CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Robert Sander

Student #: 5002102412

Student Email: sander1@unlv.nevada.edu

Primary Github address: <https://github.com/sanderUNLV/submission_DA.git>

Youtube link: <https://youtu.be/Ajpad1mIYJk>

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1**

// Assignment: DA4B Task 1

// Author : Robert Sander

#define F\_CPU 16000000UL // Frequency of Xplained Mini (16MHz)

#include <avr/io.h> // Standard AVR Library

#include <stdio.h> // AVR library containing printf functions

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define BAUDRATE 9600 // Baudrate in Bits per second (bps)

#define BAUD\_PRESCALLER ((F\_CPU / (BAUDRATE \* 16UL)) - 1) // Baudrate Prescaler UBRR0H:UBRR0L

//Declaration of our functions

void Timer1\_init(void); // Function to Initialize Timer1 properties

void adc\_init(void); // Function to initialize Analog to Digital Converter

void read\_adc(void); // Function to read temperature received from ADC

void USART\_init(void); // Function to initialize USART

unsigned char USART\_receive(void); // Function to receive Serial data from UDR0

void USART\_send(unsigned char data); // Function to send individual char data into UDR0

void USART\_putstring(char\* StringPtr); // Function to break string into individual chars and send

volatile double adc\_val; // Stores ADC Value representing Analog Voltage

char outs[20]; // 'outs[]' used to store integer and float values into array of chars size 20

volatile double volt; // voltage 0-4.75

volatile double percent; // percentage of voltage 0-95%

volatile int potOCR; // corresponding value of voltage out of 255 to later become new OCR value for timer

int main(void) {

DDRB |= 0x3C; //PB5-PB2 as outputs

DDRC = 0x00; // All of PC7:0 Inputs (PC0 Input ADC, PC1 Switch)

PORTC = (1<<DDC1); // Set pull-up for PC1

DDRD |= (1 << DDD6)|(1 << DDD7); // PD6 and PD7 are Outputs (DC Motor)

PORTD |= (1 << DDD6)|(0 << DDD7); // PD6 is HIGH, PD7 is LOW (DC Motor)

Timer1\_init(); // Call the Timer1/PWM initialization code

adc\_init(); // Call the ADC initialization code

USART\_init(); // Call the USART initialization code

USART\_putstring("Connected!\r\n"); // Pass 'Connected!' to function to send serial of chars

//\_delay\_ms(125); // Wait a bit

OCR1A = potOCR;

while (1) { // Infinite loop

read\_adc(); // Read value of ADC Value

snprintf(outs,sizeof(outs),"%.2f", volt); // Stores integer 'volt' into the string 'outs'

USART\_putstring(outs); // Displays the voltage out of 5V on serial monitor

USART\_putstring(" \n");

snprintf(outs,sizeof(outs),"%.1f", percent); // Stores integer 'percent' into the string 'outs'

USART\_putstring(outs); // Displays the percentage out of 5V on serial monitor

USART\_putstring("%\r\n");

snprintf(outs,sizeof(outs),"%d", potOCR); // Stores integer 'potOCR' into the string 'outs'

USART\_putstring(outs); // Displays the corresponding value for new OCR on serial monitor

//\_delay\_ms(100);

TCNT1 = 0;

while(TCNT1 != OCR1A);

PORTB = 0x0C; // PB.5 0000 0110 ---> 0000 1100 0x0C

TIFR1 |= (1 << OCF1A);

TCNT1 = 0;

while(TCNT1 != OCR1A);

PORTB = 0x18; // PB.4 0000 1100 ---> 0001 1000 0x18

TIFR1 |= (1 << OCF1A);

TCNT1 = 0;

while(TCNT1 != OCR1A);

PORTB = 0x30; // PB.3 0000 1001 ---> 0011 0000 0x30

TIFR1 |= (1 << OCF1A);

TCNT1 = 0;

while(TCNT1 != OCR1A);

PORTB = 0x24; // PB.2 0000 0110 ---> 0010 0100 0x24

TIFR1 |= (1 << OCF1A);

}

return 0;

}

//-----------------------------------------------------------------------------------------------------------

void Timer1\_init(void) { // Function to Initialize Timer0 properties

TCCR1B |= (1 << WGM12 | 0 << CS12 | 1 << CS10); // CTC MODE WITH OCR1A AS TOP, 0 PRESCALER WITH CS12=256

}

//-----------------------------------------------------------------------------------------------------------

void USART\_init(void) { // Function to Initialize USART properties

UBRR0H = (uint8\_t)(BAUD\_PRESCALLER >> 8); // Store Upper Baudrate values into UBRR0H

