

DA5

Student Name: Meral Abu-Jaser

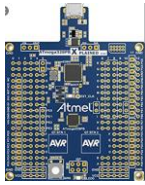
Student #: 5003137888

Student Email: abujaser@unlv.nevada.edu

Primary Github address: https://github.com/MeralAbuJaser/Submission_da.git

Directory: https://github.com/MeralAbuJaser/Submission_da/tree/master/DA5

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS



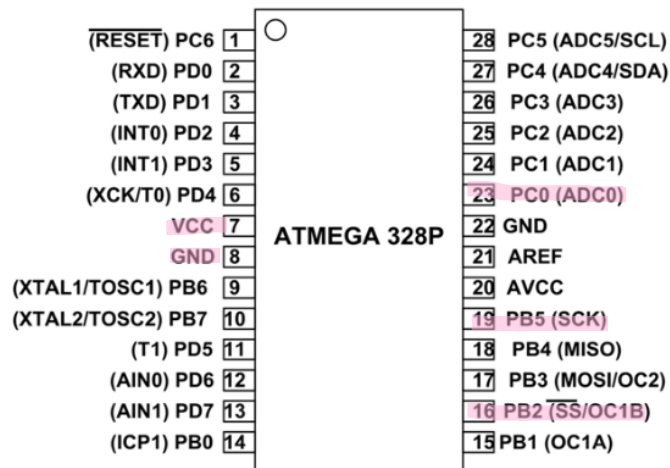
Atmega 328pb

Atmel Studio 7.0

- debugger
- simulator
- assembler
- programmer
- terminal window

Additional components

- LM35



2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A

```
/*
 * DA5_TASK1.c
 *
 * Created: 5/1/2020 11:28:34 PM
 * Author : Meral
 */
#define F_CPU 16000000UL
#define BAUD 9600
#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>
#include <util/delay.h>

#define LATCH 4 /* PD4-RCK */
#define CLOCK 7 /* PD7-SRCK */
#define DATA 0 /* PB0-SER IN */
#define LSBFIRST 0
#define MSBFIRST 1

volatile uint8_t counter = 0; //global variable
volatile int adc_cel; //global variable FOR C

/* Segment byte maps for numbers 0 to 9 */
const uint8_t SEGMENT_MAP[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
                                0x92, 0x82, 0xF8, 0x80, 0x90};

/* Byte maps to select digit 1 to 4 */
const uint8_t SEGMENT_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};
void shift_out_init(void) {
    DDRB |= (1 << DATA);
    DDRD |= (1 << CLOCK) | (1 << LATCH);
}
void uart_init(void) {
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRL_VALUE;

    #if USE_2X
    UCSR0A |= _BV(U2X0);
    #else
    UCSR0A &= ~(_BV(U2X0));
    #endif

    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00); /* 8-bit data */
    UCSR0B = _BV(RXEN0) | _BV(TXEN0); /* Enable RX and TX */
}

int uart_putchar(char c, FILE *stream) {
    if (c == '\n') {
        uart_putchar('\r', stream);
    }
    loop_until_bit_is_set(UCSR0A, UDRE0);
    UDR0 = c;
    return 0;
}

int uart_getchar(FILE *stream) {
    loop_until_bit_is_set(UCSR0A, RXC0);
    return UDR0;
}
```

```

FILE uart_output = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
FILE uart_input = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

void shift_out(uint8_t indata) {
    for (uint8_t i = 0; i < 8; i++) {
        /* Write bit to data port. */
        if (0 == (indata & _BV(7 - i))) {
            // digital_write(SHIFT_OUT_DATA, LOW);
            PORTB &= (0 << DATA);
        } else {
            // digital_write(SHIFT_OUT_DATA, HIGH);
            PORTB |= (1 << DATA);
        }

        /* Pulse clock to write next bit. */
        PORTD |= (1 << CLOCK);
        PORTD &= (0 << CLOCK);
    }
}

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder,
uint8_t val) {
    uint8_t i;

    for (i = 0; i < 8; i++) {
        if (bitOrder == LSBFIRST)
            dataPin |= !(val & (1 << i));
        else
            dataPin |= !(val & (1 << (7 - i)));

        PORTD |= (1 << CLOCK);
        PORTD &= (0 << CLOCK);
    }
}

void shift_out_latch(void) {
    PORTD &= (0 << LATCH);
    PORTD |= (1 << LATCH);
}

void USART_init(void){
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRL_VALUE;
    UCSRC = _BV(UCSZ01) | _BV(UCSZ00); //8-bit data
    UCSRB = _BV(RXEN0) | _BV(TXEN0); //Enable RX and TX
}

//Send data to the serial port
void USART_tx_string( char *data ){
    while ((*data != '\0')){ //while the register is empty enter date
        while (!(UCSR0A & (1 << UDRE0)));
        UDR0 = *data;
        data++; //increment data location forward
    }
}

ISR(TIMER0_OVF_vect){
    counter++; //increment counter
}

