

ISTANBUL TECHNICAL UNIVERSITY

CONTROL AND AUTOMATION ENGINEERING DEPARTMENT

KON 309E - Microcontroller Systems

Final Project

Instructor : Doç.Dr. Osman Kaan Erol

Assistans : Res. Assist. Aykut Özdemir, Res. Asst. Ertuğrul Keçeci

Student's;

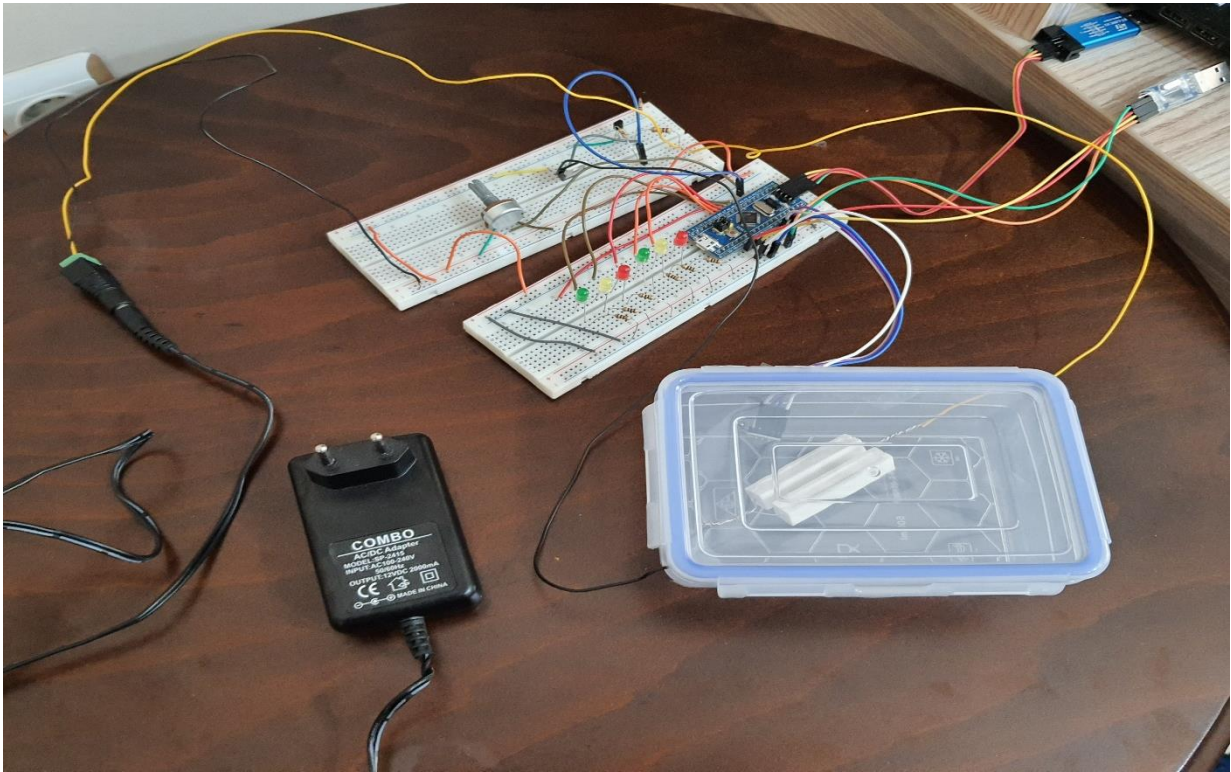
Name Surname :Aydoğan Soylu

Student Number :040170138

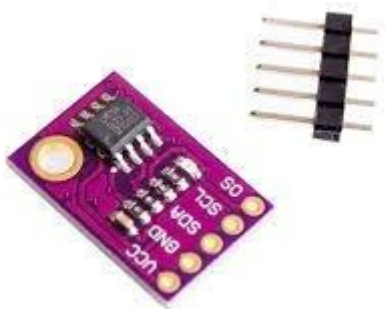


Date:04.02.2021

## My Circuit



In the final project of the microcontroller system, by comparing the data received from the temperature sensor with the reference temperature, the control of the system temperature has been achieved. LM7A is used as a temperature sensor in this project. The stone resistors heated by the pwm sent from the circuit cause the container in which it is located to heat.