UBRR0L = (uint8\_t)(BAUD\_PRESCALLER); // Store Lower Baudrate values into UBRR0L

UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable Receiver and Enable Transmitter

UCSR0C = (3 << UCSZ00); // Set UCSZ02:1 as 8-bit character data

}

//-----------------------------------------------------------------------------------------------------------

unsigned char USART\_receive(void) { // Function to receive ASCII value from UDR0

while (!(UCSR0A & (1 << RXC0))); // Keep Checking until RXC0 is 'High' to break loop

return UDR0; // Return received serial into unsigned char data

}

//-----------------------------------------------------------------------------------------------------------

void USART\_send(unsigned char data) { // Function to transmit ASCII value into UDR0

while (!(UCSR0A & (1 << UDRE0))); // Keep Checking until UDRE0 data register 'High' to break loop

UDR0 = data; // Store unsigned char serial data into UDR0

}

//-----------------------------------------------------------------------------------------------------------

void USART\_putstring(char\* StringPtr) { // Function to break string into chars, then USART\_send()

while (\*StringPtr != 0x00) { // Keep Looping until String Completed (null/0-bits)

USART\_send(\*StringPtr); // Send the unsigned char value pointed by the string pointer

StringPtr++; // Increment pointer to next char array location

}

}

//-----------------------------------------------------------------------------------------------------------

void adc\_init(void) {

DIDR0 = 0x3F; // Disable Digital Input

ADMUX = (1<<REFS1)|(1<<REFS0)| // Reference Selection Bits, AVcc - External cap at AREF

(0<<ADLAR)| // ADC Left Adjust Result for 10-bit result

(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0);// Analog Channel Selection Bits 'ADC0' (PC0)

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)|(0<<ADPS1)|(1<<ADPS0); // ADC Prescaler Select Bits '32'

}

//-----------------------------------------------------------------------------------------------------------

void read\_adc(void) {

unsigned char i = 4; // Set 'i' for iterations

adc\_val = 0; // set float 'adc\_val'

while (i--) { // Decrement 'i' until 4 samples take

ADCSRA |= (1<<ADSC); // If ADSC is high (ADC Start Conversion)...

while (ADCSRA & (1<<ADSC)); // Start the ADC Conversion

adc\_val += ADC; // Store the analog value on of current adc\_val

\_delay\_ms(100); // delay 50ms for sampling

}

adc\_val = (adc\_val/4); // Average of 4 samples taken into adc\_val

//adc\_val = (adc\_val - 512)/2; // Lets 10-bit Analog voltage become 8-bit value

volt = (adc\_val\*4.75)/1023.0;

percent = (volt/5.00)\*100;

potOCR = (percent\*40000)/100;

}

1. **DEVELOPED MODIFIED CODE OF TASK 2**

// Assignment: DA4B Task 2

// Author : Robert Sander

#define *F\_CPU* 16000000UL // Frequency of Xplained Mini (16MHz)

#include <avr/io.h> // Standard AVR Library

#include <stdio.h> // AVR library containing printf functions

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define BAUDRATE 9600 // Baudrate in Bits per second (bps)

#define BAUD\_PRESCALLER ((*F\_CPU* / (BAUDRATE \* 16UL)) - 1) // Baudrate Prescaler UBRR0H:UBRR0L

//Declaration of our functions

void Timer1\_init(void); // Function to Initialize Timer1 properties

void adc\_init(void); // Function to initialize Analog to Digital Converter

void read\_adc(void); // Function to read temperature received from ADC

void USART\_init(void); // Function to initialize USART

unsigned char USART\_receive(void); // Function to receive Serial data from UDR0

void USART\_send(unsigned char data); // Function to send individual char data into UDR0

void USART\_putstring(char\* StringPtr); // Function to break string into individual chars and send

volatile double adc\_val; // Stores ADC Value representing Analog Voltage

char outs[20]; // 'outs[]' used to store integer and float values into array of chars size 20

volatile double volt; // voltage 0-4.75

volatile double percent; // percentage of voltage 0-95%

volatile int potOCR; // corresponding value of voltage out of 255 to later become new OCR value for timer

int main(void) {

ICR1 = 4999; //fPWM=50Hz (Period = 20ms Standard).