```

```

// Initialize ADC
void adc_init(void) {
    /**Setup and enable ADC**/
    ADMUX = (0<<REFS1) | //Reference selection bits
    (1<<REFS0) | //AVcc - external cap at AREF (5)V
    (0<<ADLAR) | //ADC right adjust result
    (1<<MUX2) | //Analog channel selection bits
    (0<<MUX1) | //ADC4 (PC4 PIN27)
    (0<<MUX0);

    ADCSRA = (1<<ADEN) | //ADC enable
    (0<<ADSC) | //ADC start conversion
    (0<<ADATE) | //ADC auto trigger enable
    (0<<ADIF) | //ADC interrupt flag
    (0<<ADIE) | //ADC interrupt enable
    (1<<ADPS2) | //ADC Prescaler select bits
    (1<<ADPS1) | //128 AS PRESCALAR SELECTION BIT
    (1<<ADPS0); //Select channel
}

void read_adc(void){
    unsigned char i = 10;
    adc_cel = 0;
    while(i--){
        ADCSRA |= (1<<ADSC);
        while(ADCSRA & (1<<ADSC));
        adc_cel += ADC;
    }
    adc_cel = (adc_cel/18); //convert fahrenheit to celcius
}

void display(){
    _delay_ms(300);
    read_adc();

    int digit1 = adc_cel/10; //to display the seven segment position 1-4
    int digit2 = adc_cel%10; //to display the number 0-9

    switch(digit1){
        //display 1-4 for 300ms
        case 1:
            //display 1
            PORTD &= (0 << LATCH);
            shift_out(SEGMENT_MAP[1]); //0xF9
            shift_out(SEGMENT_SELECT[0]); //0xF1
            PORTD |= (1 << LATCH);
            break;

        case 2:
            //display 2
            PORTD &= (0 << LATCH);
            shift_out(SEGMENT_MAP[2]); // 0xA4
            shift_out(SEGMENT_SELECT[0]); //0xF1
            PORTD |= (1 << LATCH);
            break;

        case 3:
            //display 3
            PORTD &= (0 << LATCH);
            shift_out(SEGMENT_MAP[3]); //0xB0
            shift_out(SEGMENT_SELECT[0]); //0xF1
            PORTD |= (1 << LATCH);
            break;
    }
}

```

```

case 4:
    //display 4
    PORTD &= (0 << LATCH);
    shift_out(SEGMENT_MAP[4]); //0x99
    shift_out(SEGMENT_SELECT[0]); //0xF1
    PORTD |= (1 << LATCH);
    break;
}
_delay_ms(200);

switch(digit2){
    //display 0-9 for 200 ms
    case 0:
        //display 0
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[0]); //0xc0
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 1:
        //display 1
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[1]); //0xF9
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 2:
        //display 2
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[2]); //0xF9
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 3:
        //display 3
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[3]); //0xB0
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 4:
        //display 4
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[4]); //0x99
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 5:
        //display 5
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[5]); //0x92
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 6:
        //display 6
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[6]); //0x82
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
    case 7:
        //display 7
        PORTD &= (0 << LATCH);
        shift_out(SEGMENT_MAP[7]); //0xF8
        shift_out(SEGMENT_SELECT[1]); //0xF2
        PORTD |= (1 << LATCH);
        break;
}