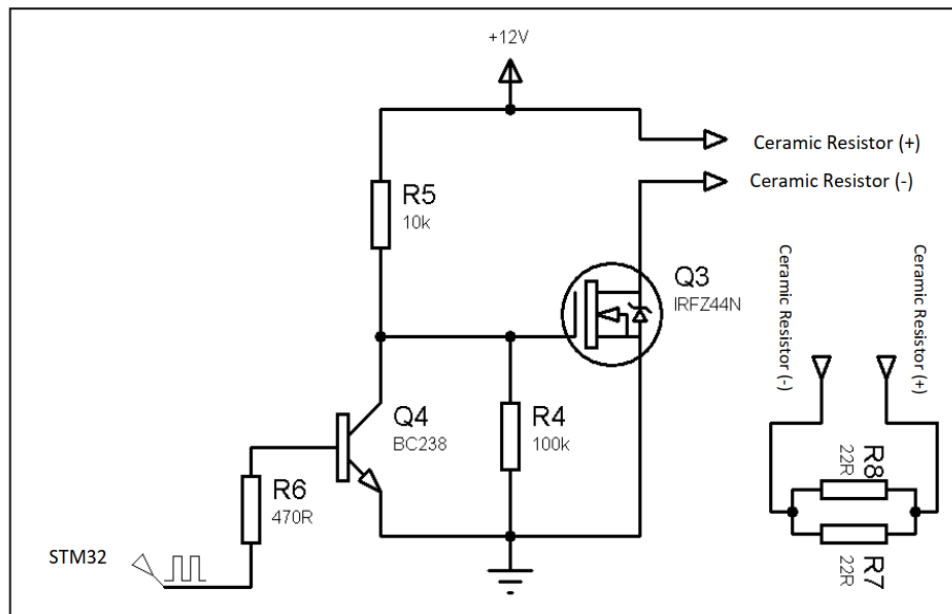


**LM75a**

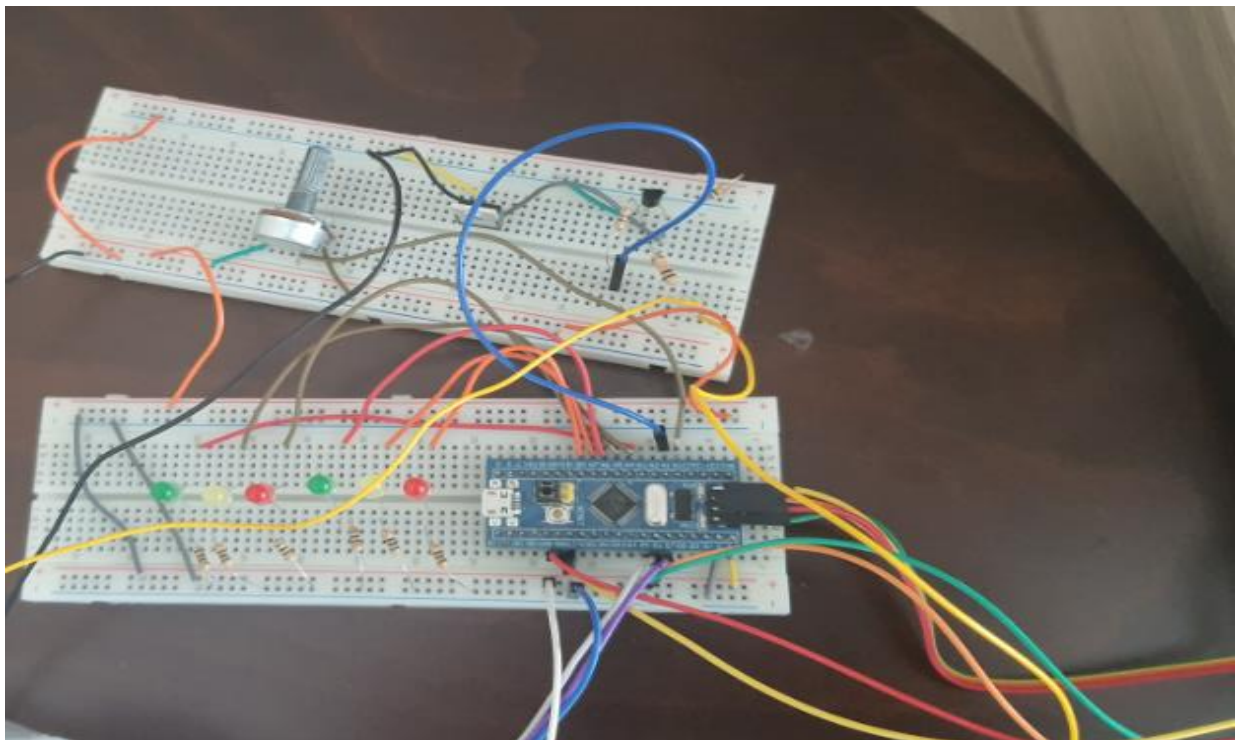


**2 x 22 $\Omega$  Ceramic Resistor (11W)**

## Design of Circuit



**Figure 2:** Circuit diagram of heating mechanism



The circuit was designed by looking at the diagram (Figure 2).

Reference voltage is given to the system by using potentiometer.

Potentiometer (Analog Input)	Reference Degree	Lighting Led
0	0	All reference leds are off
1	28	Reference Green Led is on (Low)
2	33	Reference Yellow Led is on (Medium)
3	37.5	Reference Red Led is on (High)

**Once the steady state is reached, give the system some disturbance (e.g. open the lid) and wait for the system output to reach the steady state before changing the reference**

While the system was running, I opened the lid of the container and let the cold air come out. In this case, the temperature is low. When the lid is closed again, since the temperature is below the reference temperature, the system temperature will increase and reach the reference temperature. This is done by giving the necessary pwm signals with if conditions.

