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

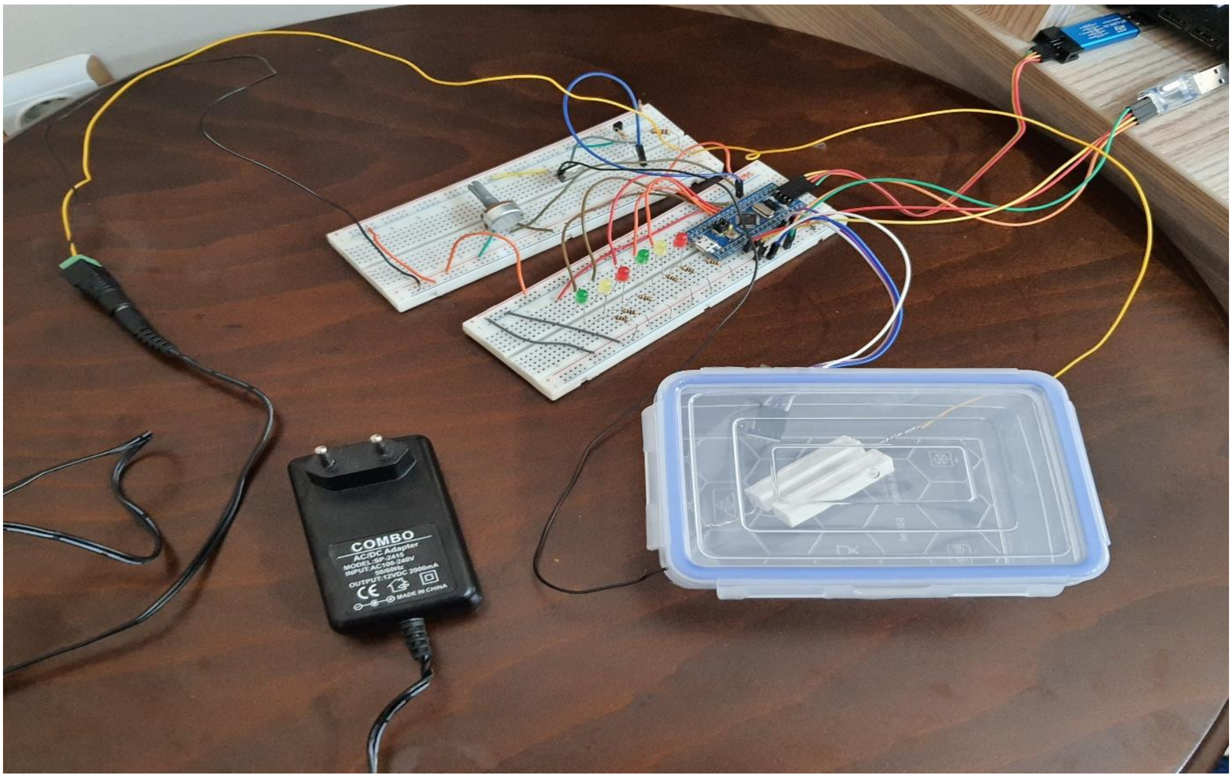
Student's;

Name Surname :Aydoğan Soylu

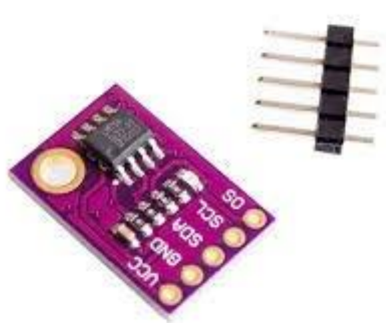
Student Number :040170138

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 22Ω Ceramic Resistor (11W)**

