CPE301 – SPRING 2024

Design Assignment 5

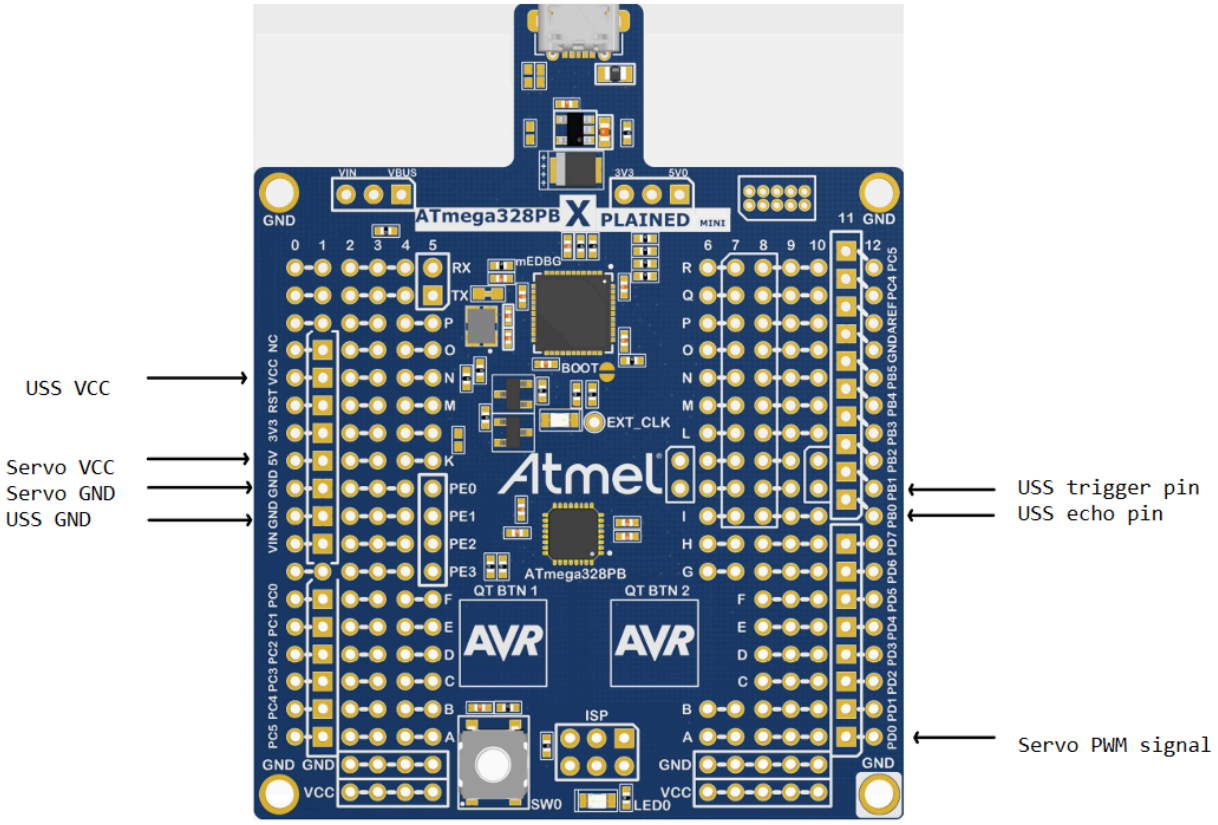
Student Name: Nathan Ramos
Student #: 5006437353
Student Email: ramosn8@unlv.nevada.edu
Primary Github address: https://github.com/n8ramos/
Directory: /atmega328pb
Video Playlist:
https://www.youtube.com/playlist?list=PL2RpCRW8TC6YOj-NnLPqqfRTV48RcUe48

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Components Used:

- MG966R servo
- HC-SR04 Ultrasonic Sensor
- atmega328pb Xplained Mini
- microchip studio 7

Block diagram with pins used in the Atmega328PB (only)