```

```

        case 8:
            //display 8
            PORTD &= (0 << LATCH);
            shift_out(SEGMENT_MAP[8]);           //0x80
            shift_out(SEGMENT_SELECT[1]);        //0xF2
            PORTD |= (1 << LATCH);
            break;
        case 9:
            //display 9
            PORTD &= (0 << LATCH);
            shift_out(SEGMENT_MAP[9]);           //0x90
            shift_out(SEGMENT_SELECT[1]);        //0xF2
            PORTD |= (1 << LATCH);
            break;
    }
}

int main(){
    USART_init(); //call function to initialize
    TCCR0A = 0x00; //normal mode timer
    TCCR0B = 0x05; //set pre-scaler = 1024
    TCNT0 = 0; //counter = 0
    TIMSK0 = (1<<TOIE0); //enable interrupt
    sei(); //enable global
    adc_init(); //initialize adc
    shift_out_init();
    char celcius[20];

    while (1){
        //call the display function to output on temperture of the senser
        display();

        USART_tx_string("Celcius degree: "); //print string
        USART_tx_string(celcius); //print the temperture
        sprintf(celcius,sizeof(celcius),"%u\r\n",adc_cel); //print formatted output
    }
    return 0;
}

```

Task 2

```
/*
 * DA5_TASK2.c
 *
 * Created: 5/5/2020 9:02:20 PM
 * Author : Meral
 */
#define F_CPU 16000000UL
#define BAUD 9600
#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>
#include <util/delay.h>

/* Segment byte maps for numbers 0 to 9 */
const uint8_t SEGMENT_MAP[] = {0x03, 0x9F, 0x25, 0x0D, 0x99,
                                0x49, 0x41, 0x1F, 0x01, 0x09};

/* Byte maps to select digit 1 to 4 */
const uint8_t SEGMENT_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};

//variables to use for calculations
volatile uint8_t data; //to send data
volatile uint8_t adc_value; //to read the analoge value
volatile uint8_t adc_cel; //to save the temperture value

void USART_init(void){
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRL_VALUE;
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00); //8-bit data
    UCSR0B = _BV(RXEN0) | _BV(TXEN0); //Enable RX and TX
}

//Send data to the serial port
void USART_tx_string(char *data ){
    while ((*data != '\0')){ //while the register is empty enter date
        while (!(UCSR0A & (1 <<UDRE0)));
        UDR0 = *data;
        data++; //increment data location forward
    }
}

void enableADC(){
    ADMUX |= (1<<REFS0)|(1<<MUX2); //AREF ,analog channel bit selection

    ADCSRA |= (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1); //Enable ADC
}

uint8_t Read_ADC() {
    DDRC &= ~(1<<4); //PINC4 set as input
    enableADC(); //enable adc
    ADCSRA |= (1<<ADSC); //adc conversion
    while (!(ADCSRA&(1<<4))); //Wait
    ADCSRA |= (1<<4); //clear ADIF
    return ADC; //returns adc value
}

void SPIinitialize() {
    DDRD |= (1<<PIND4); //sets latch clock pin as output
    DDRB |= (1<<PINB3); //sets MOSI pin as output
    DDRB |= (1<<PINB5);
    PORTD &= ~(1<<PIND4); //sets clock output to zero
    SPCR0 = (1<< MSTR)| 0xFF; //enable SPI mode as master
}
```

```

void SPIwrite(data){
    SPDR0 = data;                // Sends data
    while (!(SPSR0&(1<<SPIF)));  // Waits until data is sent
    SPSR0 |= (1<<SPIF);          //clears flag
}

void CSC(){
    PORTD |= (1<<PIND4); //high
    _delay_ms(20);        //delay for 20ms
    PORTD &= ~(1<<PIND4); //low
}

void SendDigits(uint8_t digits[]) {
    for (uint8_t i = 0; i < 4; i++) {
        SPIwrite(SEGMENT_MAP[digits[i]]); //send seven-segment value
        SPIwrite(1<<(4+i));               //send value to seven-segment
        CSC(); //Clear the clock, 20ms delay, set clock
    }
}

void SetDigits(uint16_t temp, uint8_t array[]) {
    for (uint8_t i = 0; i < 4; i++) {
        array[i] = ((uint8_t)(temp/pow(10, i)))%10;
    }
}

int main(void) {
    uint8_t digits[4]; //array to stores the the 4 digits for display
    SPIinitialize();   //initialize SPI
    USART_init();
    char celcius[15]; //to hole the value of the temperture

    while (1) {
        adc_value = Read_ADC(); //setting the value read by sensor to adc_value
        adc_cel = adc_value-27; //temp in celcius
        SetDigits(adc_cel, digits); //to output the degree on the seven-segment display
        SendDigits(digits);

        USART_tx_string("Celcius degree: "); //print string
        USART_tx_string(celcius); //print the temperture
        snprintf(celcius, sizeof(celcius), "%u\r\n", adc_cel); //print formatted output
    }
}