Design of Circuit

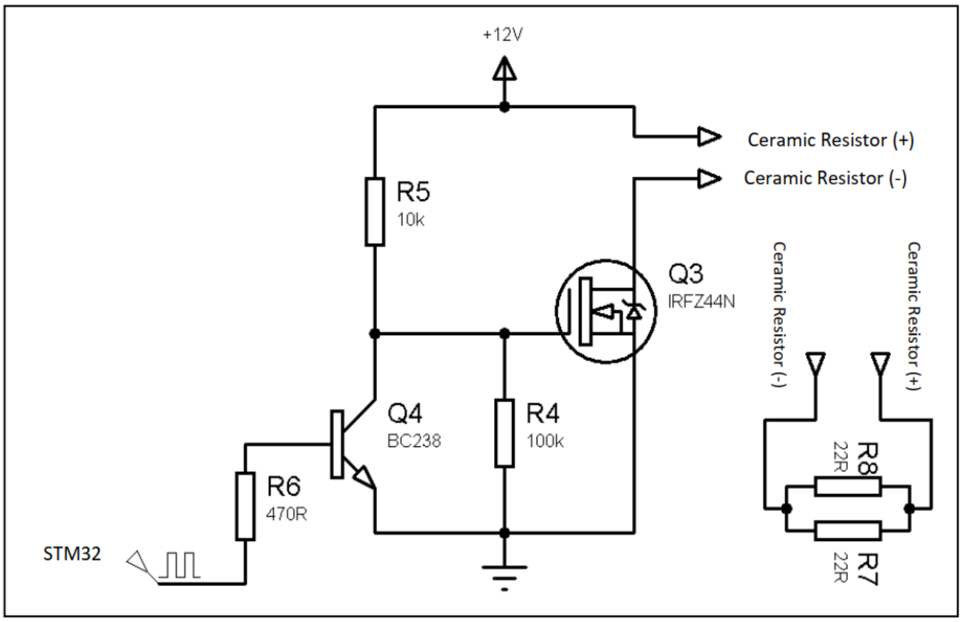
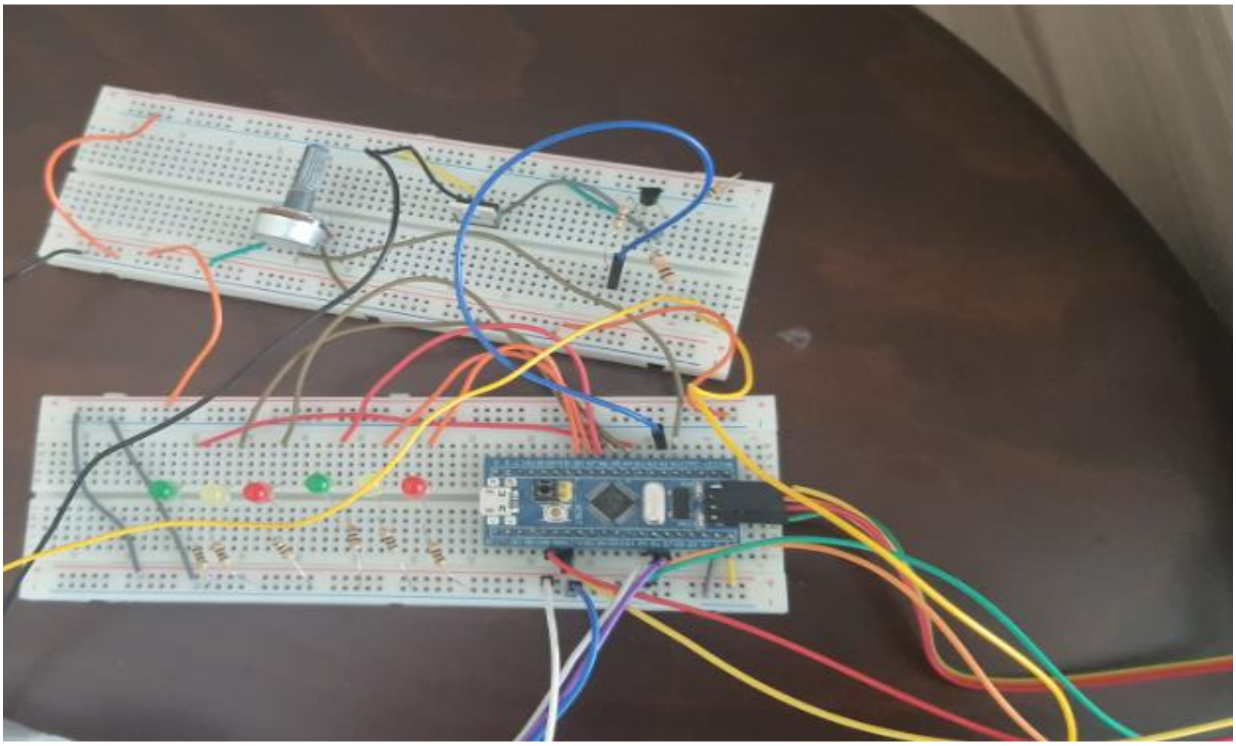


Figure 2: Circuit diagram of heating mechanism



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

Reference voltage is given to the system by using potentiometer.

