Control & Automation Engineering Department

KON309E Microcontroller Systems

Experiment-7

(Lab-7)

Name of the project	Sensing the temperature and sending the data to PC via UART.
Prepared by:	Mehmet Ali ARSLAN 040170402 Control and Automation Engineering

- First of all the defined pins as follows:

Potentiometer → A0 pin

Red led → A6 pin

UART TX for stm32 → A9 pin

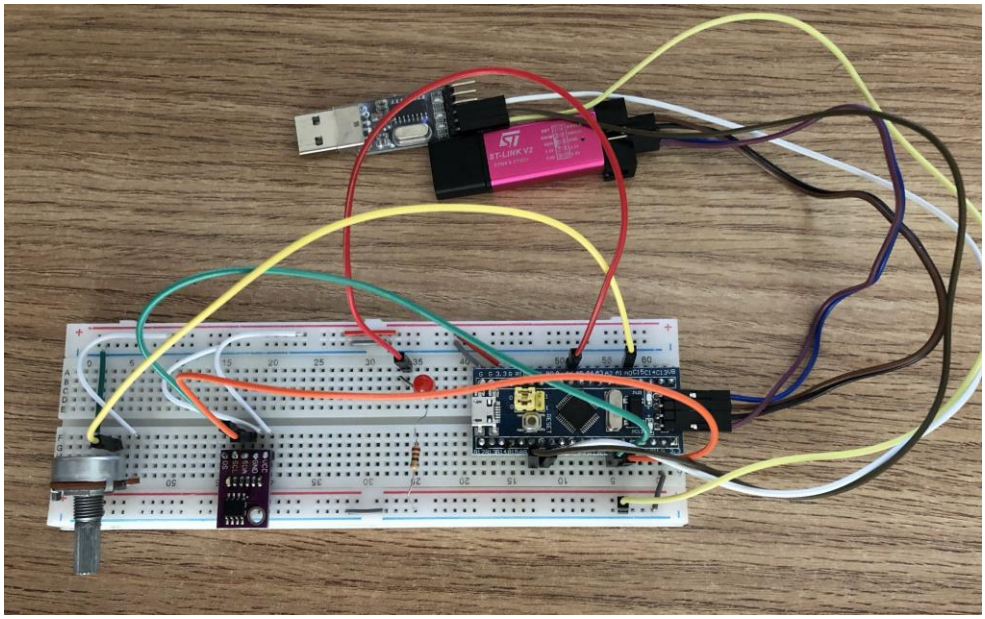
UART RX for stm32 → A10 pin

Temp. Sensor's SCL → B6 pin

Temp. Sensor's SDA → B7 pin

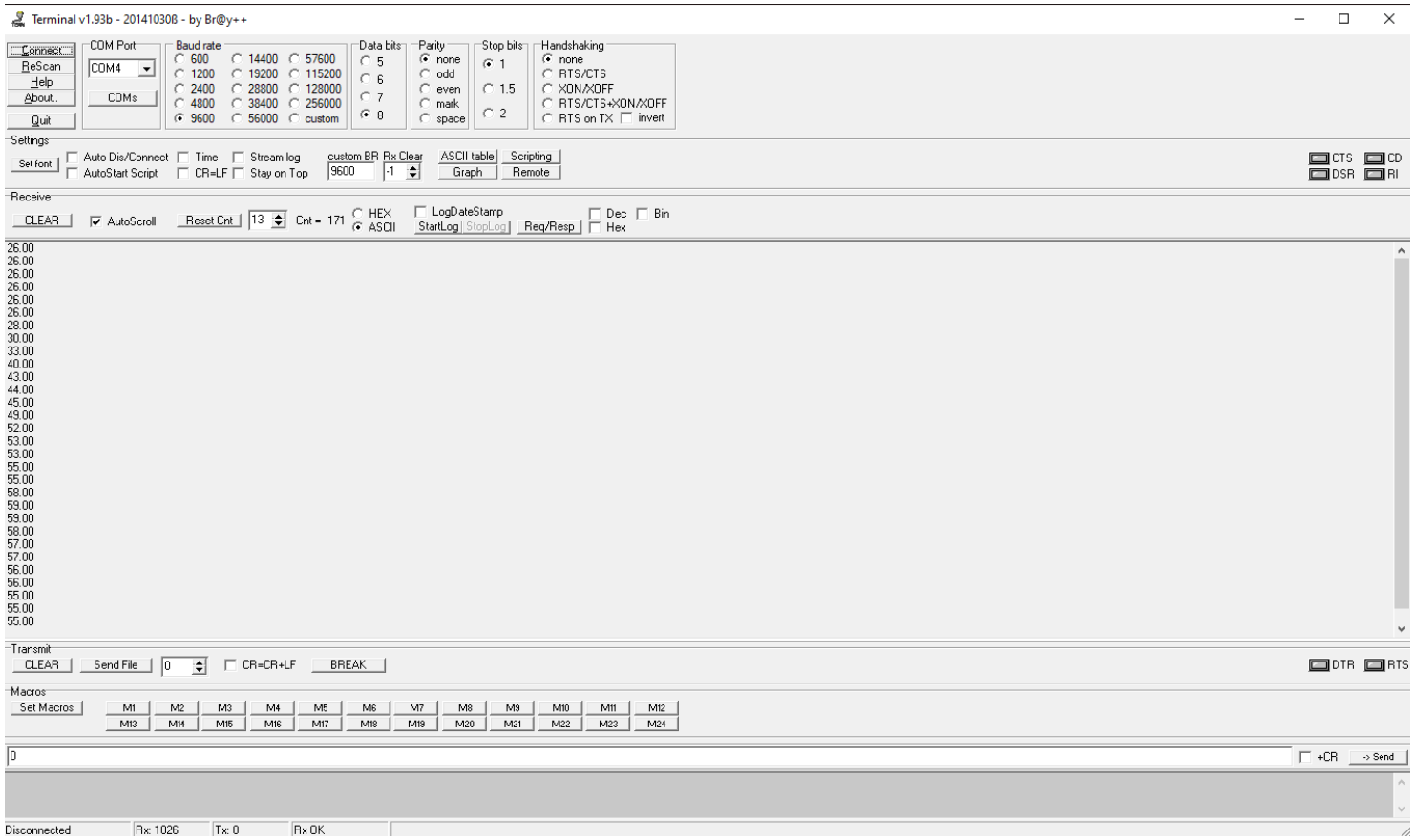
- I used TIM2 for sending temperature data to PC with the interval of 1 seconds. Tim period is 49999 and tim prescaler is 1439. So 72M /1440*50K = 1Hz = 1 second period.
- I have configured the USART and I2C as needed. USART's baud rate is 9600.
- For sending data to pc, in timer interrupt my sendtime value is changing every 1 seconds. In while, if sendtime==true → sprintf(data,"%0.2f\r",Temperature); UART_Transmit(data); sendtime=false; so i can send the temperature data to PC.
- I am recieving data from PC in my while loop.
- I am getting the potValue from potentiometer via ADC and my TempThreshold is potValue/81.7 so when the potentiometer is max, TempThreshold value is 50.
- I have I2C algorithm for reading data from temprature sensor in while loop.
- I have Temp_val for processing the data and i use shifting for assuming the data to Temp_val. Then I divided the Temp_val by 256 to getting the temperature value.
- Finally our algorithm is when the temperature is above the threshold AND the sent data is '1', led is on. When the temperature is below the threshold OR the sent data is '0', led is off. **Other explanations are with the code as a comment.**