DDRB |= 0x3E; //PB5-PB2 as outputs 0011 1110

DDRC = 0x00; // All of PC7:0 Inputs (PC0 Input ADC, PC1 Switch)

PORTC = (1<<DDC1); // Set pull-up for PC1

DDRD |= (1 << DDD6)|(1 << DDD7); // PD6 and PD7 are Outputs (DC Motor)

PORTD |= (1 << DDD6)|(0 << DDD7); // PD6 is HIGH, PD7 is LOW (DC Motor)

Timer1\_init(); // Call the Timer1/PWM initialization code

adc\_init(); // Call the ADC initialization code

USART\_init(); // Call the USART initialization code

USART\_putstring("Connected!\r\n"); // Pass 'Connected!' to function to send serial of chars

//\_delay\_ms(125); // Wait a bit

while (1) { // Infinite loop

read\_adc(); // Read value of ADC Value

*snprintf*(outs,sizeof(outs),"%.2f", volt); // Stores integer 'volt' into the string 'outs'

USART\_putstring(outs); // Displays the voltage out of 5V on serial monitor

USART\_putstring(" \n");

*snprintf*(outs,sizeof(outs),"%.1f", percent); // Stores integer 'percent' into the string 'outs'

USART\_putstring(outs); // Displays the percentage out of 5V on serial monitor

USART\_putstring("%\r\n");

*snprintf*(outs,sizeof(outs),"%d", potOCR); // Stores integer 'potOCR' into the string 'outs'

USART\_putstring(outs); // Displays the corresponding value for new OCR on serial monitor

//\_delay\_ms(100);

OCR1A=potOCR; //0 degree around 130/40 and 180 degree around 550/580

*\_delay\_ms*(250);

}

return 0;

}

//-----------------------------------------------------------------------------------------------------------

void Timer1\_init(void) { // Function to Initialize Timer0 properties

//Configure TIMER1

TCCR1A |= (1<<COM1A1)|(1<<WGM11);

//NON Inverted PWM //PRESCALER=64 MODE 14(FAST PWM)

TCCR1B |= (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);

}

//-----------------------------------------------------------------------------------------------------------

void USART\_init(void) { // Function to Initialize USART properties

UBRR0H = (*uint8\_t*)(BAUD\_PRESCALLER >> 8); // Store Upper Baudrate values into UBRR0H

UBRR0L = (*uint8\_t*)(BAUD\_PRESCALLER); // Store Lower Baudrate values into UBRR0L

UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable Receiver and Enable Transmitter

UCSR0C = (3 << UCSZ00); // Set UCSZ02:1 as 8-bit character data

}

//-----------------------------------------------------------------------------------------------------------

unsigned char USART\_receive(void) { // Function to receive ASCII value from UDR0

while (!(UCSR0A & (1 << RXC0))); // Keep Checking until RXC0 is 'High' to break loop

return UDR0; // Return received serial into unsigned char data

}

//-----------------------------------------------------------------------------------------------------------

void USART\_send(unsigned char data) { // Function to transmit ASCII value into UDR0

while (!(UCSR0A & (1 << UDRE0))); // Keep Checking until UDRE0 data register 'High' to break loop

UDR0 = data; // Store unsigned char serial data into UDR0

}

//-----------------------------------------------------------------------------------------------------------

void USART\_putstring(char\* StringPtr) { // Function to break string into chars, then USART\_send()

while (\*StringPtr != 0x00) { // Keep Looping until String Completed (null/0-bits)

USART\_send(\*StringPtr); // Send the unsigned char value pointed by the string pointer

StringPtr++; // Increment pointer to next char array location

}

}

//-----------------------------------------------------------------------------------------------------------

void adc\_init(void) {

DIDR0 = 0x3F; // Disable Digital Input

ADMUX = (1<<REFS1)|(1<<REFS0)| // Reference Selection Bits, AVcc - External cap at AREF

(0<<ADLAR)| // ADC Left Adjust Result for 10-bit result

(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0);// Analog Channel Selection Bits 'ADC0' (PC0)

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)|(0<<ADPS1)|(1<<ADPS0); // ADC Prescaler Select Bits '32'

}

//-----------------------------------------------------------------------------------------------------------

void read\_adc(void) {

unsigned char i = 4; // Set 'i' for iterations

adc\_val = 0; // set float 'adc\_val'

while (i--) { // Decrement 'i' until 4 samples take

ADCSRA |= (1<<ADSC); // If ADSC is high (ADC Start Conversion)...

while (ADCSRA & (1<<ADSC)); // Start the ADC Conversion

adc\_val += ADC; // Store the analog value on of current adc\_val

*\_delay\_ms*(100); // delay 50ms for sampling

}

adc\_val = (adc\_val/4); // Average of 4 samples taken into adc\_val

//adc\_val = (adc\_val - 512)/2; // Lets 10-bit Analog voltage become 8-bit value

