CPE301 – SPRING 2023

Design Assignment 5

Student Name: David Lenzin
Student #: 2001654470
Student Email: lenzin@unlv.nevada.edu
Primary Github address: https://github.com/dlenzin15/submissions
Directory: submissions/DA5

Submit the following for all Labs:

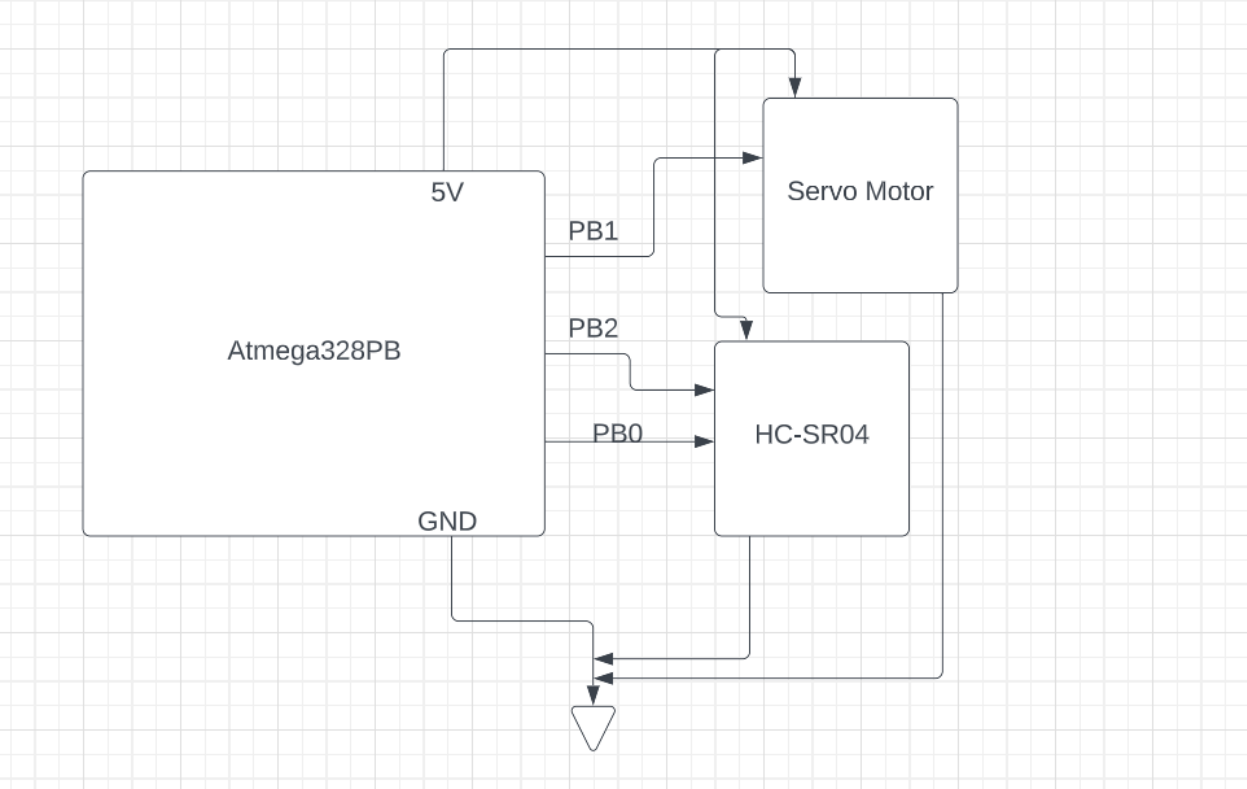
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder 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

List of Components used:

- Atmega328PB board
- HC-SR04 ultrasonic sensor
- Smraza micro servo 9G
- Breadboard