Potentiometer (Analog İnput)	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 lit 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

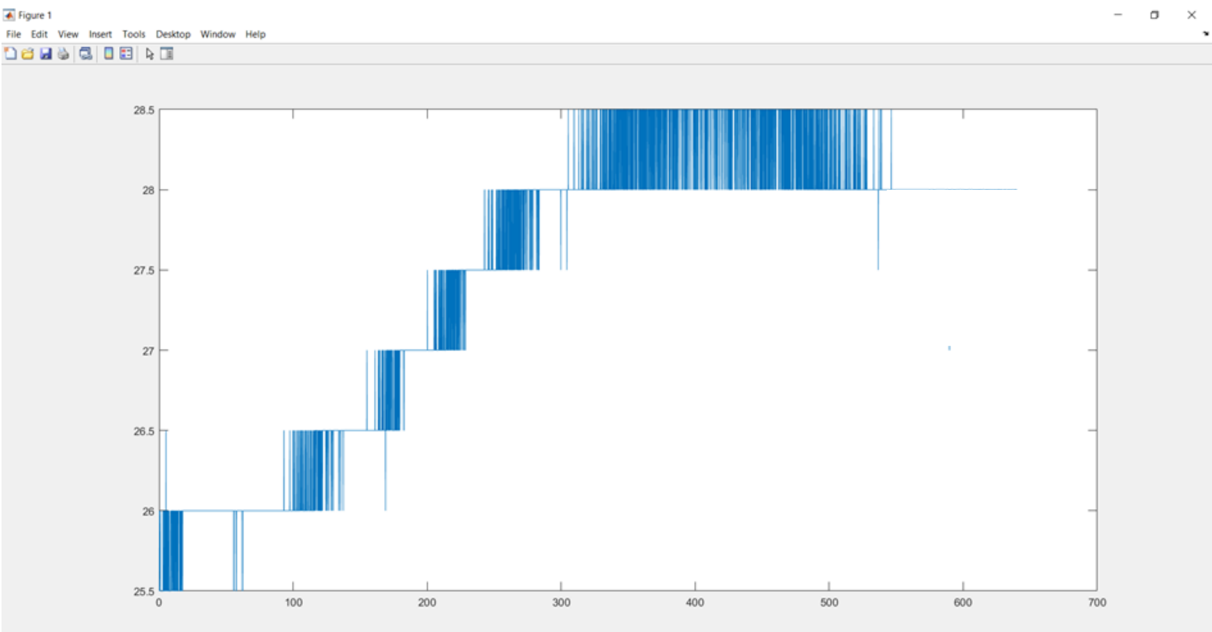
Reference degree values are given to the system according to the analog values entered through the potentiometer. The temperature of the system is set differently according to each reference temperature. Increasing the pwm value was made differently for each reference and the pwm values were determined 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, ıf 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