2. DEVELOPED CODE OF TASK 1

```
#define F_CPU 16000000UL //define the XTAL in the MicroController

// Ultra-Sonic Sensor
#define Trigger_pin PINB1  //This is the UltraSonic Sensors Trigger Pin

// USART output
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

// decimal to string dtostrf
#define PRECISION 0
#define MINWIDTH 1

// servo
#define SERVO180 600 // OCR1A value for 180 degrees
#define SERVO0 100 // OCR1A value for 0 degrees
#define DEGSTEP (SERVO180 - SERVO0)/180 // 1 degree step
#define STEPDELAY 50
#define WAITDELAY 1000

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

//Global Variables
const double ANGLE_SCALE = 0.72; // 180 deg divided by number of servo steps
int TimerOverflow = 0;

//Interrupt Subroutines (for USS)
ISR(TIMER1_OVF_vect)
{
	TimerOverflow++;	//Increment Timer Overflow count
}
// initialize ultrasonic sensor
void UltrasonicInit() {
	/*Ultrasonic Initialization
	PB0 is the Echo Pin & PB1 is the Trigger*/

	//GPIO Programming
	DDRB = 0x02;  //Output for Ultrasonic Trigger Pin
	DDRC = 0xFF;  //PORTC as outputs for LEDs

	//Timer 1 Initialization
	TIMSK1 = (1 << TOIE1);	//Enable Timer1 overflow interrupts
	TCCR1A = 0;			//Set all bit to zero Normal operation
```

```
	//Enable Global Interrupts
	sei();
}

// Function to determine distance from ultrasonic sensor
double distance() {
	TCCR1B |= (1 << CS10);  //Pre-Scalar, Start Timer

	/*Declare variables*/
	long count;			//var to store the received input from
ultrasonic
	double distance = 0;//var to store the received distance from the USART

	/*Receive the UltraSonic sensors values*/
	PORTB |= (1 << Trigger_pin);			//Begin Trigger
	_delay_us(10);
	PORTB &= (~(1 << Trigger_pin));			//Cease Trigger
	TCNT1 = 0;							//Clear Timer
counter
	TCCR1B = (1 << ICES1) | (1 << CS10);	//Capture rising edge, Pre-Scalar 1
	TIFR1 = (1 << ICF1) | (1 << TOV1);	// Clear ICP flag & Clear Timer Overflow
flag

	/*Calculate width of Echo by Timer 1 ICP*/
	while ((TIFR1 & (1 << ICF1)) == 0);		// Wait for rising edge
	TCNT1 = 0;							// Clear Timer
counter
	TCCR1B = (1 << CS10);					// Capture falling edge
	TIFR1 = (1 << ICF1) | (1 << TOV1);	// Clear ICP flag & 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 and
calculate width
	distance = (double)count / (58*16);		//Calculate Distance

	TCCR1B ^= (1 << CS10);  //No Pre-Scalar, Stop Timer
	return distance;			//Return distance
}

//initialize USART
void USART_init(unsigned int ubrr)
{
	//Set baud rate
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char) ubrr;
	// enable transmitter
	UCSR0B = (1<<TXEN0);
	// Set frame format: async, no parity, 1 stop bit, , 8 data bits
```

```
	UCSR0C =
(0<<UMSEL01)|(0<<UMSEL00)|(0<<UPM01)|(0<<UPM00)|(0<<USBS0)|(1<<UCSZ01)|(1<<UCSZ00);
}
//send strings thru USART
void USART_transmit(const char* data)
{
	while (*data) {
		//check if buffer is empty so that data can be written to transmit
		while (!(UCSR0A & (1 << UDRE0)));
		UDR0 = *data; //copy “data” to be sent to UDR0
		++data;
	}
}
//send characters thru USART
void USART_transmitChar(const char data)
{
	//check if buffer is empty so that data can be written to transmit
	while (!(UCSR0A & (1 << UDRE0)));
	UDR0 = data; //copy character to be sent to UDR0
}

int main()
{
	USART_init(MYUBRR);
	UltrasonicInit();
	//Configure TIMER1
	// OC3A is output to PIN D0
	TCCR3A|=(1<<COM3A1)|(1<<COM3B1)|(1<<WGM31);		// clear OC1A/OC1B on compare
match, NON Inverted PWM (period resets OC1A to high at BOTTOM)
	TCCR3B|=(1<<WGM33)|(1<<WGM32)|(1<<CS31)|(1<<CS30); // PRESCALER=64, MODE 14(FAST
PWM)

	ICR3=4999;  //fPWM=50Hz (Period = 20ms Standard).

	DDRD |= (1<<PIND0);   //PWM Pins as Out

	OCR3A = SERVO0;
	_delay_ms(WAITDELAY);

	while(1)
	{
		char strAng[20];
		char strDist[10];
		double angle;
		double dist;

		int j = 0; // counter tracking servo steps
		for(int i = SERVO0; i < SERVO180; i+= DEGSTEP) {
			_delay_ms(STEPDELAY);
			OCR3A = i; // servo position
			angle = j * ANGLE_SCALE; // calculate angle
```

```
                    dtostrf(angle, MINWIDTH, PRECISION, strAng); // turn angle into
integer string
                    USART_transmit(strAng);
                    USART_transmitChar(',');
                    ++j;

                    dist = distance(); // get distance from ultra-sonic sensor
                    dtostrf(dist, MINWIDTH, PRECISION, strDist); // turn distance into
integer string
                    USART_transmit(strDist);
                    USART_transmitChar('.');
             }
             _delay_ms(WAITDELAY);
             for(int i = SERVO180; i > SERVO0; i-= DEGSTEP) {
                    _delay_ms(STEPDELAY);
                    OCR3A = i; // servo position
                    angle = j * ANGLE_SCALE; // calculate angle
                    dtostrf(angle, MINWIDTH, PRECISION, strAng); // turn angle into
integer string
                    USART_transmit(strAng);
                    USART_transmitChar(',');
                    --j;

                    dist = distance();
                    dtostrf(dist, MINWIDTH, PRECISION, strDist); // turn distance into
integer string
                    USART_transmit(strDist);
                    USART_transmitChar('.');
             }
             _delay_ms(WAITDELAY);
      }
      return 0;
}
```