```


Task 3

```
/*
 * DA5_TASK3.c
 *
 * Created: 5/6/2020 1:08:53 AM
 * Author : Meral
 */
#define F_CPU 16000000UL
#include <stdlib.h>
#include <string.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
#define PRESCALE (((F_CPU / (BAUDRATE * 16UL))) - 1)

void write(uint8_t value){
    //set low and high input/output
    PORTC &= ~(1<<PC1);
    DDRC |= (1<<PC1);

    if(value) //if 1 is passed go high
        DDRC &= ~(1<<PC1);
    else //if not 1 stay low
        DDRC |= (1<<PC1);

    //wait 60uS and release data
    _delay_us(50);
    DDRC &= ~(1<<PC1); //input
}

uint8_t ds18b20_readbit(void){
    uint8_t bit=0;
    //1us delay on low
    PORTC &= ~(1<<PC1); //low
    DDRC |= (1<<PC1); //output
    _delay_us(1);

    //send data and wait for 14us
    DDRC &= ~(1<<PC1); //input
    _delay_us(14);

    //read the value
    if(PINC & (1<<PC1))
        bit=1;

    //wait for 45us and return bit value
    _delay_us(45);
    return bit;
}

void write_byte(uint8_t byte){
    uint8_t i=8;
    while(i--){
        write(byte&1);
        byte >>= 1;
    }
}

uint8_t ds18b20_readbyte(void){
    uint8_t i=8, n=0;
    while(i--){
        n >>= 1;
        n |= (ds18b20_readbit())<<7;
    }
    return n;
}
```

```

void USART_Init(unsigned long BAUDRATE)    {
    UCSRB |= (1 << RXEN0) | (1 << TXEN0);    //enable USART transmitter and receiver
    UCSRC |= (1 << UCSZ00) | (1 << UCSZ01); //write USCR and 1 stop bit
    UBRR0L = PRESCALE;                        //load UBRRL
    UBRR0H = (PRESCALE >> 8);                 //load UBRRH
}

void USART_Tx(char data){
    UDR0 = data;                             //Write data
    while (!(UCSR0A & (1<<UDRE0)));          //data transmit? buffer set to empty
}

void USART_SendString(char *str){
    int i=0;
    while (str[i]!=0){
        USART_Tx(str[i]);
        i++;
    }
}

double Read_temp(){
    PORTC &= ~(1<<PC1); //low
    DDRC |= (1<<PC1); //output
    _delay_us(480); //low for 480us

    //release line and wait for 60uS
    DDRC &= ~(1<<PC1); //input
    _delay_us(500);

    write(0xCC); //skip next
    write(0x44); //convert the temp
    if(!ds18b20_readbit()); //while still converting stay here

    //reset
    PORTC &= ~(1<<PC1);
    DDRC |= (1<<PC1);
    _delay_us(480);

    //input the data
    DDRC &= ~(1<<PC1); //input
    _delay_us(500);

    write_byte(0xCC); //skip next
    write_byte(0xBE); //read

    //read 2 byte
    uint8_t temp1 = ds18b20_readbyte();
    uint8_t temp2 = ds18b20_readbyte();

    //return the converted temp
    return (( temp2 << 8 ) + temp1) * 0.0625;
}

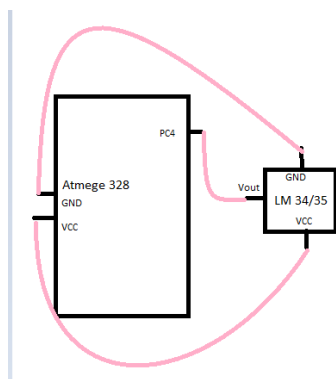
void display(){
    char celcius[50];
    dtostrf(Read_temp(), 3, 0, celcius);
    USART_SendString("Celcius degree: ");
    USART_SendString(celcius);
    USART_SendString("\n\n");
    _delay_ms(500);
}

int main(void) {
    USART_Init(9600);
    while (1){
        display();
    }
    return 0;
}

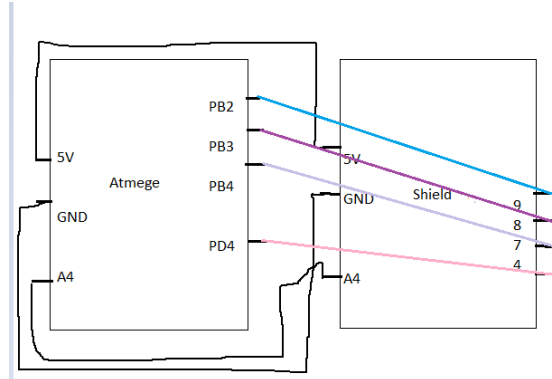
```