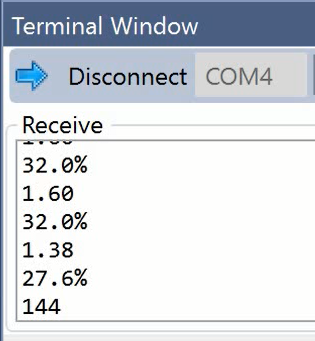
volt = (adc\_val\*5.0)/1023.0;

percent = (volt/5.0)\*100.0;

potOCR = (volt/5.0)\*524.0;

}

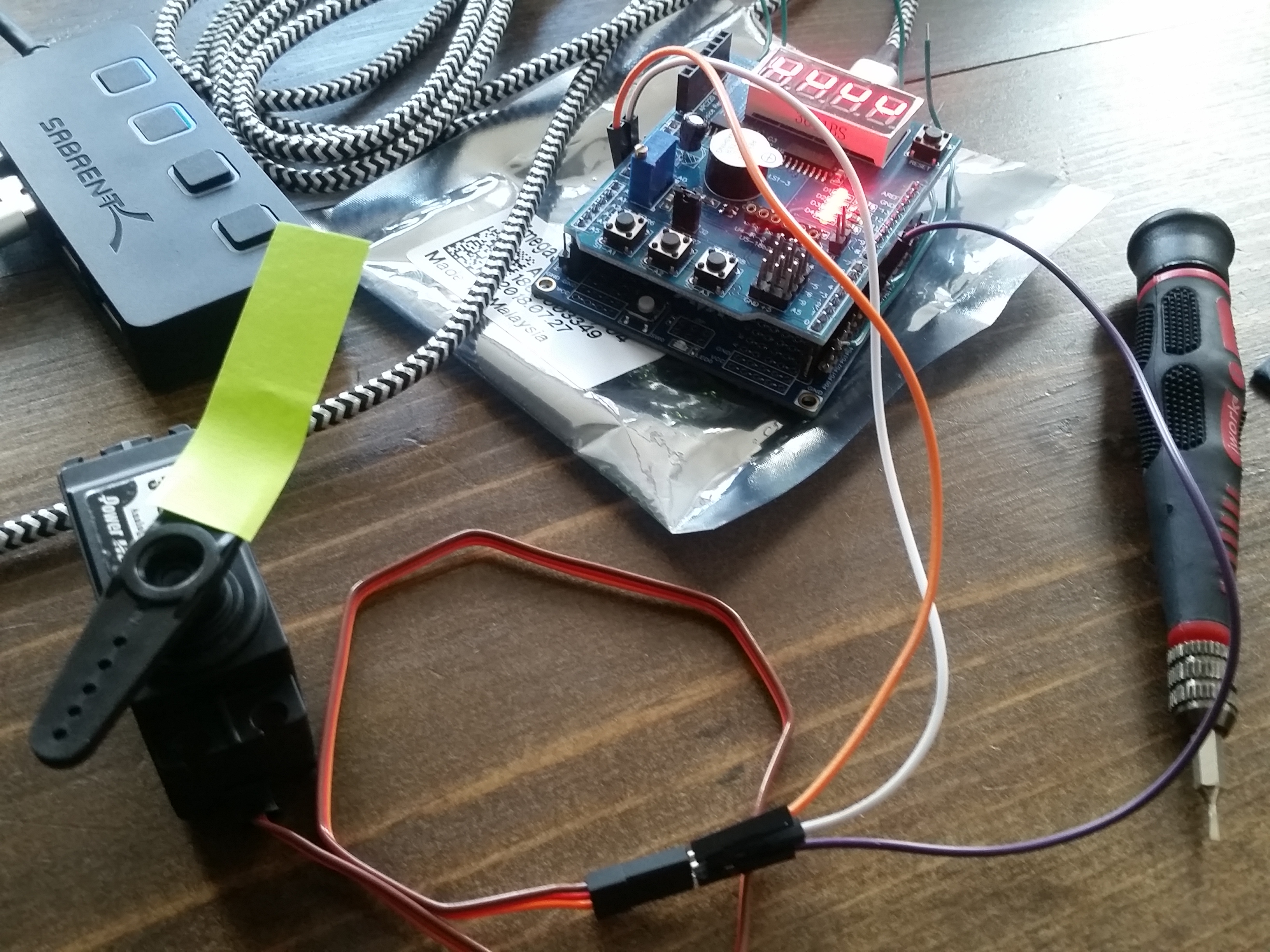
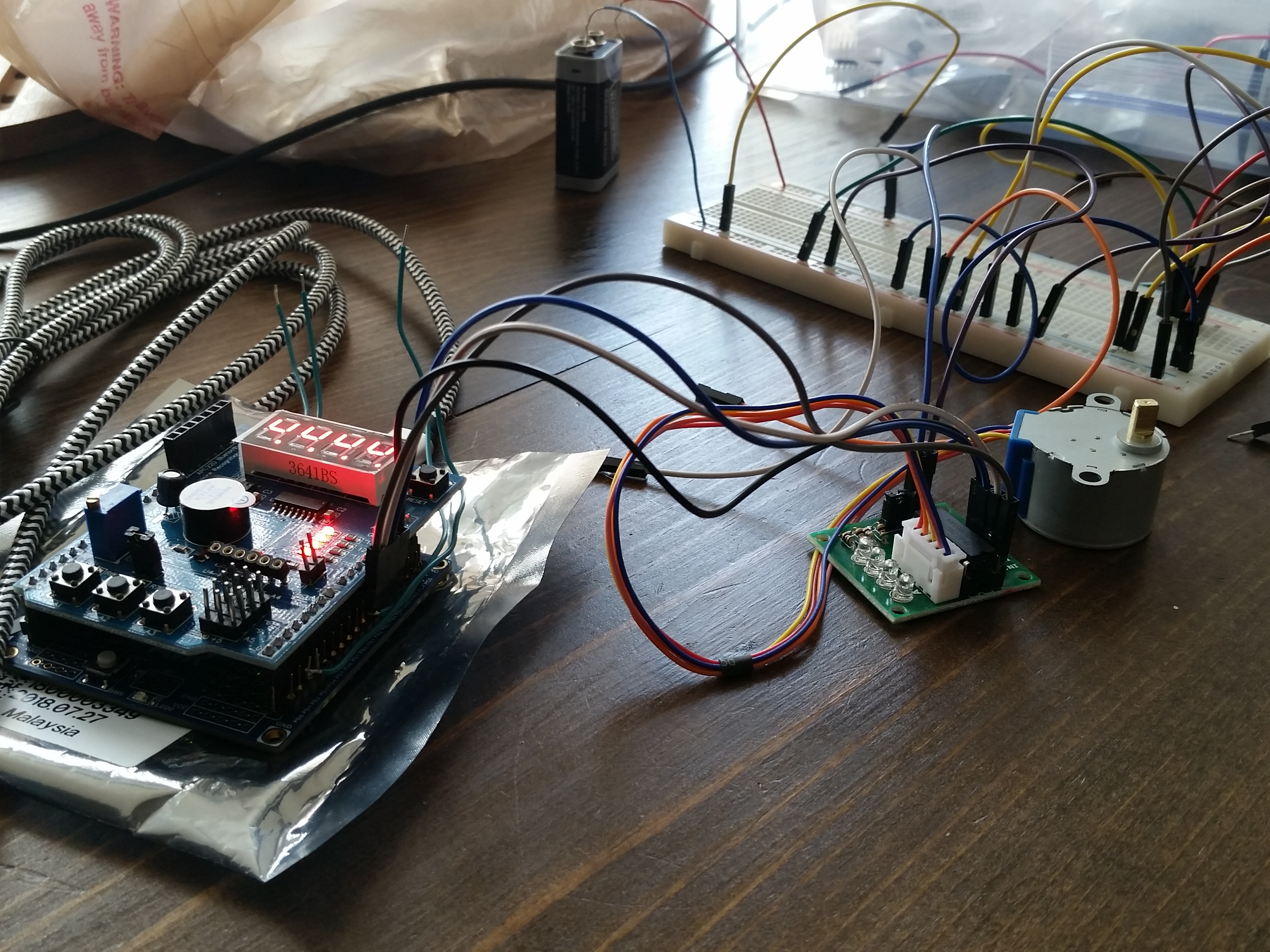
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



*Figure 2 – Servo Motor Output*

*Figure 1 – Stepper Motor Output*

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



*Figure 4 – Servo Motor Setup*

*Figure 3 – Stepper Motor Setup*

1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/Ajpad1mIYJk>

1. **GITHUB LINK OF THIS DA**

<https://github.com/sanderUNLV/submission_DA.git>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

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

-Robert Sander