    case 24:
    case 25:
        analogWrite(11, F + 100);
        Serial.print("21-25h");
        delay(500);
        break;

    case 26:
    case 27:
    case 28:
    case 29:
    case 30:
        analogWrite(11, 220); //when DT-CT large use full speed to heat up quickly
        Serial.print("26-30h");
        delay(500);
        break;

    case 31:
    case 32:
    case 33:
    case 34:
    case 35:
        analogWrite(11, 220);
```

```
            Serial.print("31-35h");
            delay(500);
            break;

        case 36:
        case 37:
        case 38:
        case 39:
        case 40:
            analogWrite(11, 220);
            Serial.print("36-40h");
            delay(500);
            break;
        }

}
```

Table of SenseValue and temperature

X（SensorValue）	Y（Temperature）
136	5
152	6
169	7
187	8
205	9
223	10
242	11
261	12
302	14
344	16
389	18
435	20
482	22

531	24
579	26
627	28
674	30
719	32
762	34
804	36
843	38
880	40
915	42
948	44
964	45