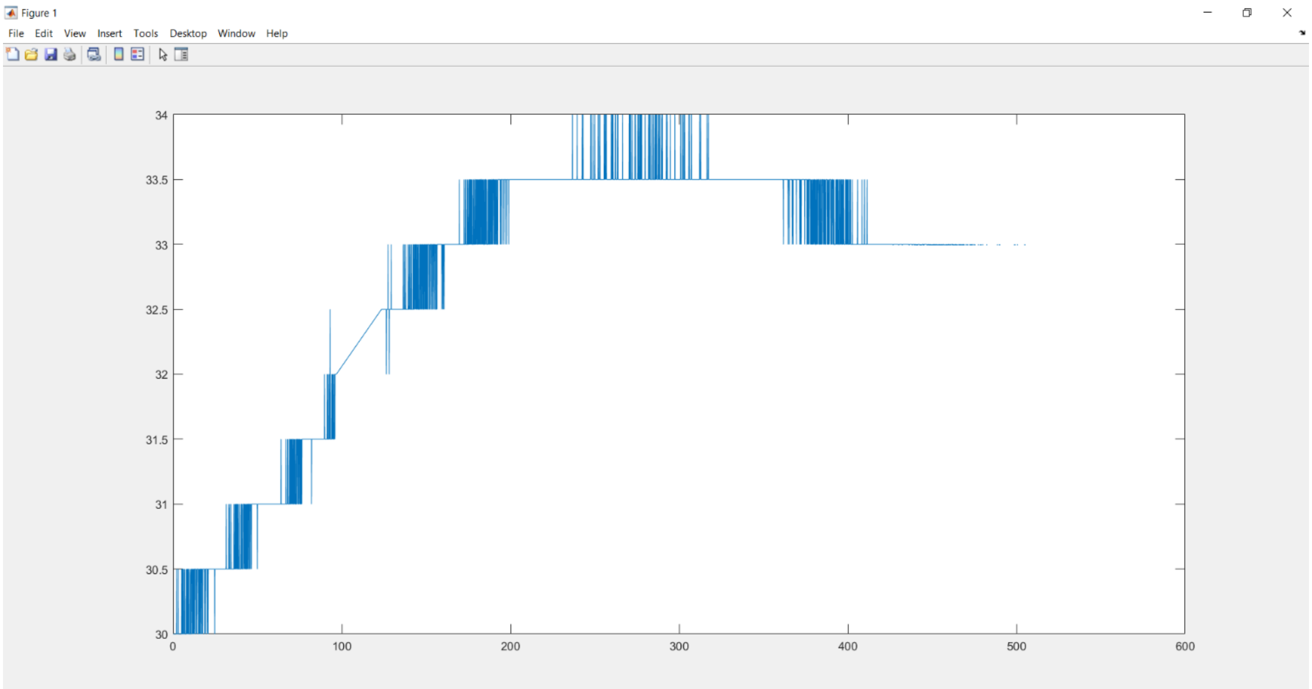
Matlab Plotting

Reference Tempature = 28 degree



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

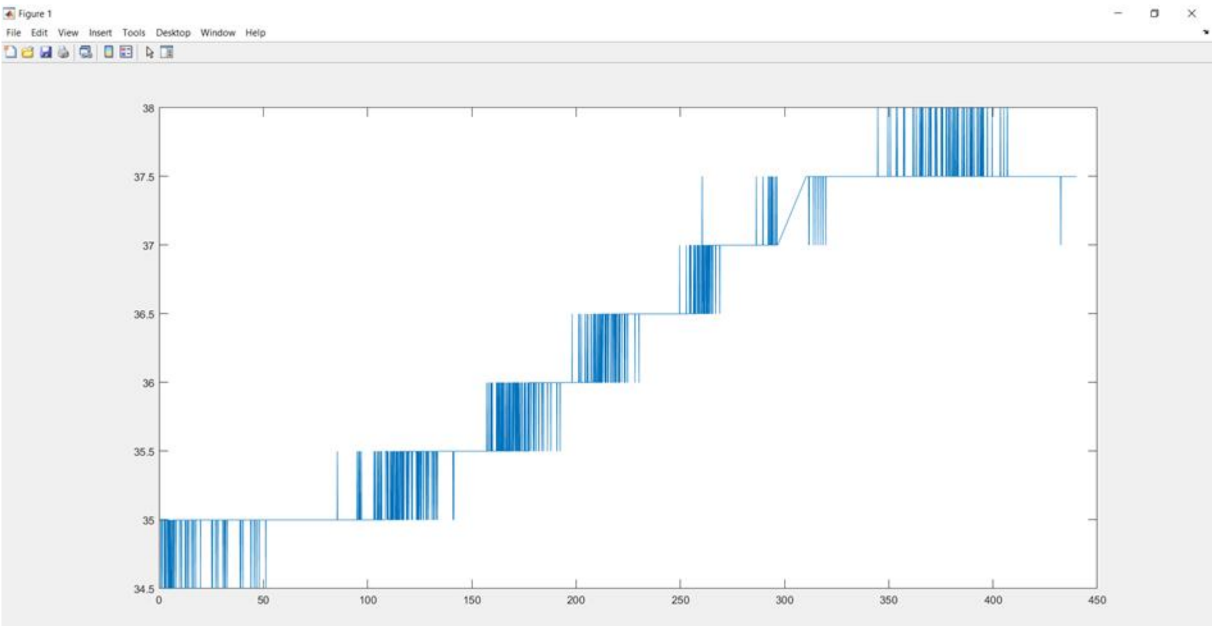
Reference Tempature = 33 degree



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

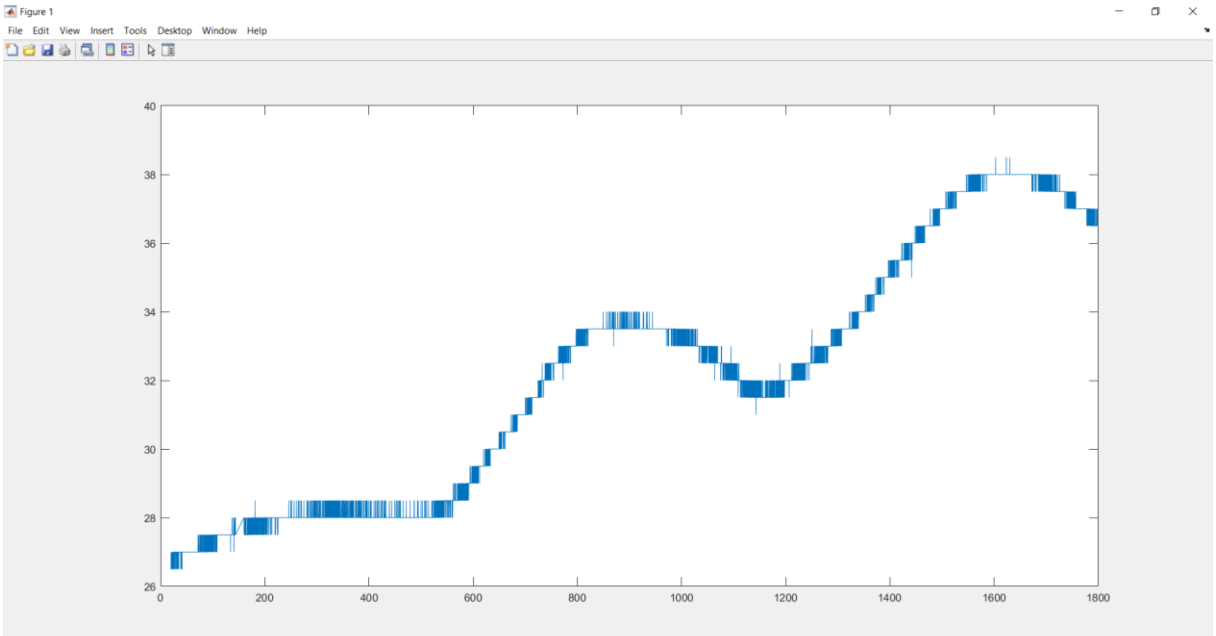
Overshoot = %3.030303   (Yellow Led is on because overshoot is bigger than 2 percent and smaller than 10 percent.)

Reference Tempature = 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:

```
#include "stm32f10x.h"
#include "delay.h"
#include <stdio.h>



GPIO_InitTypeDef GPIO_InitStructure;
I2C_InitTypeDef  I2C_InitStructure;
USART_InitTypeDef USART_InitStructure;
ADC_InitTypeDef ADC_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
NVIC_InitTypeDef NVIC_InitStructure;


uint16_t Pwm_Value = 36000;//Inital condition pwm value
double Input_Analog = 0; //Analog value which is read from potantiometer
char Buffer_Value[256]; //The variable in which temperature values are kept
static float data=0; //Data which is will be transferred.
char ADC_Value[20]; //ADC reading
int Sent_data=0;
float data_Final; //where updated data is kept
float Reference_Temp = 0;
float OverShoot =0; //Inital condition Overshoot

//Inital condition anolog_Value..This value will be used in switch case condition.
int analog_Value=0;
int counter=0;
int time=1050; //ADC is used to set the value between 0 and 3.


```

```c

void Gpio_Config(){

  //Setting the pin mode and speed of the reference LEDs
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3 |  GPIO_Pin_4 |  GPIO_Pin_5 ;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //Setting the pin mode and speed of the reference LEDs
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 |  GPIO_Pin_7 ;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //Setting the pin mode and speed of the reference LED
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  //Adjusting the potentiometer pin
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //GPIOA1 pins which connected to TIM2
  GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_1 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  // Configure pins (SDA, SCL)
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

}

void Uart_config(){
  // Configue UART RX - UART module's TX should be connected to this pin
  GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_10;
  GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Configue UART TX - UART module's RX should be connected to this pin
  GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  //USART configuration
  USART_InitStructure.USART_BaudRate = 19200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
  USART_Init(USART1, &USART_InitStructure);