3. SCHEMATICS

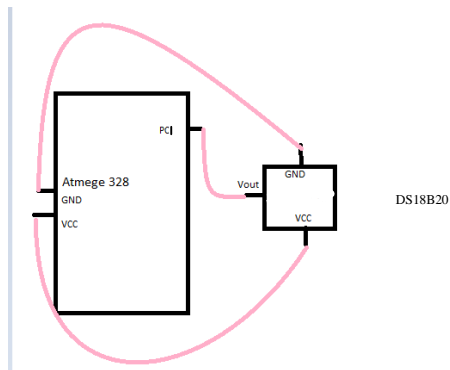
Task 1



task 2

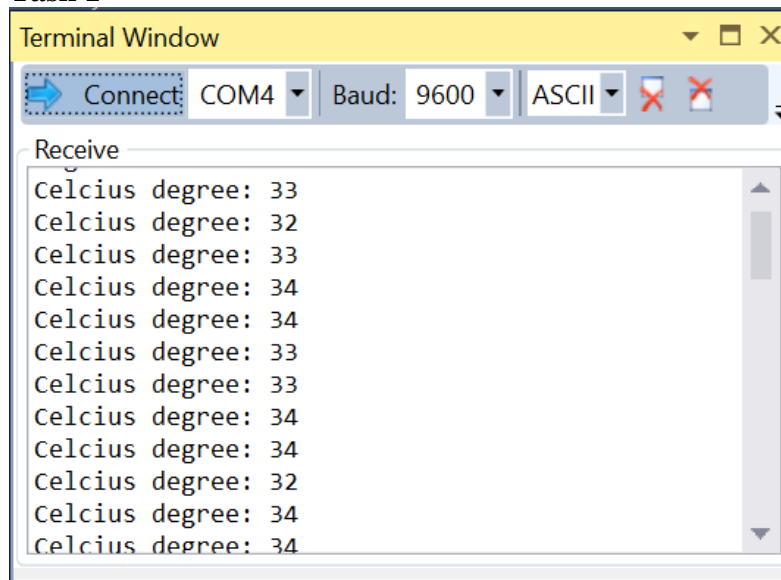


Task 3



4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1



Task 2

Terminal Window

Receive _____

```

    Done executing task "RunOutputFileVerifyTask".
    Done building target "CoreBuild" in project "DA5_TASK2.cproj".
    Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
    Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Meral\Do
    Done building target "Build" in project "DA5_TASK2.cproj".
    Done building project "DA5_TASK2.cproj".

Build succeeded.
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====
|

```

Terminal Window

- Receive _____

[illegible]

```

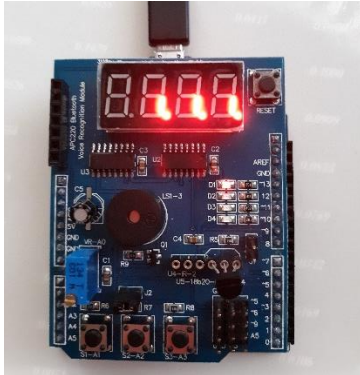
Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "DA5_TASK3.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Meral\Docum
Done building target "Build" in project "DA5_TASK3.cproj".
Done building project "DA5_TASK3.cproj".

Build succeeded.
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====
|

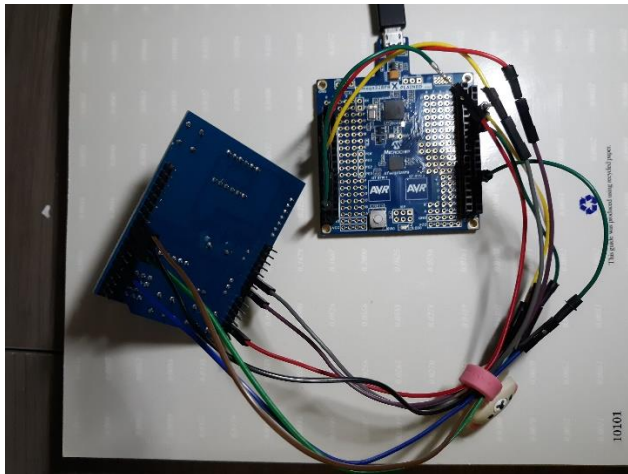
```

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

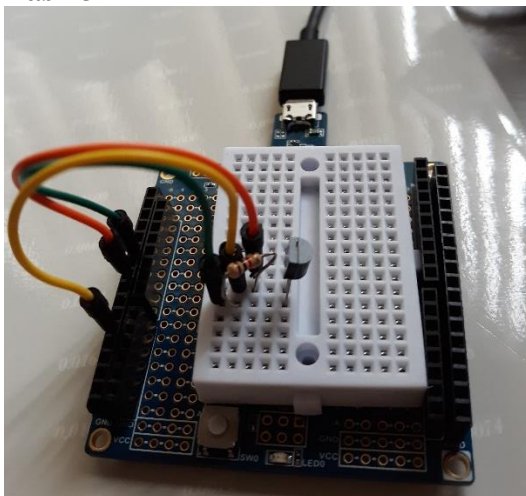
Task 1



Task 2



Task 3



6. VIDEO LINKS OF EACH DEMO

Task 1

<https://www.youtube.com/watch?v=LOpEfzyZpzU>

task 2

<https://www.youtube.com/watch?v=QG6HupvmTyA>

task 3

<https://www.youtube.com/watch?v=AU19JpWueyE>

7. GITHUB LINK OF THIS DA

https://github.com/MeralAbuJaser/Submission_da/tree/master/DA5

“This assignment submission is my own, original work”.

Meral Abu-Jaser