### **The Procedure Of The Experiment And The Results Obtained**

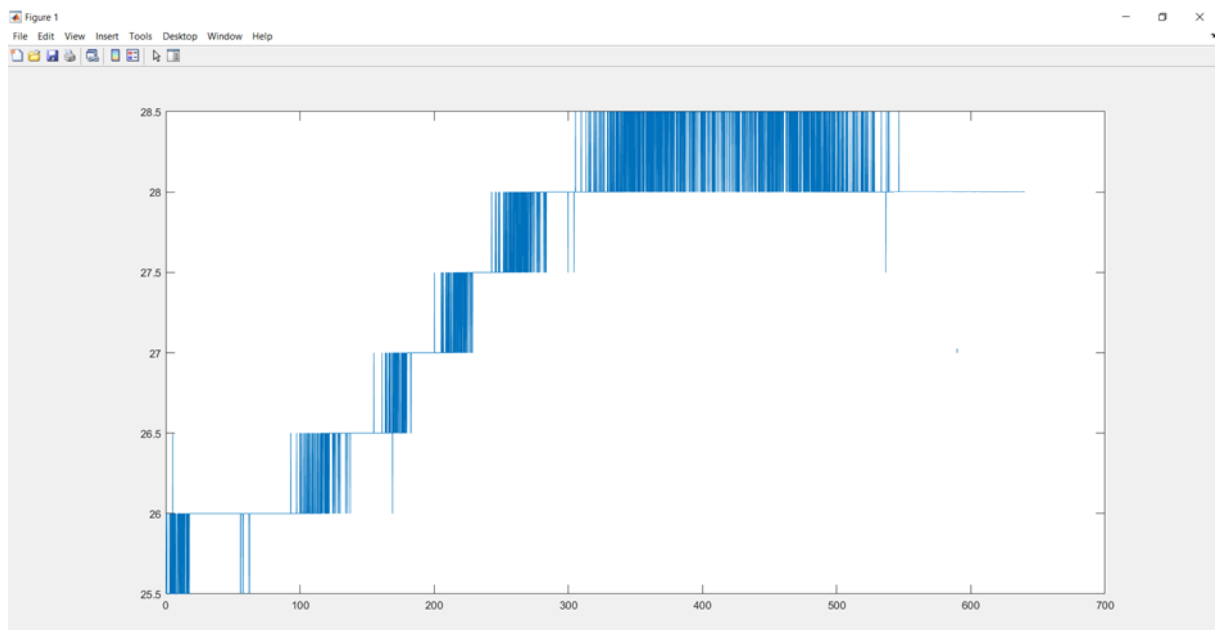
Reference degree values are given to the system according to the analog values entered through the potentiometer. The temperature of the system is set differently according to each reference temperature. Increasing the pwm value was made differently for each reference and the pwm values were determined by testing. (STMStudio). When the temperature of the system caused by stone resistors approaches the reference temperature, the pwm value is increased and the temperature is adjusted so that it does not exceed the reference value too much, and temperature control is facilitated. (Overshoot is small). Also, if the temperature caused by the stone resistors exceeds the reference temperature, the pwm value is maximized and the temperature decreases. Thus, the system temperature is brought closer to the reference temperature without increasing it too much. Therefore, pwm increased or decreased according to the increasing temperature of the system and each reference temperature.

## State Diagram

States	Reference Voltage	Compare Tempature
Analog Value=0	Reference Tempature=0	CompareTempLow()  The temperature of the system is adjusted with pwm according to the reference voltage related to this function.
Analog Value=1	Reference Tempature=28	CompareTempLow()  The temperature of the system is adjusted with pwm according to the reference voltage related to this function
Analog Value=2	Reference Tempature=33	CompareTempMedium()  The temperature of the system is adjusted with pwm according to the reference voltage related to this function
Analog Value=3	Reference Tempature=37.5	CompareTempHigh()  The temperature of the system is adjusted with pwm according to the reference voltage related to this function