DEVELOPED CODE OF TASK 2

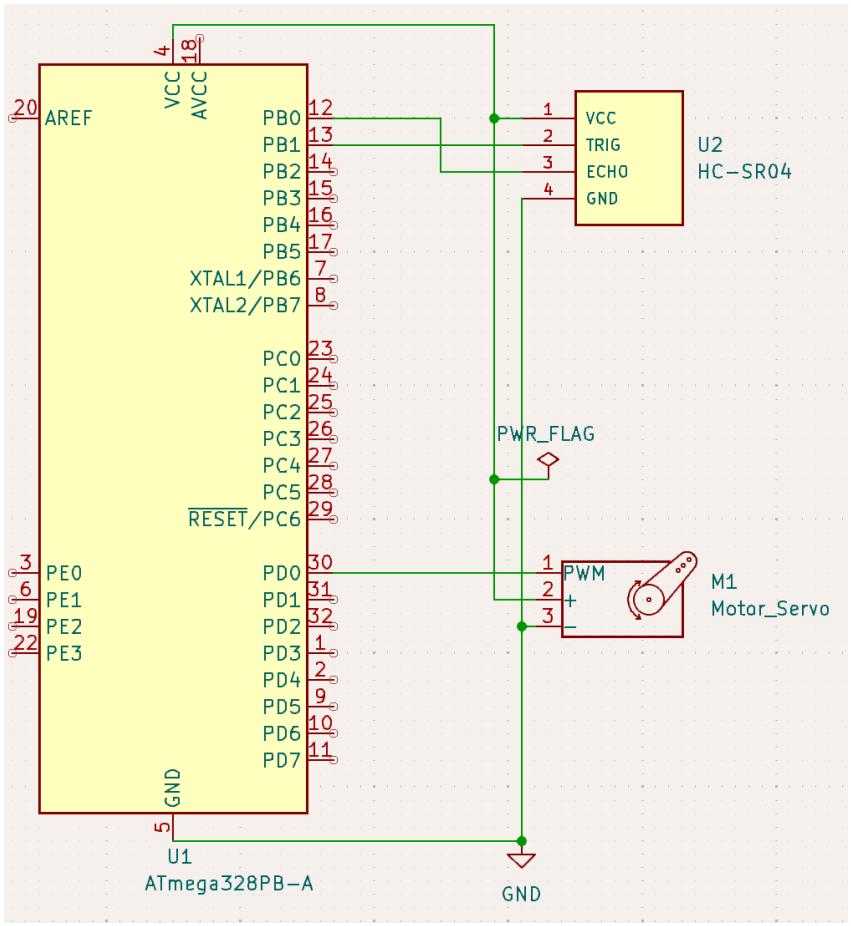
```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial
port
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {
```

```
 size (1366, 768); //CHANGE THIS TO YOUR SCREEN RESOLUTION  ---
 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(iAng
le))); // 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);
```