```

```c
    //  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_Cmd(USART1, ENABLE);

}

//Reference red led is on.
void Ref_RedLedOn(){
    GPIO_SetBits(GPIOA , GPIO_Pin_5);
}
//Reference yellow led is on.
void Ref_YellowLedOn(){
    GPIO_SetBits(GPIOA , GPIO_Pin_4);
}
//Reference green led is on.
void Ref_GreenLedOn(){
    GPIO_SetBits(GPIOA , GPIO_Pin_3);
}
//Overshoot red led is on.
void Osilas_RedLedOn(){
    GPIO_SetBits(GPIOB , GPIO_Pin_0);
}
//Overshoot yellow led is on.
void Osilas_YellowLedOn(){
    GPIO_SetBits(GPIOA , GPIO_Pin_7);
}
//Overshoot green led is on.
void Osilas_GreenLedOn(){
    GPIO_SetBits(GPIOA , GPIO_Pin_6);
}
```

```c
//All Reference leds is off.
void Ref_AllLedOf(){
    GPIO_ResetBits(GPIOA , GPIO_Pin_3);
    GPIO_ResetBits(GPIOA , GPIO_Pin_4);
    GPIO_ResetBits(GPIOA , GPIO_Pin_5);
}

//All Overshoot leds is off.
void Oslias_AllLedOf(){
    GPIO_ResetBits(GPIOA , GPIO_Pin_6);
    GPIO_ResetBits(GPIOA , GPIO_Pin_7);
    GPIO_ResetBits(GPIOB , GPIO_Pin_0);
}

//Data is sent using this function.
void UART_Transmit(char *string)
{
    while(*string)
    {
        while(!(USART1->SR & 0x00000040));
        USART_SendData(USART1,*string);
        *string++;
    }
}
```

```c
//With this function, settings of the I2C sensor are made.
void I2C_Config(){
 // I2C configuration
  I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
  I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
  I2C_InitStructure.I2C_OwnAddress1 = 0x00;
  I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
  I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
  I2C_InitStructure.I2C_ClockSpeed = 100000;
  I2C_Init(I2C1, &I2C_InitStructure);
  I2C_Cmd(I2C1, ENABLE);
}

//I2C function to read datas from the sensor
void I2c(){
  // Wait if busy
  while (I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));
      // Enable ACK
  I2C_AcknowledgeConfig(I2C1, ENABLE);
  // Generate START condition
  I2C_GenerateSTART(I2C1, ENABLE);
  while (!I2C_GetFlagStatus(I2C1, I2C_FLAG_SB));
  // Send device address for read
  I2C_Send7bitAddress(I2C1, 0x91, I2C_Direction_Receiver);//adress of my sensor
  while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
  // Read the first data
  while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
  Buffer_Value[0] = I2C_ReceiveData(I2C1);
  // Disable ACK and generate stop condition
  I2C_AcknowledgeConfig(I2C1, DISABLE);
```

```c
  I2C_GenerateSTOP(I2C1, ENABLE);
  // Read the second data
  while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
  Buffer_Value[1] = I2C_ReceiveData(I2C1);
}


void TIM2_Config(){
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  //Configiration TIMER 2
  TIM_TimeBaseStructure.TIM_Period = 35999;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_Prescaler = 19;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

  //Configiration NVIC
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);


  TIM_ITConfig(TIM2, TIM_IT_Update | TIM_IT_CC2, ENABLE);
  TIM_Cmd(TIM2, ENABLE);//Enable Timer2

  //Timer2 is defined as Clock 2 PWM output.
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
```

```c
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse =0;
    TIM_OC2Init(TIM2, &TIM_OCInitStructure);

  }

//The pwm voltage is used inside the timer 2 interrupt so that the voltage value is given continuously.
void TIM2_IRQHandler(void)
{

if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
{

//PWM voltage is sent as time passes.
TIM_OCInitStructure.TIM_Pulse = Pwm_Value;
TIM_OC2Init(TIM2, &TIM_OCInitStructure);
}

TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  if(TIM_GetITStatus(TIM2, TIM_IT_CC2) == SET)
  {
  TIM_ClearITPendingBit(TIM2, TIM_IT_CC2);
  }
}

void ADC_Config(){
  RCC_ADCCLKConfig(RCC_PCLK2_Div6);
  //ADC configuration
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
```

```c
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
  // Enable/disable external conversion trigger (EXTI | TIM | etc.)
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  // Configure data alignment (Right | Left)
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  // Set the number of channels to be used and initialize ADC
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);
  ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1,ADC_SampleTime_7Cycles5);
  ADC_Cmd(ADC1, ENABLE);
  ADC_ResetCalibration(ADC1);
  while(ADC_GetResetCalibrationStatus(ADC1));
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
  // Start the conversion
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);

}

//With this function,
//the temperature measured from the sensor and the reference temperature is compared to ensure that the system temperature is the reference temperature