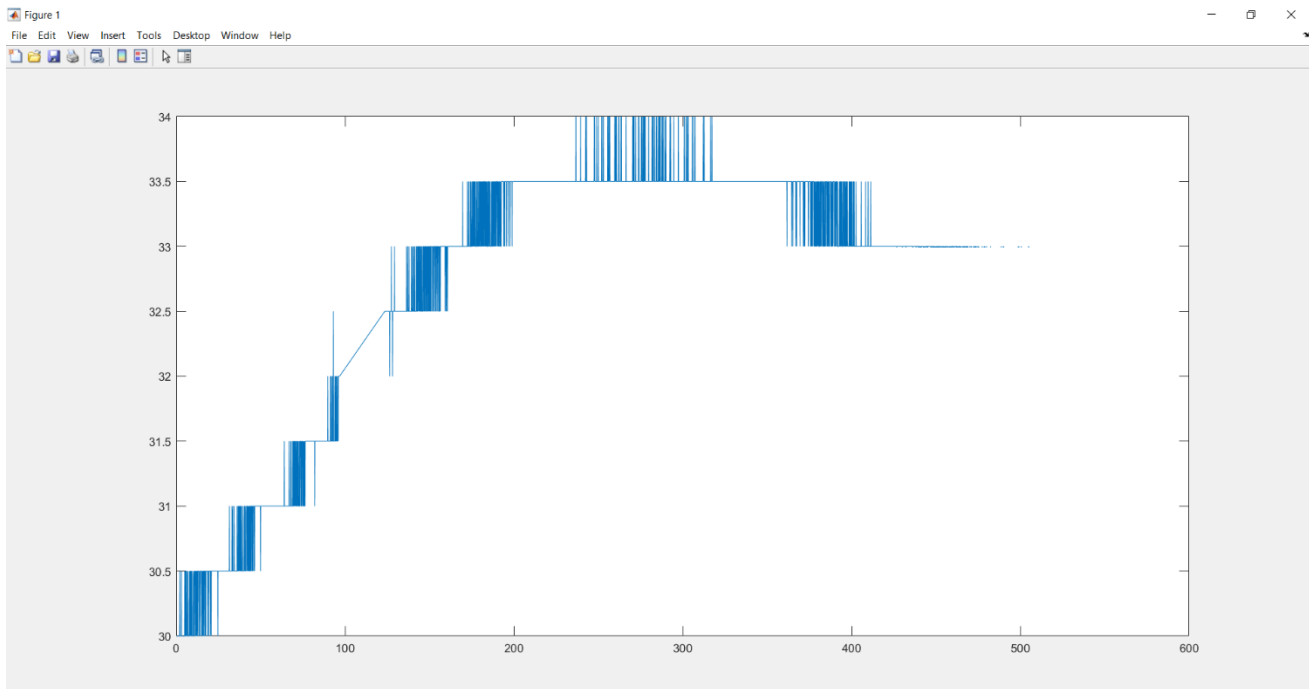
## Matlab Plotting

### Reference Tempature = 28 degree



Overshoot = %1.7857 (Green Led is on because overshoot is smaller than 2 percent.)

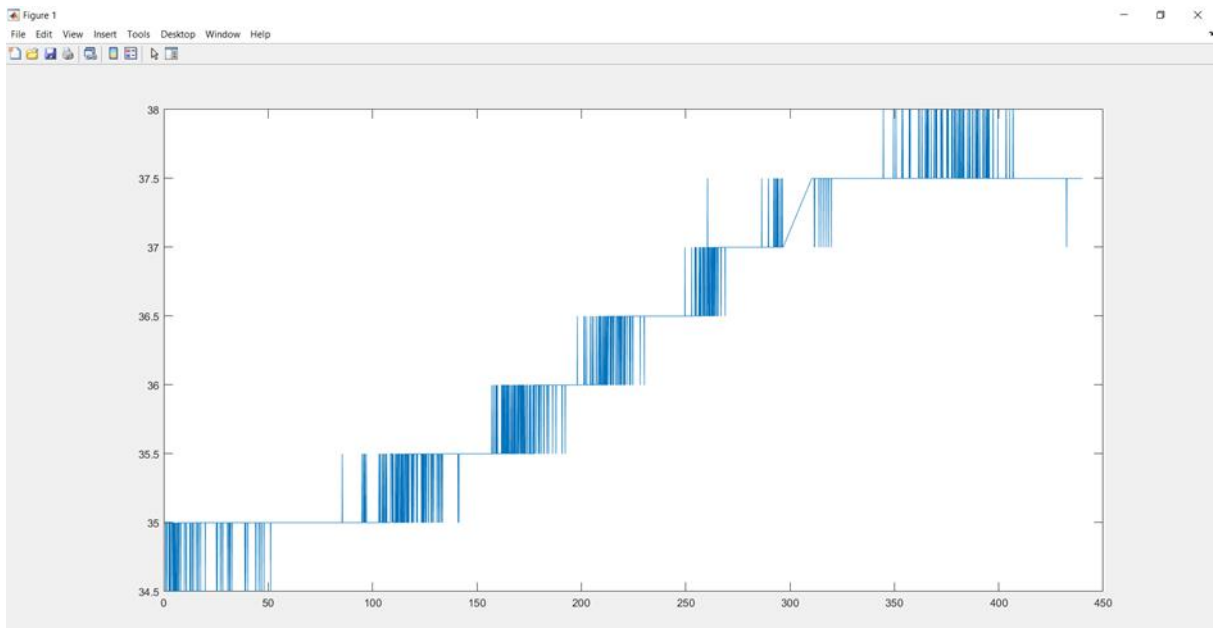
### Reference Temperature = 33 degree



Overshoot = %1.51515 (Green Led is on because overshoot is smaller than 2 percent.)

Overshoot = %3.030303 (Yellow Led is on because overshoot is bigger than 2 percent and smaller than 10 percent.)