PHOTO OF MY CİRCUİT:



LİNK FOR YOUTUBE VİDEO: https://youtu.be/Yn-0y3qI5jk

Terminal screenshot (Not the video captured one):



MAIN CODE :

main.c

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

void UART_Transmit(char *string){  //Our data processing function for sending to PC.
  while(*string)
  {
  while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
  USART_SendData(USART1,*string++);
  }
}

GPIO_InitTypeDef GPIO_InitStructure; // Peripheral libraries
EXTI_InitTypeDef EXTI_InitStructure; //External interrupt library
NVIC_InitTypeDef NVIC_InitStructure; //NVIC Library
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure; //Timer library
TIM_OCInitTypeDef TIM_OCInitStructure; //Oc library
ADC_InitTypeDef ADC_InitStructure; //ADC library
USART_InitTypeDef USART_InitStructure; //USART Library
I2C_InitTypeDef I2C_InitStructure;  //I2C Library

void GPIO_config(void);
void ADC_config(void);
void TIM2_config(void);
void NvicConfig(void);
void USART_config(void);
void I2C_config(void);


uint32_t potValue;  //Our potentiometer value.

static int Sent_data;  //The data sent from PC to stm32.

static float Temperature;   //Temperature value

static int TempThreshold;     //Threshold value of temperature.

bool sendtime = false;    //Sending time value

char data[20];          // The value generated for processing the sending temp. data.

char dataBuffer[20];    //This value is for getting the temperature value from the sensor.
```

```
void TIM2_IRQHandler(void){   //TIMER Function for sending data to pc as period of 1 seconds.

    if((TIM_GetITStatus(TIM2, TIM_IT_Update) == SET) ){  // 1 Hz = 1 seconds of period.

      sendtime=!sendtime; //this variable changes every 1 second.