Block diagram with pins used in the Atmega328P



2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1

Insert initial code here

```
/*
 * DA5.c
 *
 * Created: 4/12/2023 10:31:35 AM
 * Author : david
 */

// Definitions
#define F_CPU 16000000UL
#define TRIGGER_PIN PB2
#define ECHO_PIN PB0
#define CONTROL_PIN PB1
#define BAUD 9600
```

```c
#define MYUBRR F_CPU/16/BAUD-1

// Included Files
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>

// Function Declarations
void UART_init();
void timer_init();
void UART_transmit_string();
void Wait();
uint32_t calculateDistance();

void UART_init(unsigned int ubrr)
{
	//Set baud rate
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;

	//Enable transmitter and receiver and receiver interrupt
	UCSR0B = (1<<RXEN0) | (1<<TXEN0);

	//Set frame format: 8 bits data, 1 stop bit
	UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);

	sei();
}

void UART_transmit_string(char *data) {
	while ((*data != '\0')) {   // Check if NULL char
		while (!(UCSR0A & (1 <<UDRE0)));  // Wait for register to be
		UDR0 = *data; // Store data in the data register
		data++;
	}
}

void timer_init()
{
	//Configure TIMER1
	TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);        //NON Inverted PWM
	TCCR1B|=(1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10); //PRESCALER=64 MODE 14(FAST PWM)

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

//Simple Wait Function
void Wait()
{
	uint8_t i;
	for(i=0;i<2;i++)
	{
		_delay_loop_2(0);
	}

}
```

```c
uint32_t calculateDistance()
{
	PORTB &= (~(1<<TRIGGER_PIN));
	_delay_us(2); // Pull trigger low before pulse

	/* Give 10 ms trigger pulse on trig. pin to HC-SR04 */
	PORTB |= (1<<TRIGGER_PIN);
	_delay_ms(10);
	PORTB &= (~(1<<TRIGGER_PIN));

	// Measure duration of pulse on echoPin
	unsigned long duration = 0;
	while (!(PINB & (1 << ECHO_PIN))); // Wait for echo to go high
	while ((PINB & (1 << ECHO_PIN)))
		duration++; // Measure pulse width

	uint32_t distance= (uint32_t)duration*0.034/2;

	return distance;
}

void main()
{
	// Set data directions
	DDRB|=(1<<CONTROL_PIN) | (1<<TRIGGER_PIN);   //PWM Pins as Out
	PORTD &=  ~(1<<TRIGGER_PIN); // Pull down
	DDRB &= ~(1<<ECHO_PIN); // Set trigger pin to input

	timer_init();
	UART_init(MYUBRR);

	int angle = 0;
	OCR1A = 97;          // Initialize motor to 0 degrees

	while(1)
	{
		uint32_t distance = 0;
		char buffer[100];
		sprintf(buffer, "Distance = %d cm\n", distance);
		UART_transmit_string(buffer);

		while (OCR1A < 535)
		{
			OCR1A += 5; // Increment every 2 degrees until we reach 180 degrees
			angle += 2;
			Wait();
			distance = calculateDistance();
			sprintf(buffer, "%d,%d.", angle, distance);
			UART_transmit_string(buffer);
		}

		OCR1A = 535; // 180 degrees
		angle = 180;

		while (OCR1A > 97)
		{
			OCR1A -= 5; // decrement every 2 degrees until we reach 0 degrees
			angle -= 2;
```

```
                    Wait();
                    distance = calculateDistance();
                    sprintf(buffer, "%d,%d.", angle, distance);
                    UART_transmit_string(buffer);
                }
                OCR1A = 97; // 0 degrees
        }
}
```

