

Design Assignment 5

Student Name: Abraham Garcia

Student #: 5005262049

Student Email: garci11@unlv.nevada.edu

Primary Github address: https://github.com/SON-Abe/submission_da.git

Directory: submission_da/Design_Assignments/DA5

Video Playlist: [DA5](#)

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

- Microchip Studio Debugger
- Microchip Studio Terminal Window
- Microchip Studio Simulator
- ATmega328PB Microcontroller
- SG90 Micro Servo
- mini breadboard
- usb
- Male to Male wires
- Female to Male wires

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

ULTRASONIC:

```
/*
 * ATmega_UltrasonicHCSR05.c
 *
 * Created: 6/29/2021 11:48:54 PM
 * Author : VenkatesanMuthukumar
 */

#define F_CPU 16000000UL

#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>

#define BAUDRATE 9600
#define BAUD_PRESCALER (((F_CPU / (BAUDRATE * 16UL)) - 1)
char buffer[5]; //Output of the itoa function

void USART_init(void){

    UBRR0H = (uint8_t)(BAUD_PRESCALER>>8);
    UBRR0L = (uint8_t)(BAUD_PRESCALER);
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
    UCSR0C = (3<<UCSZ00);
}
```

```

void USART_send( unsigned char data){

    while( !(UCSR0A & (1<<UDRE0)));
    UDR0 = data;

}

void USART_putstring(char* StringPtr){

    while(*StringPtr != 0x00){
        USART_send(*StringPtr);
        StringPtr++;
    }

}

#define Trigger_pin      PB1 /* Trigger pin */

int TimerOverflow = 0;

ISR(TIMER1_OVF_vect)
{
    TimerOverflow++;           /* Increment Timer Overflow count */
}

int main(void)
{
    char string[10];
    long count;
    double distance;

    DDRB = 0x02;                /* Make trigger pin as output */
    /* PB0 is the Echo Pin & PB1 is the Trigger in */
    USART_init();

    sei();                      /* Enable global interrupt */
    TIMSK1 = (1 << TOIE1);   /* Enable Timer1 overflow interrupts */
    TCCR1A = 0;                 /* Set all bit to zero Normal operation
*/

```

```

while(1)
{
    PORTB |= (1 << Trigger_pin);/* Give 10us trigger pulse on
trig. pin to HC-SR04 */
    _delay_us(10);
    PORTB &= (~(1 << Trigger_pin));

    TCNT1 = 0;                  /* Clear Timer counter */
    TCCR1B = 0x41;              /* Setting for capture rising edge, No
pre-scaler*/
    TIFR1 = 1<<ICF1;           /* Clear ICP flag (Input Capture
flag) */
    TIFR1 = 1<<TOV1;           /* Clear Timer Overflow flag */

    /*Calculate width of Echo by Input Capture (ICP) on PortD
PD6 */

    while ((TIFR1 & (1 << ICF1)) == 0); /* Wait for rising edge
*/
    TCNT1 = 0;                  /* Clear Timer counter */
    TCCR1B = 0x01;              /* Setting for capture falling edge, No
pre-scaler */
    TIFR1 = 1<<ICF1;           /* Clear ICP flag (Input Capture
flag) */
    TIFR1 = 1<<TOV1;           /* Clear Timer Overflow flag */
    TimerOverflow = 0; /* Clear Timer overflow count */

    while ((TIFR1 & (1 << ICF1)) == 0); /* Wait for falling edge
*/
    count = ICR1 + (65535 * TimerOverflow); /* Take value of
capture register */
    /* 8MHz Timer freq, sound speed =343 m/s, calculation
mentioned in doc. */
    distance = (double)count / (58*16);

    dtostrf(distance, 2, 2, string);/* Convert distance into
string */
    strcat(string, " cm   ");
    USART_putstr("Dist = ");
}

```

```
    USART_putstring(string); /* Print distance on Terminal */
    _delay_ms(200);
}
}
```

SERVO:

```
#include <Servo.h>

//-----
//servo setup
//Servo Pin must have PWM
const int servoPin = 8;
Servo myServo;
//-----

//-----
//ultrasonic sensor setup
const int trigPin = 10;
const int echoPin = 11;
long duration;
int distance;
//-----


void setup() {

    Serial.begin(9600);

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    myServo.attach(servoPin);
}

void loop() {

    //-----
    //servo move forward
    for(int i=10;i<=170;i++) // servo position 10-170
    {
        myServo.write(i);
    }
}
```

```

        delay(30);
        distance = measureDistance();
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance );
        Serial.print(".");
    }

//-----


//-----


//servo move backward
for(int i=170;i>10;i--) // servo position 170-10
{
    myServo.write(i);
    delay(40);
    distance = measureDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance );
    Serial.print(".");
}

//-----


}

int measureDistance(){

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);

    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance= duration*0.034/2;
    return distance;
}