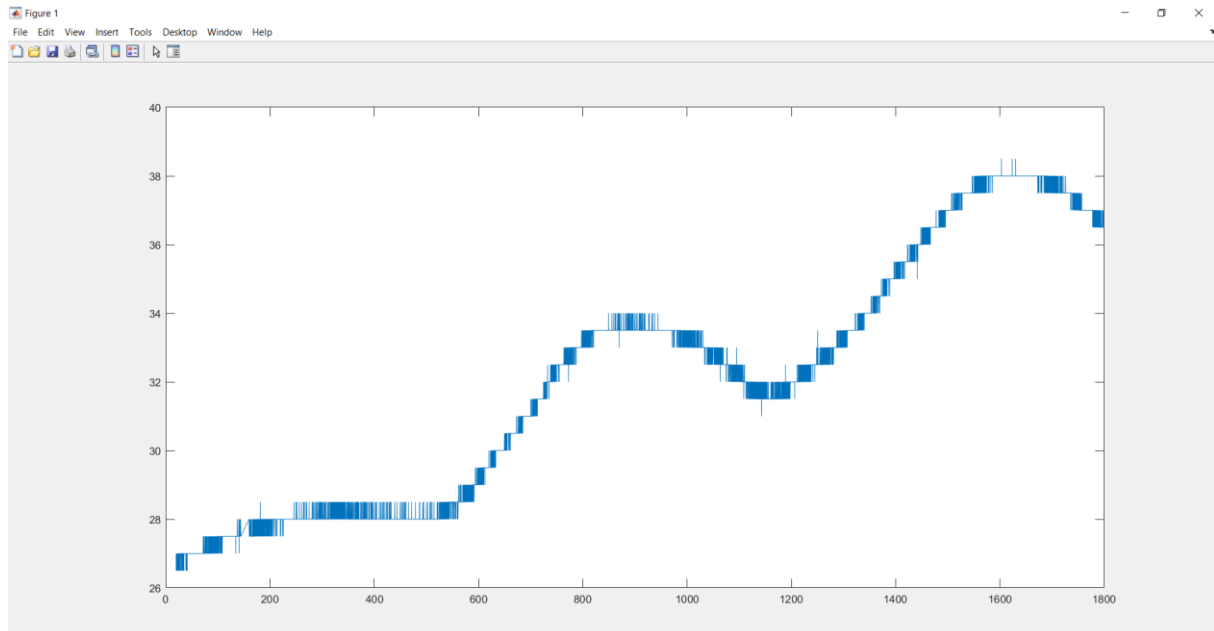
### Reference Temperature = 37.5 degree



Overshoot = %1.3333 (Green Led is on because overshoot is smaller than 2 percent.)



## Plotting The Graph Shown In The Video



The reason for the vibrations in the graph is the temperature value fluctuates between those two temperatures until a temperature dominates in temperature shifts. (For example 29-29.5 etc.)

### Code:

```
1  #include "stm32f10x.h"
2  #include "delay.h"
3  #include <stdio.h>
4
5
6
7  GPIO_InitTypeDef GPIO_InitStructure;
8  I2C_InitTypeDef I2C_InitStructure;
9  USART_InitTypeDef USART_InitStructure;
10 ADC_InitTypeDef ADC_InitStructure;
11 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
12 TIM_OCInitTypeDef TIM_OCInitStructure;
13 NVIC_InitTypeDef NVIC_InitStructure;
14
15
16 uint16_t Pwm_Value = 36000; //Initial condition pwm value
17 double Input_Analog = 0; //Analog value which is read from potentiometer
18 char Buffer_Value[256]; //The variable in which temperature values are kept
19 static float data=0; //Data which is will be transferred.
20 char ADC_Value[20]; //ADC reading
21 int Sent_data=0;
22 float data_Final; //where updated data is kept
23 float Reference_Temp = 0;
24 float OverShoot =0; //Initial condition Overshoot
25
26 //Initial condition analog_Value..This value will be used in switch case condition.
27 int analog_Value=0;
28 int counter=0;
29 int time=1050; //ADC is used to set the value between 0 and 3.
30
31
```

```

31
32 void Gpio_Config() {
33
34     //Setting the pin mode and speed of the reference LEDs
35     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 ;
36     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
37     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
38     GPIO_Init(GPIOA, &GPIO_InitStructure);
39
40     //Setting the pin mode and speed of the reference LEDs
41     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 ;
42     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
43     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
44     GPIO_Init(GPIOA, &GPIO_InitStructure);
45
46     //Setting the pin mode and speed of the reference LED
47     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
48     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
49     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
50     GPIO_Init(GPIOB, &GPIO_InitStructure);
51
52     //Adjusting the potentiometer pin
53     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
54     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
55     GPIO_Init(GPIOA, &GPIO_InitStructure);
56
57     //GPIOA1 pins which connected to TIM2
58     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 ;
59     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
60     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;

```

```

61     GPIO_Init(GPIOA, &GPIO_InitStructure);
62
63     // Configure pins (SDA, SCL)
64     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
65     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
66     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
67     GPIO_Init(GPIOB, &GPIO_InitStructure);
68
69 }
70
71 void Uart_config() {
72     // Configure UART RX - UART module's TX should be connected to this pin
73     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
74     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
75     GPIO_Init(GPIOA, &GPIO_InitStructure);
76     // Configure UART TX - UART module's RX should be connected to this pin
77     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
78     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
79     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
80     GPIO_Init(GPIOA, &GPIO_InitStructure);
81
82     //USART configuration
83     USART_InitStructure.USART_BaudRate = 19200;
84     USART_InitStructure.USART_WordLength = USART_WordLength_8b;
85     USART_InitStructure.USART_StopBits = USART_StopBits_1;
86     USART_InitStructure.USART_Parity = USART_Parity_No;
87     USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
88     USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
89     USART_Init(USART1, &USART_InitStructure);
90

```



```

91 // USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
92 USART_Cmd(USART1, ENABLE);
93
94 }
95
96 //Reference red led is on.
97 void Ref_RedLedOn(){
98     GPIO_SetBits(GPIOA , GPIO_Pin_5);
99 }
100 //Reference yellow led is on.
101 void Ref_YellowLedOn(){
102     GPIO_SetBits(GPIOA , GPIO_Pin_4);
103 }
104 //Reference green led is on.
105 void Ref_GreenLedOn(){
106     GPIO_SetBits(GPIOA , GPIO_Pin_3);
107 }
108 //Overshoot red led is on.
109 void Oslas_RedLedOn(){
110     GPIO_SetBits(GPIOB , GPIO_Pin_0);
111 }
112 //Overshoot yellow led is on.
113 void Oslas_YellowLedOn(){
114     GPIO_SetBits(GPIOA , GPIO_Pin_7);
115 }
116 //Overshoot green led is on.
117 void Oslas_GreenLedOn(){
118     GPIO_SetBits(GPIOA , GPIO_Pin_6);
119 }