3. DEVELOPED MODIFIED CODE OF TASK 2

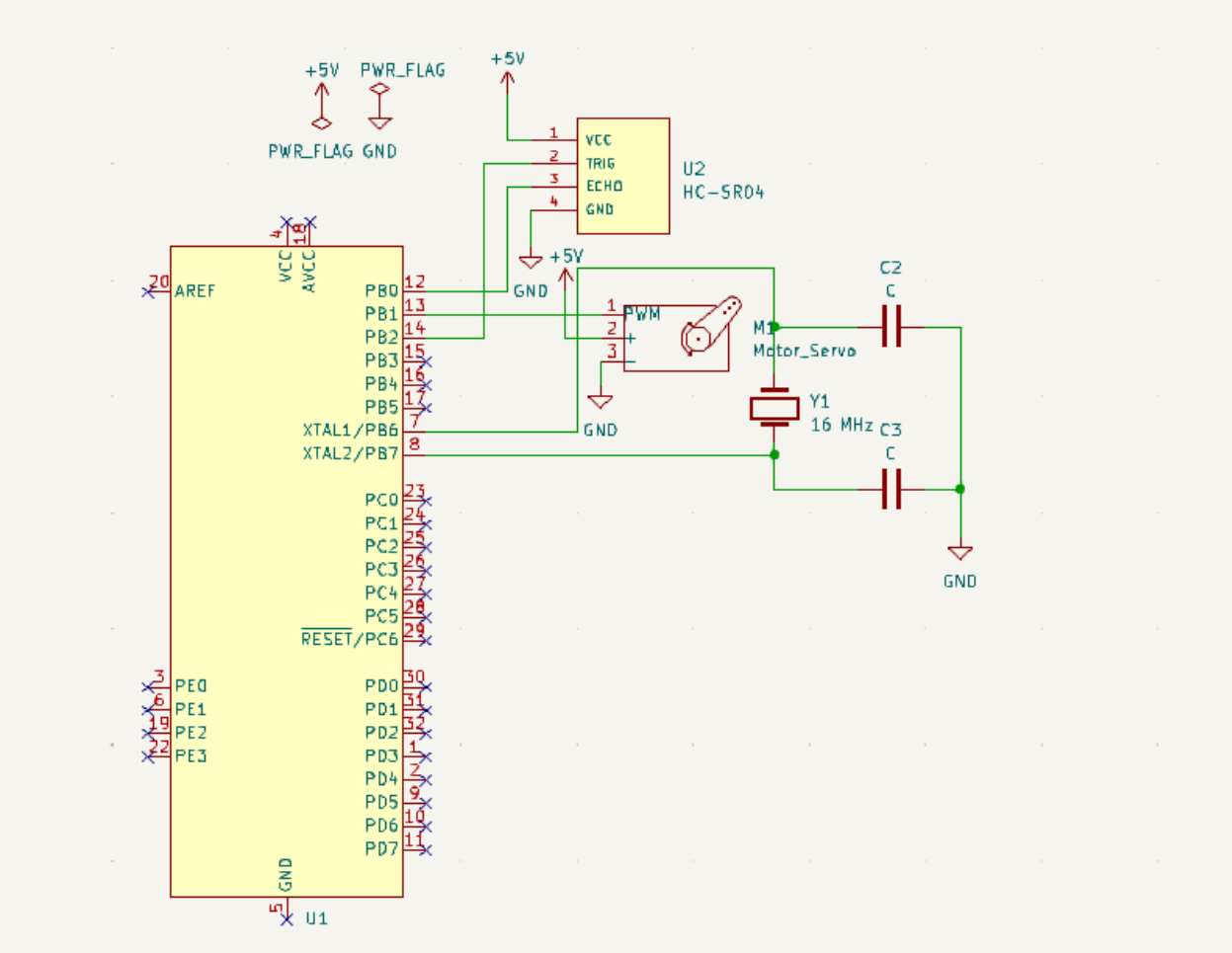
Insert only the modified sections here

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;
Serial myPort; // defines Object Serial

// Variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (1200, 700);
 smooth();
 myPort = new Serial(this,"COM3", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it reads this:
angle,distance.
}
void draw() {

  fill(98,245,31);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}
```

```
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1"
  angle= data.substring(0, index1); // read the data from position "0" to position of the variable index1 or thats the
value of the angle the Arduino Board sent into the Serial Port
  distance= data.substring(index1+1, data.length()); // read the data from position "index1" to the end of the data pr
thats the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-
(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
```

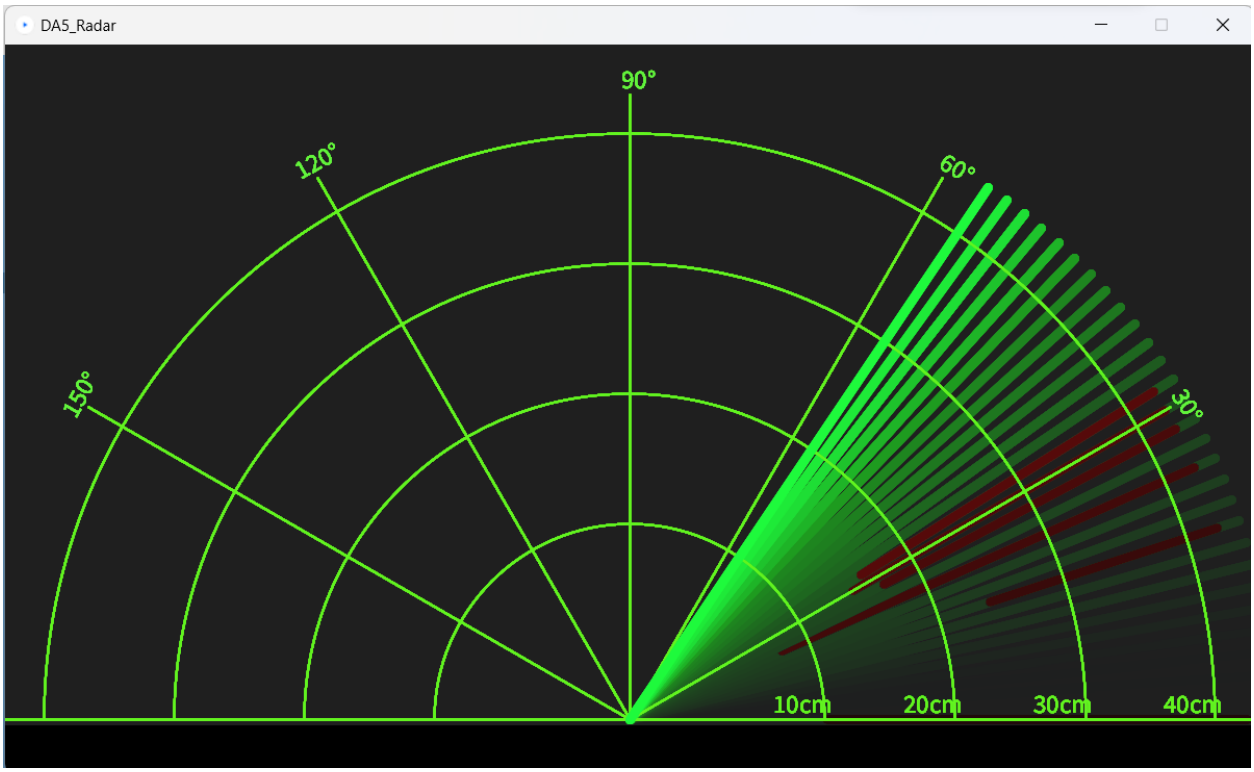
```
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle))); // draws the line
according to the angle
  popMatrix();
}
void drawText() { // draws the texts on the screen