```

3. DEVELOPED/MODIFIED CODE OF TASK 3/A

```

#define F_CPU 16000000UL           // 16MHz
clock

```

```

#define BAUDRATE 9600 //  

baudrate  

#define BAUD_PRESCALE (((F_CPU / (BAUDRATE * 16UL)) - 1)  

//BAUD_PRESCALEFORMULA  

#define Trigger_pin PINB1  

//trigger pin  

#include <stdio.h>  

#include <stdlib.h>  

#include <avr/io.h>  

#include <util/delay.h>  

#include <avr/interrupt.h>  

#include <string.h>  
  

int TimerOverflow = 0;  

char string[20];  

char angle_str[10];  

char dist_str[10];  

unsigned long count;  

int distance;  

int angle;  
  

void usart_init(void)  

{  

    UBRR0H = (uint8_t) (BAUD_PRESCALE >> 8); // LOAD UBRR0 HIGH 8  

BITS  

    UBRR0L = (uint8_t) (BAUD_PRESCALE); // LOAD UBRR0 LOW  

BITS  

    UCSR0B = (1 << TXEN0); // USART TRANSMITTER  

AND RECEIVER ENABLED  

    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // 8 DATA BITS, 1  

STOP BIT, NO PARITY  

}  
  

void usart_send (unsigned char ch)  

{  

    while (!(UCSR0A & (1<<UDRE0))); // WAIT TILL UDR0 IS  

EMPTY  

    UDR0 = ch; // TRANSMIT  

CHARACTER  

}

```

```

void usart_print(char* ChArrPtr)
{
    while ((*ChArrPtr) != '\0') // WHILE CHAR ARRAY
ISNT EMPTY
    {
        while (! (UCSR0A & (1 << UDRE0))); // WAIT TILL UDR0 IS
EMPTY
        UDR0 = *ChArrPtr; // TRANSMIT CHAR
        ChArrPtr++; // MOVE TO NEXT CHAR
IN ARRAY
    }
}

//COUNTER FOR ULTRASONIC
ISR(TIMER1_OVF_vect)
{
    TimerOverflow++; // Increment Timer
Overflow count
}

//ULTRASONIC FUNCTION
void ultrasonic()
{

    PORTB |= (1 << Trigger_pin); // 10us TRIGGER
PULSE
    _delay_us(10); // DELAY
    PORTB &= (~(1 << Trigger_pin)); // FALLING EDGE

    TCNT1 = 0; // CLEAR TIMER1
COUNTER
    TCCR1B = 0x41; // CAPTURE AT RISING
EDGE SETUP
    TIFR1 = 1<<ICF1; // CLEAR INPUT
CAPTURE FLAG FOR TIMER1
    TIFR1 = 1<<TOV1; // CLEAR TIMER1
OVERFLOW
}

```

```

while ((TIFR1 & (1 << ICF1)) == 0); // WAITS FOR RISING EDGE
TCNT1 = 0; // CLEAR TIMER1
COUNTER
TCCR1B = 0x01; // SET TO CAPTURE
FALLING EDGE
TIFR1 = 1<<ICF1; // CLEAR INPUT
CAPTURE FLAG FOR TIMER1
TIFR1 = 1<<TOV1; // CLEAR TIMER1
OVERFLOW
TimerOverflow = 0; // CLEARS TIMER
OVERFLOW COUNTER

while ((TIFR1 & (1 << ICF1)) == 0); // WAITS TILL FALLING EDGE
count = ICR1 + (65535 * TimerOverflow); // COUNT FORMULA
distance = (unsigned long)count / (58*16); // DISTANCE FORMULA

if(angle <= 9) // IF THE ANGLE IS <= 9
    dtostrf(angle, 1, 0, angle_str); // CONVERT ANGLE TO STRING WITH 1 DIGIT RESERVED FOR INTEGER PART AND 0 DIGITS FOR THE DECIMAL PART
else if(angle <= 99) // ELSE IF THE ANGLE IS <= 99
    dtostrf(angle, 2, 0, angle_str); // CONVERT ANGLE TO STRING WITH 2 DIGIT RESERVED FOR INTEGER PART AND 0 DIGITS FOR THE DECIMAL PART
else // ELSE
    dtostrf(angle, 3, 0, angle_str); // CONVERT ANGLE TO STRING WITH 3 DIGIT RESERVED FOR INTEGER PART AND 0 DIGITS FOR THE DECIMAL PART

strcat(angle_str, ","); // APPEND "," TO ANGLE_STR CHAR ARRAY

if(distance <= 9) // IF THE ANGLE IS <= 9