```

```

120 //All Reference leds is off.
121 void Ref_AllLedOf(){
122     GPIO_ResetBits(GPIOA , GPIO_Pin_3);
123     GPIO_ResetBits(GPIOA , GPIO_Pin_4);
124     GPIO_ResetBits(GPIOA , GPIO_Pin_5);
125 }
126
127 //All Overshoot leds is off.
128 void Oslas_AllLedOf(){
129     GPIO_ResetBits(GPIOA , GPIO_Pin_6);
130     GPIO_ResetBits(GPIOA , GPIO_Pin_7);
131     GPIO_ResetBits(GPIOB , GPIO_Pin_0);
132 }
133
134 //Data is sent using this function.
135 void UART_Transmit(char *string)
136 {
137     while(*string)
138     {
139         while(!(USART1->SR & 0x00000040));
140         USART_SendData(USART1,*string);
141         *string++;
142     }
143 }
144

```

```

145 //With this function, settings of the I2C sensor are made.
146 void I2C_Config(){
147     // I2C configuration
148     I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
149     I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
150     I2C_InitStructure.I2C_OwnAddress1 = 0x00;
151     I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
152     I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
153     I2C_InitStructure.I2C_ClockSpeed = 100000;
154     I2C_Init(I2C1, &I2C_InitStructure);
155     I2C_Cmd(I2C1, ENABLE);
156 }
157
158 //I2C function to read datas from the sensor
159 void I2C(){
160     // Wait if busy
161     while (I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));
162     // Enable ACK
163     I2C_AcknowledgeConfig(I2C1, ENABLE);
164     // Generate START condition
165     I2C_GenerateSTART(I2C1, ENABLE);
166     while (!I2C_GetFlagStatus(I2C1, I2C_FLAG_SB));
167     // Send device address for read
168     I2C_Send7bitAddress(I2C1, 0x91, I2C_Direction_Receiver); //address of my sensor
169     while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
170     // Read the first data
171     while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
172     Buffer_Value[0] = I2C_ReceiveData(I2C1);
173     // Disable ACK and generate stop condition
174     I2C_AcknowledgeConfig(I2C1, DISABLE);

```

```

175     I2C_GenerateSTOP(I2C1, ENABLE);
176     // Read the second data
177     while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
178     Buffer_Value[1] = I2C_ReceiveData(I2C1);
179 }
180
181
182 void TIM2_Config(){
183     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
184
185     //Configuration TIMER 2
186     TIM_TimeBaseStructure.TIM_Period = 35999;
187     TIM_TimeBaseStructure.TIM_ClockDivision = 0;
188     TIM_TimeBaseStructure.TIM_Prescaler = 19;
189     TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
190     TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
191
192     //Configuration NVIC
193     NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
194     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
195     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
196     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
197     NVIC_Init(&NVIC_InitStructure);
198
199
200     TIM_ITConfig(TIM2, TIM_IT_Update | TIM_IT_CC2, ENABLE);
201     TIM_Cmd(TIM2, ENABLE); //Enable Timer2
202
203     //Timer2 is defined as Clock 2 PWM output.
204     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;

```

```

205 TIM_OCInitStructure.TIM_OCpolarity = TIM_OCpolarity_High;
206 TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
207 TIM_OCInitStructure.TIM_Pulse = 0;
208 TIM_OC2Init(TIM2, &TIM_OCInitStructure);
209
210 }
211
212 //The pwm voltage is used inside the timer 2 interrupt so that the voltage value is given continuously.
213 void TIM2_IRQHandler(void)
214 {
215
216 if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
217 {
218
219 //PWM voltage is sent as time passes.
220 TIM_OCInitStructure.TIM_Pulse = Pwm_Value;
221 TIM_OC2Init(TIM2, &TIM_OCInitStructure);
222 }
223
224 TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
225 if(TIM_GetITStatus(TIM2, TIM_IT_CC2) == SET)
226 {
227 TIM_ClearITPendingBit(TIM2, TIM_IT_CC2);
228 }
229 }
230
231 void ADC_Config(){
232 RCC_ADCCLKConfig(RCC_PCLK2_Div6);
233 //ADC configuration
234 ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;

```

```

235 ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
236 // Enable/disable external conversion trigger (EXTI | TIM | etc.)
237 ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
238 // Configure data alignment (Right | Left)
239 ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
240 // Set the number of channels to be used and initialize ADC
241 ADC_InitStructure.ADC_NbrOfChannel = 1;
242 ADC_Init(ADC1, &ADC_InitStructure);
243 ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1, ADC_SampleTime_7Cycles5);
244 ADC_Cmd(ADC1, ENABLE);
245 ADC_ResetCalibration(ADC1);
246 while(ADC_GetResetCalibrationStatus(ADC1));
247 ADC_StartCalibration(ADC1);
248 while(ADC_GetCalibrationStatus(ADC1));
249 // Start the conversion
250 ADC_SoftwareStartConvCmd(ADC1, ENABLE);
251
252 }
253
254 //With this function,
255 //the temperature measured from the sensor and the reference temperature is compared to ensure that the system temperature is the reference temperature
256 //or is very close to the reference temperature.
257 //Reference temperature 28 degree.
258 void CompareTempLow() {
259
260 if(Reference_Temp == 0){
261 Pwm_Value = 36000;
262 OverShoot = 0;
263 }