    TIM_ClearITPendingBit(TIM2, TIM_IT_Update);   //we need to clear line pending bit manually
  }
   }

int main(void) {

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //A port clock enabled
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //B port clock enabled
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,  ENABLE); //AFIO clock enabled
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE); // Timer clock enabled for send data
  RCC_ADCCLKConfig(RCC_PCLK2_Div6); // Setting Adc clock
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);  // ADC clock
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE); // USART CLOCK enabled
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE); //I2C Clock enabled

  delayInit(); //delay initialize

  GPIO_config(); //Init. of configurations.
  ADC_config();
  NvicConfig();
  TIM2_config();
  USART_config();
  I2C_config();


   while(1)
   {
      // Wait if busy
      while (I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));

      // Generate START condition
      I2C_GenerateSTART(I2C1, ENABLE);
      while (!I2C_GetFlagStatus(I2C1, I2C_FLAG_SB));

      // Send device address for read
      I2C_Send7bitAddress(I2C1, 0x90, I2C_Direction_Receiver);
      while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));

      // Read the first data
      while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
      dataBuffer[0] = I2C_ReceiveData(I2C1);

      // Disable ACK and generate stop condition
      I2C_AcknowledgeConfig(I2C1, DISABLE);
      I2C_GenerateSTOP(I2C1, ENABLE);

      // Read the second data
      while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
      dataBuffer[1] = I2C_ReceiveData(I2C1);

      // Disable ACK and generate stop condition
      I2C_AcknowledgeConfig(I2C1, DISABLE);
      I2C_GenerateSTOP(I2C1, ENABLE);


     potValue = ADC_GetConversionValue(ADC1); // getting the value from our potentiometer. Max pot. value is 4085, i checked from value viewer of stmstudio.
     TempThreshold = potValue/81.7;  // Calibrated for when potentiometer is max, its value is 50. I checked from value viewer of stmstudio.


     int16_t Temp_val = (dataBuffer[0] << 8) | dataBuffer[1];   // Calculate temperature value in Celcius
     Temperature = Temp_val / 256;    //Dividing by 256 to get the celcius temperature value

     Sent_data = USART_ReceiveData(USART1); //Data sent from PC terminal
    //If a '1' is sent from the computer to the microcontroller via UART AND the temperature is above the threshold, the LED will be ON.
    if(Temperature > TempThreshold && Sent_data=='1' ){
      GPIO_SetBits(GPIOA,GPIO_Pin_6);
    }
    //If a '0' is sent from the computer to the microcontroller via UART OR the temperature is below the threshold, the LED will be OFF.
    if(Temperature < TempThreshold | Sent_data=='0'){
      GPIO_ResetBits(GPIOA,GPIO_Pin_6);
    }
     //This is for sending data to PC algorithm. sendtime=true for interval of 1 seconds but we need to assume it false at the end of the process.
     if(sendtime==true){
       sprintf(data,"%0.2f\r",Temperature);
       UART_Transmit(data);
       sendtime=false;
     }


  } //Closing while
} //Closing main
void GPIO_config(void)      //GPIO configuration
{
  // Configure analog input
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;  // Configuring pin A0 for analog input (potentiometer).
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  // configure leds' output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;  //Red Led's pin
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; //clock Speed
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // Alternate Function Push-pull mode
  GPIO_Init(GPIOA, &GPIO_InitStructure); //A port

  // 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);
```

```
  // 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);

  // Configure pins (SCL, SDA)
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;  //Our pins for recieve data from temperature sensor.
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;   // AF open drain
  GPIO_Init(GPIOB, &GPIO_InitStructure);

}
void NvicConfig(void)  //NVIC Configuration
{
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn; //Choosing timer2 for NVIC
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
  NVIC_Init(&NVIC_InitStructure);
}

void ADC_config(void)   // ADC configuration
{
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;   //For continious conversation of pot.Value.
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  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);
}

void TIM2_config(void)    // TIMER configuration for TIM2
{
  TIM_TimeBaseStructure.TIM_Period = 49999;
  TIM_TimeBaseStructure.TIM_Prescaler = 1439;            // 72M /1440*50K = 1Hz = 1 second period.
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM2, ENABLE); //Enabling the timer

}
void USART_config(void)   // USART configuration
{
  // USART settings
  USART_InitStructure.USART_BaudRate = 9600;     //Our Baud rate.
  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;   //We use both of them TX and RX
  USART_Init(USART1, &USART_InitStructure);
  USART_Cmd(USART1, ENABLE);
}

void I2C_config(void)   // I2C configuration for temperature sensor
{
  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);
}
```

THE END OF THE REPORT. THANK YOU.