  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);

  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30° ",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60° ",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*sin(radians(90)));
  rotate(radians(0));
  text("90° ",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120° ",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150° ",0,0);
  popMatrix();
}
```

4. SCHEMATICS



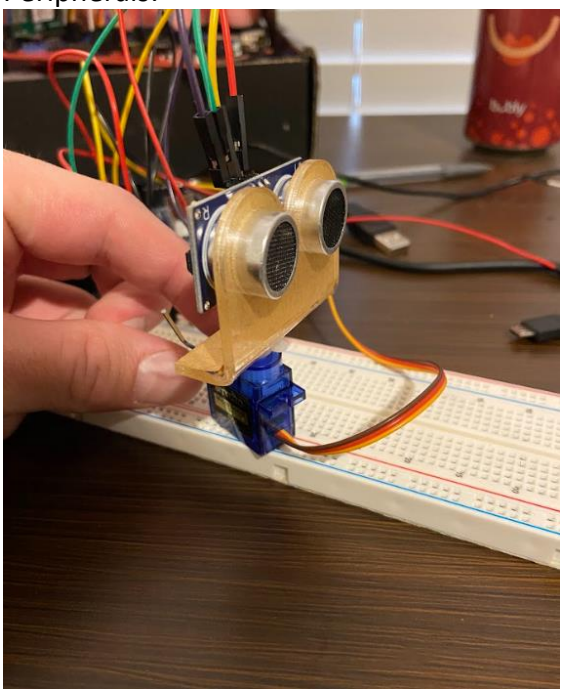
5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

- The only output on the computer is from the distance graph.

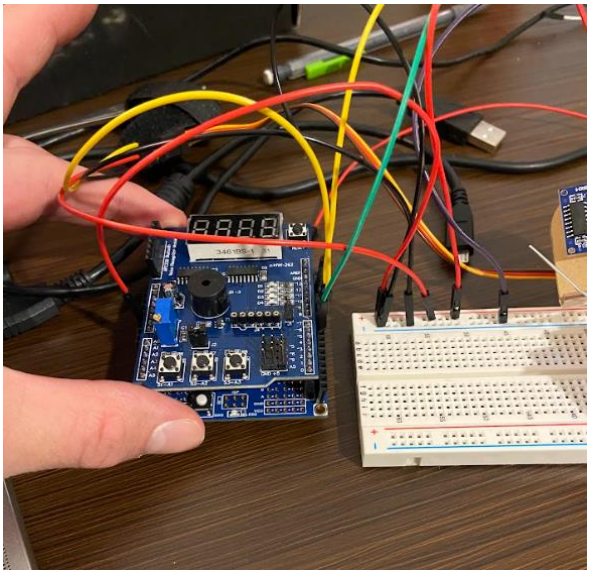


6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Peripherals:



Atmega328PB Board:



7. VIDEO LINKS OF EACH DEMO

Complete Demo: https://www.youtube.com/watch?v=E6sbZxdOmO8&ab_channel=DavidLenzin

8. GITHUB LINK OF THIS DA

https://github.com/dlenzin15/submissions/tree/main/DA5

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
David Lenzin