//or is very close to the reference temperature.
//Reference tempature 28 degree.
void CompareTempLow(){

if(Reference_Temp == 0){
    Pwm_Value = 36000;
    OverShoot = 0;
  }
```

```c
if(Reference_Temp > data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 0;
    OverShoot = 0;
  }
}
//If the temperature reference temperature difference obtained from the sensor is 2, slow down the heating process.
if(Reference_Temp - 2 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 17000;
  }
  }

//If the temperature reference temperature difference obtained from the sensor is 1, it slows down the heating process more.
if(Reference_Temp - 1 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 29000;
  }
  }

//If the temperature read from the sensor exceeds the reference temperature,
//the temperature read from the sensor is reduced by not giving any power to the circuit.
//Thus, the temperature value read from the sensor is reduced and brought closer to the reference temperature.
if(Reference_Temp  <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 36000;
  }
}
```

```c
//Overshoot calculation is made according to the situations of exceeding the reference temperature.
if(Reference_Temp <= data_Final){
  if(Reference_Temp > 0){
    OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
      if(OverShoot < 2 ){
        Oslias_AllLedOf();
        Osilas_GreenLedOn();
      }
      if(OverShoot >= 2 && OverShoot <=10 ){
        Oslias_AllLedOf();
        Osilas_YellowLedOn();
      }
      if(OverShoot > 10 ){
        Oslias_AllLedOf();
        Osilas_RedLedOn();
      }
    }
  }
  if (OverShoot == 0 ){
      Oslias_AllLedOf();
  }

}
```

```
//The function it does with CompareTempLow () is the same. By applying different power to the circuit under different conditions,
//the temperature obtained from the sensor is brought closer to the reference temperature.
////Reference tempature 33 degree.
void CompareTempMedium(){

if(Reference_Temp == 0){
    Pwm_Value = 36000;
    OverShoot = 0;
  }
if(Reference_Temp > data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 0;
    OverShoot = 0;
  }
}
if(Reference_Temp - 2 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 15000;
  }
  }

if(Reference_Temp - 1 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 19000;
  }
  }
if(Reference_Temp  <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 36000;
  }
}
```

```
if(Reference_Temp <= data_Final){
  if(Reference_Temp > 0){
    OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
      if(OverShoot < 2 ){
        Oslias_AllLedOf();
        Osilas_GreenLedOn();
      }
      if(OverShoot >= 2 && OverShoot <=10 ){
        Oslias_AllLedOf();
        Oslias_YellowLedOn();
      }
      if(OverShoot > 10 ){
        Oslias_AllLedOf();
        Osilas_RedLedOn();
      }
    }
  }
  if (OverShoot == 0 ){
      Oslias_AllLedOf();
  }

}

//The function it does with CompareTempLow () is the same. By applying different power to the circuit under different conditions,
//the temperature obtained from the sensor is brought closer to the reference temperature.
////Reference tempature 37.5 degree.
void CompareTempHigh(){

if(Reference_Temp == 0){
    Pwm_Value = 36000;
    OverShoot = 0;
  }
if(Reference_Temp > data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 0;
    OverShoot = 0;
  }
}
```

```c
if(Reference_Temp - 2 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 11000;
  }
  }

if(Reference_Temp - 1 <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 20000;
  }
  }
if(Reference_Temp  <= data_Final){
  if(Reference_Temp>0){
    Pwm_Value = 36000;
  }
}
if(Reference_Temp <= data_Final){
  if(Reference_Temp > 0){
    OverShoot = ((data_Final -Reference_Temp)/Reference_Temp)*100;
      if(OverShoot < 2 ){
        Oslias_AllLedOf();
        Osilas_GreenLedOn();
      }
      if(OverShoot >= 2 && OverShoot <=10 ){
        Oslias_AllLedOf();
        Osilas_YellowLedOn();
      }
      if(OverShoot > 10 ){
        Oslias_AllLedOf();
        Osilas_RedLedOn();
      }
    }
  }
  if (OverShoot == 0 ){
      Oslias_AllLedOf();
  }

}
```

```c
int main(void)
{
  //Enable clocks
  RCC_ADCCLKConfig(RCC_PCLK2_Div6);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_ADC1, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  //Required functions are called.
  Gpio_Config();
  Uart_config();
  I2C_Config();
  TIM2_Config();
  ADC_Config();


  while(1){
  //Tempature sensor function is called
  I2c();
  counter++;
  //When the counter is 300, the data obtained from the tempature sensor are sent
  if(counter==300){