```



```

{
    /*MOTOR*/
    TCCR3A |= (1 << COM3A1) | (1 << COM3B1) | (1 << WGM31); // Compare Output Mode, Fast PWM: Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
    TCCR3B |= (1 << WGM33) | (1 << WGM32) | (1 << CS31); //FAST PWM & PRESCALE 8
    DDRD |= (1 << PIND0); //PWM PINS AS OUT
    ICR3 = 39999; //FWPM =
50 HZ, T = 20ms. 40k-1

    /*ULTRASONIC*/
    DDRB = (1<<DDRB1); // PB0
is the Echo Pin & PB1 is the Trigger in
    usart_init(); // INITIALIZE USART

    sei(); // ENABLE GLOBAL INTERRUPTS
    TIMSK1 = (1 << TOIE1); // ENABLE TIMER1 OVERFLOW INTERRUPTS
    TCCR1A = 0; // SET TCCR1A 0

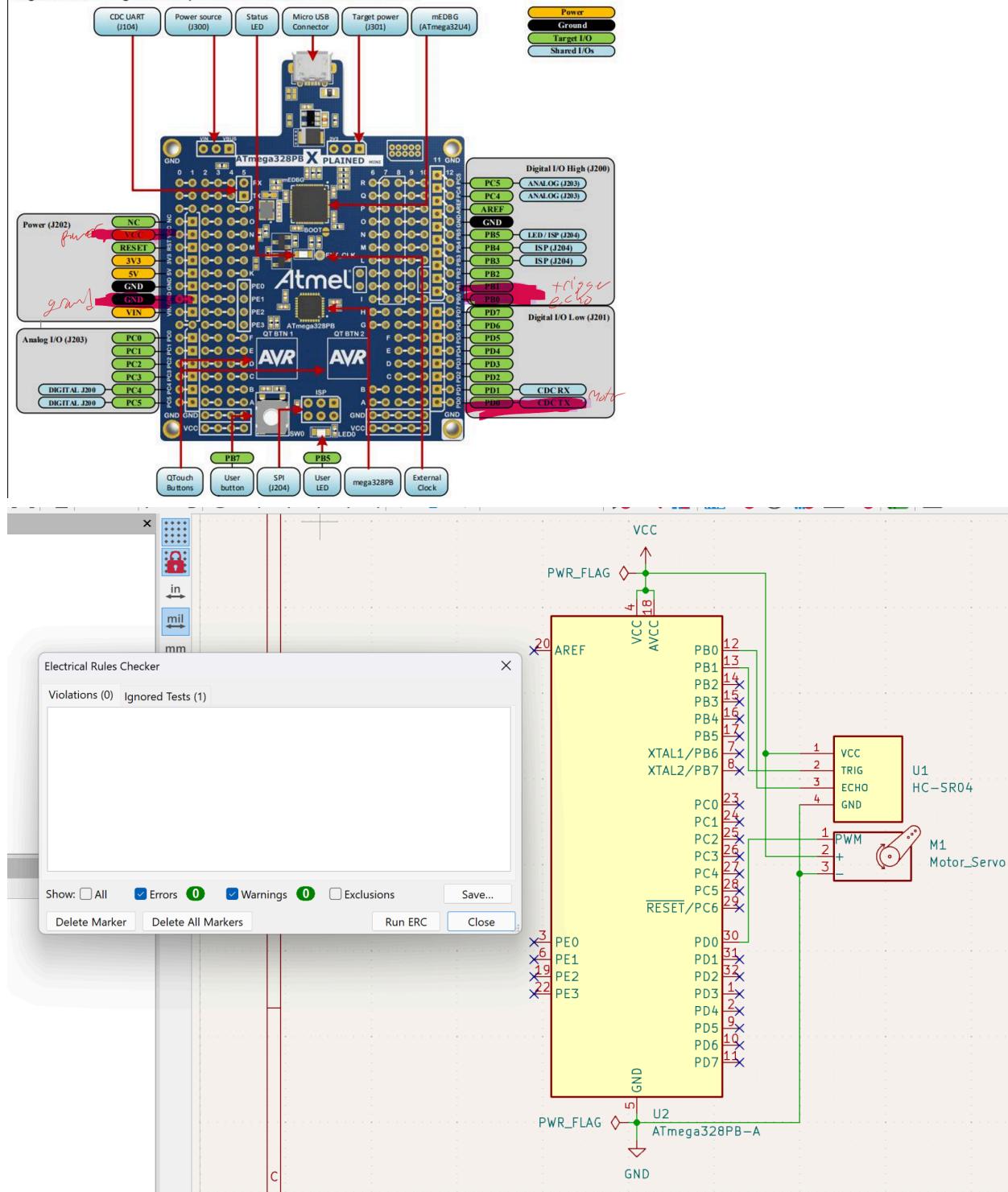
    while(1)
        motor(); // RUN THE MOTOR FUNCTION
}

```

4. SCHEMATICS

ATmega328PB Xplained Mini

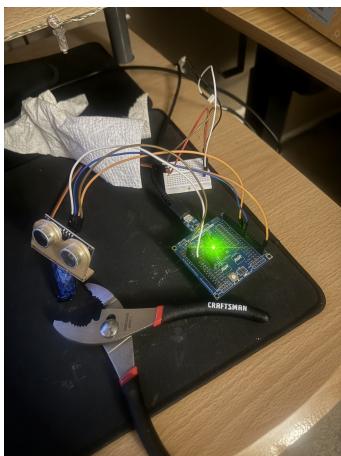
Figure 1-1. ATmega328PB Xplained Mini Headers and Connectors



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



6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



7. VIDEO LINKS OF EACH DEMO

[DA5](#)

8. GITHUB LINK OF THIS DA

[DA5](#)

Student Academic Misconduct Policy

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

"This assignment submission is my own, original work".

Abraham Garcia