```

```

264 if(Reference_Temp > data_Final){
265     if(Reference_Temp>0){
266         Pwm_Value = 0;
267         OverShoot = 0;
268     }
269 }
270 //If the temperature reference temperature difference obtained from the sensor is 2, slow down the heating process.
271 if(Reference_Temp - 2 <= data_Final){
272     if(Reference_Temp>0){
273         Pwm_Value = 17000;
274     }
275 }
276
277 //If the temperature reference temperature difference obtained from the sensor is 1, it slows down the heating process more.
278 if(Reference_Temp - 1 <= data_Final){
279     if(Reference_Temp>0){
280         Pwm_Value = 29000;
281     }
282 }
283
284 //If the temperature read from the sensor exceeds the reference temperature,
285 //the temperature read from the sensor is reduced by not giving any power to the circuit.
286 //Thus, the temperature value read from the sensor is reduced and brought closer to the reference temperature.
287 if(Reference_Temp <= data_Final){
288     if(Reference_Temp>0){
289         Pwm_Value = 36000;
290     }
291 }

292 //Overshoot calculation is made according to the situations of exceeding the reference temperature.
293 if(Reference_Temp <= data_Final){
294     if(Reference_Temp > 0){
295         OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
296         if(OverShoot < 2 ){
297             Oslia_AllLedOf();
298             Oslia_GreenLedOn();
299         }
300         if(OverShoot >= 2 && OverShoot <=10 ){
301             Oslia_AllLedOf();
302             Oslia_YellowLedOn();
303         }
304         if(OverShoot > 10 ){
305             Oslia_AllLedOf();
306             Oslia_RedLedOn();
307         }
308     }
309 }
310 if (OverShoot == 0 ){
311     Oslia_AllLedOf();
312 }
313
314 }
315

```

```

316 //The function it does with CompareTempLow () is the same. By applying different power to the circuit under different conditions,
317 //the temperature obtained from the sensor is brought closer to the reference temperature.
318 ///Reference tempature 33 degree.
319 void CompareTempMedium(){
320
321 if(Reference_Temp == 0){
322     Pwm_Value = 36000;
323     OverShoot = 0;
324 }
325 if(Reference_Temp > data_Final){
326     if(Reference_Temp>0){
327         Pwm_Value = 0;
328         OverShoot = 0;
329     }
330 }
331 if(Reference_Temp - 2 <= data_Final){
332     if(Reference_Temp>0){
333         Pwm_Value = 15000;
334     }
335 }
336
337 if(Reference_Temp - 1 <= data_Final){
338     if(Reference_Temp>0){
339         Pwm_Value = 19000;
340     }
341 }
342 if(Reference_Temp <= data_Final){
343     if(Reference_Temp>0){
344         Pwm_Value = 36000;
345     }
346 }

```

```

347 if(Reference_Temp <= data_Final){
348     if(Reference_Temp > 0){
349         OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
350         if(OverShoot < 2 ){
351             Oslas_AllLedOf();
352             Oslas_GreenLedOn();
353         }
354         if(OverShoot >= 2 && OverShoot <=10 ){
355             Oslas_AllLedOf();
356             Oslas_YellowLedOn();
357         }
358         if(OverShoot > 10 ){
359             Oslas_AllLedOf();
360             Oslas_RedLedOn();
361         }
362     }
363 }
364 if (OverShoot == 0 ){
365     Oslas_AllLedOf();
366 }
367 }
368 }
369
370 //The function it does with CompareTempLow () is the same. By applying different power to the circuit under different conditions,
371 //the temperature obtained from the sensor is brought closer to the reference temperature.
372 ///Reference tempature 37.5 degree.
373 void CompareTempHigh(){
374
375 if(Reference_Temp == 0){
376     Pwm_Value = 36000;
377     OverShoot = 0;
378 }
379 if(Reference_Temp > data_Final){
380     if(Reference_Temp>0){
381         Pwm_Value = 0;
382         OverShoot = 0;
383     }
384 }

```

```

385 if(Reference_Temp - 2 <= data_Final){
386     if(Reference_Temp>0){
387         Pwm_Value = 11000;
388     }
389 }
390
391 if(Reference_Temp - 1 <= data_Final){
392     if(Reference_Temp>0){
393         Pwm_Value = 20000;
394     }
395 }
396 if(Reference_Temp <= data_Final){
397     if(Reference_Temp>0){
398         Pwm_Value = 36000;
399     }
400 }
401 if(Reference_Temp <= data_Final){
402     if(Reference_Temp > 0){
403         OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
404         if(OverShoot < 2 ){
405             Oslas_AllLedOf();
406             Oslas_GreenLedOn();
407         }
408         if(OverShoot >= 2 && OverShoot <=10 ){
409             Oslas_AllLedOf();
410             Oslas_YellowLedOn();
411         }
412         if(OverShoot > 10 ){
413             Oslas_AllLedOf();
414             Oslas_RedLedOn();
415         }
416     }
417 }
418 if (OverShoot == 0 ){
419     Oslas_AllLedOf();
420 }
421
422 }

```

```

423 int main(void)
424 {
425     //Enable clocks
426     RCC_ADCClkConfig(RCC_PCLK2_Div6);
427     RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
428     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO, ENABLE);
429     RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
430     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_ADC1, ENABLE);
431     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
432
433     //Required functions are called.
434     Gpio_Config();
435     Uart_config();
436     I2C_Config();
437     TIM2_Config();
438     ADC_Config();
439
440
441     while(1){
442         //Temperature sensor function is called
443         I2c();
444         counter++;
445         //When the counter is 300, the data obtained from the temperature sensor are sent
446         if(counter==300){
447             //Temperature sensor set to 0.5 resolution
448             Buffer_Value[1] = (Buffer_Value[1] >> 7) & 1;
449
450             //Final version of the data to be sent
451             data_Final = Buffer_Value[0] & (Buffer_Value[1]*0.5);
452             data = data_Final;
453             Sent_data=USART_ReceiveData(USART1);
454             if(data>2000 && Sent_data=='1')
455                 GPIO_SetBits(GPIOA,GPIO_Pin_1);
456             if(data<=2000 && Sent_data=='0')
457                 GPIO_ResetBits(GPIOA,GPIO_Pin_1);
458             sprintf(ADC_Value,"%f\r",data);
459             UART_Transmit(ADC_Value);
460             counter=0;//Update counter