    //Tempature sensor set to 0.5 resolution
    Buffer_Value[1] = (Buffer_Value[1] >> 7) & 1;

    //Final version of the data to be sent
    data_Final = Buffer_Value[0] + (Buffer_Value[1]*0.5);
    data = data_Final;
    Sent_data=USART_ReceiveData(USART1);
    if(data>2000 && Sent_data=='1')
        GPIO_SetBits(GPIOA,GPIO_Pin_1);
    if(data<=2000 && Sent_data=='0')
      GPIO_ResetBits(GPIOA,GPIO_Pin_1);
      sprintf(ADC_Value,"%f\r",data);
      UART_Transmit(ADC_Value);
    counter=0;//Update counter
```

```
    }

    //Analog value read from potentiometer.
    Input_Analog=(ADC_GetConversionValue(ADC1)/time);
    //Input_Analog is converted from double to integer.
    //The input analog is converted integer and used in switch case condition.
    analog_Value=(int)Input_Analog;
    switch(analog_Value){

  //When the analog value entered through the potentiometer is zero, all reference LEDs are off.
  //No heat is applied to the circuit.
  //By reading the analog data in the state with the conditions, the state can exit and go to other states at any time.
      case 0:
      Input_Analog=(ADC_GetConversionValue(ADC1)/time);
      analog_Value=(int)Input_Analog;
      if(analog_Value==1){
        //Go to state 1
        analog_Value=1;

      }
    else if(analog_Value==2){
      //Go to state 2
      analog_Value=2;

    }
    else if(analog_Value==3){
      //Go to state 3
      analog_Value=3;

    }

      Ref_AllLedOf();
      Reference_Temp = 0;
      CompareTempLow();

      break;
  //When the state is 1, the reference temperature is 28 degrees.
  //By calling the  CompareTempLow() function, a temperature close to the reference is obtained,
```

```
  //When the state is 1, the reference temperature is 28 degrees.
  //By calling the  CompareTempLow() function, a temperature close to the reference is obtained,
  //and with this function, care is taken to ensure that the overrun is as little as possible.
  //In this case only the reference green led will be on.
      case 1:
      Input_Analog=(ADC_GetConversionValue(ADC1)/time);
      analog_Value=(int)Input_Analog;

      if(analog_Value==2){
        analog_Value=2;

      }
      else if(analog_Value==0){
      analog_Value=0;

    }
    else if(analog_Value==3){
      analog_Value=3;

    }
      Ref_AllLedOf();
      Ref_GreenLedOn();
      Reference_Temp = 28;
      CompareTempLow();
      break;

  //When the state is 2, the reference temperature is 33 degrees.
  //By calling the  CompareTempMedium() function, a temperature close to the reference is obtained,
  //and with this function, care is taken to ensure that the overshoot is as little as possible.
  //In this case only the reference yellow led will be on.
      case 2:
      Input_Analog=(ADC_GetConversionValue(ADC1)/time);
      analog_Value=(int)Input_Analog;
      if(analog_Value==3){
        analog_Value=3;

      }
```

```
    else if(analog_Value==0){
    analog_Value=0;

    }
    else if(analog_Value==2){
    analog_Value=2;

    }
    Ref_AllLedOf();
    Ref_YellowLedOn();
    Reference_Temp = 33;
    CompareTempMedium();
    break;
//When the state is 2, the reference temperature is 37.5 degrees.
//By calling the CompareTempHigh() function, a temperature close to the reference is obtained,
//and with this function, care is taken to ensure that the overshoot is as little as possible.
//In this case only the reference red led will be on.
    case 3:
    Input_Analog=(ADC_GetConversionValue(ADC1)/time);
    analog_Value=(int)Input_Analog;
    if(analog_Value==2){
      analog_Value=2;

    }
    else if(analog_Value==3){
    analog_Value=3;

    }
    else if(analog_Value==0){
    analog_Value=0;

    }
    Ref_AllLedOf();
    Ref_RedLedOn();
    Reference_Temp = 37.5;
    CompareTempHigh();
    break;

```

```
    Ref_AllLedOf();
    Ref_YellowLedOn();
    Reference_Temp = 33;
    CompareTempMedium();
    break;
//When the state is 2, the reference temperature is 37.5 degrees.
//By calling the CompareTempHigh() function, a temperature close to the reference is obtained,
//and with this function, care is taken to ensure that the overshoot is as little as possible.
//In this case only the reference red led will be on.
    case 3:
    Input_Analog=(ADC_GetConversionValue(ADC1)/time);
    analog_Value=(int)Input_Analog;
    if(analog_Value==2){
      analog_Value=2;

    }
    else if(analog_Value==3){
    analog_Value=3;

    }
    else if(analog_Value==0){
    analog_Value=0;

    }
    Ref_AllLedOf();
    Ref_RedLedOn();
    Reference_Temp = 37.5;
    CompareTempHigh();
    break;


  }
}
}

```

Video Recording of Circuit Work and Matlab Plot

https://youtu.be/dpTpRnVgiN0