```
/*
 * micro2tempertature.c
 *
 * Author : Guillermo J. Ramires
 */
#define F_CPU 16000000
#define BAUD 9600
#define BAUD_pre ((F_CPU/16/BAUD) - 1)

#include <avr/io.h>
#include <avr/eeprom.h>
#include <stdio.h>

//Custom libraries
#include "TempAndCommsLibraries/ds18b20_2021.h" //temperature sensor
#include "TempAndCommsLibraries/i2c_2021.h" //I2C
#include "TempAndCommsLibraries/serial_2021.h" //UART

#define GPIOA 0x12              //---General Purpose I/O Register A
#define GPIOB 0x13              //---General Purpose I/O Register B

#define BUFFER_SIZE 250
char buffer[BUFFER_SIZE];

int main(void)
{

    printf("Testing");
    // Sets up UART with BAUD 9600
    char buffer[6];
    UBRR0H = (BAUD_pre >> 8);
    UBRR0L = BAUD_pre;

    // Enables serial transmit
    UCSR0B = (1 << TXEN0);
    // Sets 1 stop bit and data size to 8-bits
    UCSR0C = (0 << USBS0) | (0 << UCSZ02) | (1 << UCSZ01) | (1 << UCSZ00);

    // Sets up I2C's GPIO as output
    i2c_init();
    USART_init();
    mcp_write(MCP23017_IODIRA, 0x00);

    // Sets up DS18B20
    DDRB = 0x01;
    PORTB |= (1 << PORTB0);
    uint8_t temperature[2];
    uint8_t temp;
    uint8_t half;
    while (1) {
        // Tells sensor to read temperature
        ds_init();
        ds_writebyte(SKIP_ROM);
        ds_writebyte(CONVERT_T);
        // Waits for conversion to complete

        while(!ds_readbit());

        // Reads temperature from scratchpad
        ds_init();
        ds_writebyte(SKIP_ROM);
        ds_writebyte(READ_SCRATCHPAD);
        temperature[0] = ds_readbyte();
        temperature[1] = ds_readbyte();
        //USART_send(temperature[1] + temperature[0]);
        // Sends reset to stop reading scratchpad
        ds_init(); //resets

        // Saves temperature accordingly and formats into string
        temp = temperature[0] >> 4;
        temp |= (temperature[1] & 0x7) << 4;
        half = temperature[0] & 0xF;
        half *= 625;
        if (half > 99) {
            half /= 10;
        }
        printf(buffer, "%02d.%02d\r", temp, half);
        sprintf(buffer, "%02d.%02d\r", temp, half);
        //USART_send(buffer, "%02d.%02d\r", temp, half);


        // Transmits temperature to I2C

        switch (temp) {
            case 20:
            mcp_write(GPIOA,0b0000000000000);
            break;
            case 21:
            mcp_write(GPIOA,0b0000000000001);
            break;
            case 22:
            mcp_write(GPIOA,0b0000000000011);
            break;
            case 23:
            mcp_write(GPIOA,0b0000000000111);
            break;
            case 24:
            mcp_write(GPIOA,0b0000000001111);
            break;
            case 25:
            mcp_write(GPIOA,0b0000000011111);
            break;
            case 26:
            mcp_write(GPIOA,0b0000000111111);
            break;
            case 27:
            mcp_write(GPIOA,0b0000001111111);
            break;
            case 28:
            mcp_write(GPIOA,0b0000011111111);
            break;
            case 29:
            mcp_write(GPIOA,0b0000111111111);
            break;
            case 30:
            mcp_write(GPIOA,0b0001111111111);
            break;

            default:
            mcp_write(GPIOA,0b0000010101010);
            break;
        }
        _delay_ms(1000); // Waits 10 seconds between readings
    }
}
```