```



```

461 }
462
463 //Analog value read from potentiometer.
464 Input_Analog=(ADC_GetConversionValue(ADC1)/time);
465 //Input_Analog is converted from double to integer.
466 //The input analog is converted integer and used in switch case condition.
467 analog_Value=(int)Input_Analog;
468 switch(analog_Value){
469
470 //When the analog value entered through the potentiometer is zero, all reference LEDs are off.
471 //No heat is applied to the circuit.
472 //By reading the analog data in the state with the conditions, the state can exit and go to other states at any time.
473     case 0:
474         Input_Analog=(ADC_GetConversionValue(ADC1)/time);
475         analog_Value=(int)Input_Analog;
476         if(analog_Value==1){
477             //Go to state 1
478             analog_Value=1;
479         }
480     else if(analog_Value==2){
481         //Go to state 2
482         analog_Value=2;
483     }
484     else if(analog_Value==3){
485         //Go to state 3
486         analog_Value=3;
487     }
488 }
489
490 Ref_AllLedOf();
491 Reference_Temp = 0;
492 CompareTempLow();
493
494 break;
495
496 //When the state is 1, the reference temperature is 28 degrees.
497 //By calling the CompareTempLow() function, a temperature close to the reference is obtained,

```

```

497 //When the state is 1, the reference temperature is 28 degrees.
498 //By calling the CompareTempLow() function, a temperature close to the reference is obtained,
499 //and with this function, care is taken to ensure that the overrun is as little as possible.
500 //In this case only the reference green led will be on.
501     case 1:
502         Input_Analog=(ADC_GetConversionValue(ADC1)/time);
503         analog_Value=(int)Input_Analog;
504
505         if(analog_Value==2){
506             analog_Value=2;
507         }
508     else if(analog_Value==0){
509         analog_Value=0;
510     }
511     else if(analog_Value==3){
512         analog_Value=3;
513     }
514
515     Ref_AllLedOf();
516     Ref_GreenLedOn();
517     Reference_Temp = 28;
518     CompareTempLow();
519     break;
520
521 //When the state is 2, the reference temperature is 33 degrees.
522 //By calling the CompareTempMedium() function, a temperature close to the reference is obtained,
523 //and with this function, care is taken to ensure that the overshoot is as little as possible.
524 //In this case only the reference yellow led will be on.
525     case 2:
526         Input_Analog=(ADC_GetConversionValue(ADC1)/time);
527         analog_Value=(int)Input_Analog;
528
529         if(analog_Value==3){
530             analog_Value=3;
531         }
532     }
533

```

```

534 else if(analog_Value==0){
535     analog_Value=0;
536
537 }
538 else if(analog_Value==2){
539     analog_Value=2;
540
541 }
542 Ref_AllLedOf();
543 Ref_YellowLedOn();
544 Reference_Temp = 33;
545 CompareTempMedium();
546 break;
547 //When the state is 2, the reference temperature is 37.5 degrees.
548 //By calling the CompareTempHigh() function, a temperature close to the reference is obtained,
549 //and with this function, care is taken to ensure that the overshoot is as little as possible.
550 //In this case only the reference red led will be on.
551 case 3:
552     Input_Analog=(ADC_GetConversionValue(ADC1)/time);
553     analog_Value=(int)Input_Analog;
554     if(analog_Value==2){
555         analog_Value=2;
556
557     }
558     else if(analog_Value==3){
559         analog_Value=3;
560
561     }
562     else if(analog_Value==0){
563         analog_Value=0;
564
565     }
566     Ref_AllLedOf();
567     Ref_RedLedOn();
568     Reference_Temp = 37.5;
569     CompareTempHigh();
570     break;
571

```

```

542 Ref_AllLedOf();
543 Ref_YellowLedOn();
544 Reference_Temp = 33;
545 CompareTempMedium();
546 break;
547 //When the state is 2, the reference temperature is 37.5 degrees.
548 //By calling the CompareTempHigh() function, a temperature close to the reference is obtained,
549 //and with this function, care is taken to ensure that the overshoot is as little as possible.
550 //In this case only the reference red led will be on.
551 case 3:
552     Input_Analog=(ADC_GetConversionValue(ADC1)/time);
553     analog_Value=(int)Input_Analog;
554     if(analog_Value==2){
555         analog_Value=2;
556
557     }
558     else if(analog_Value==3){
559         analog_Value=3;
560
561     }
562     else if(analog_Value==0){
563         analog_Value=0;
564
565     }
566     Ref_AllLedOf();
567     Ref_RedLedOn();
568     Reference_Temp = 37.5;
569     CompareTempHigh();
570     break;
571
572 }
573 }
574 }
575 }
576

```

**Video Recording of Circuit Work and Matlab Plot**

<https://youtu.be/dpTpRnVgiN0>