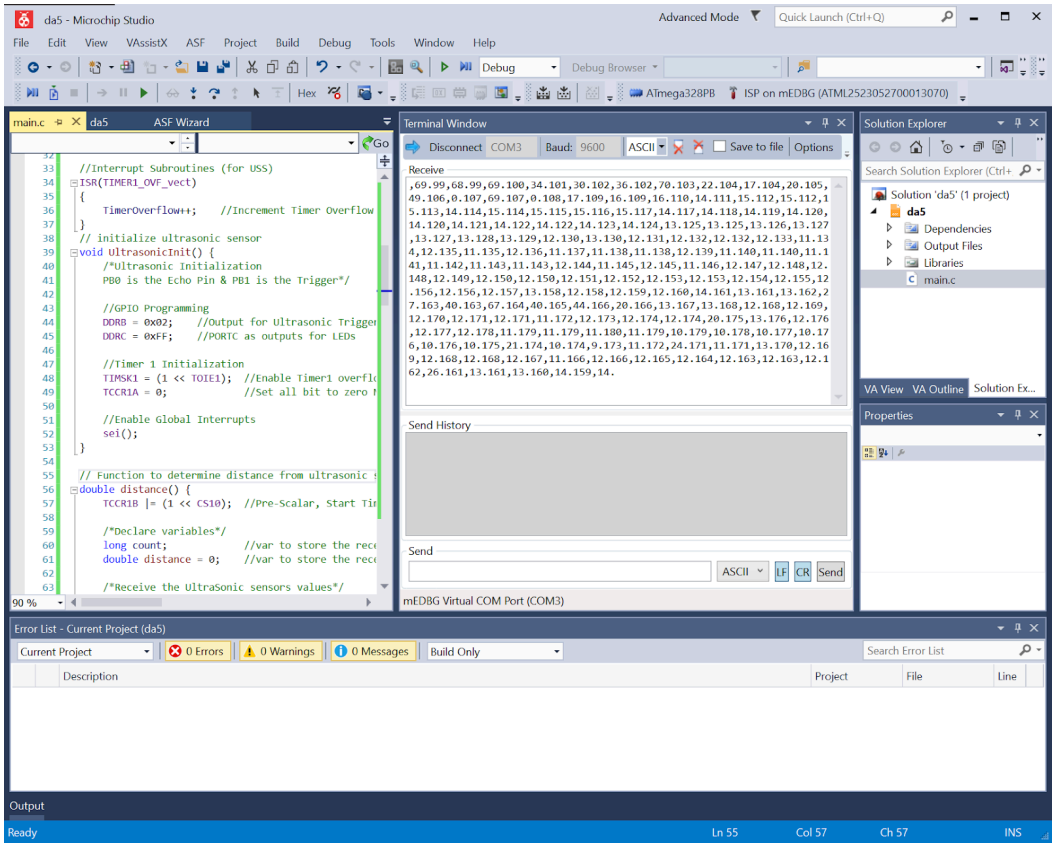
```
  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Nathan Ramos", width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" ° ", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("              " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);

translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*si
n(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*s
in(radians(150)));
  rotate(radians(-60));
  text("150° ",0,0);
  popMatrix();
}
```

3. SCHEMATICS

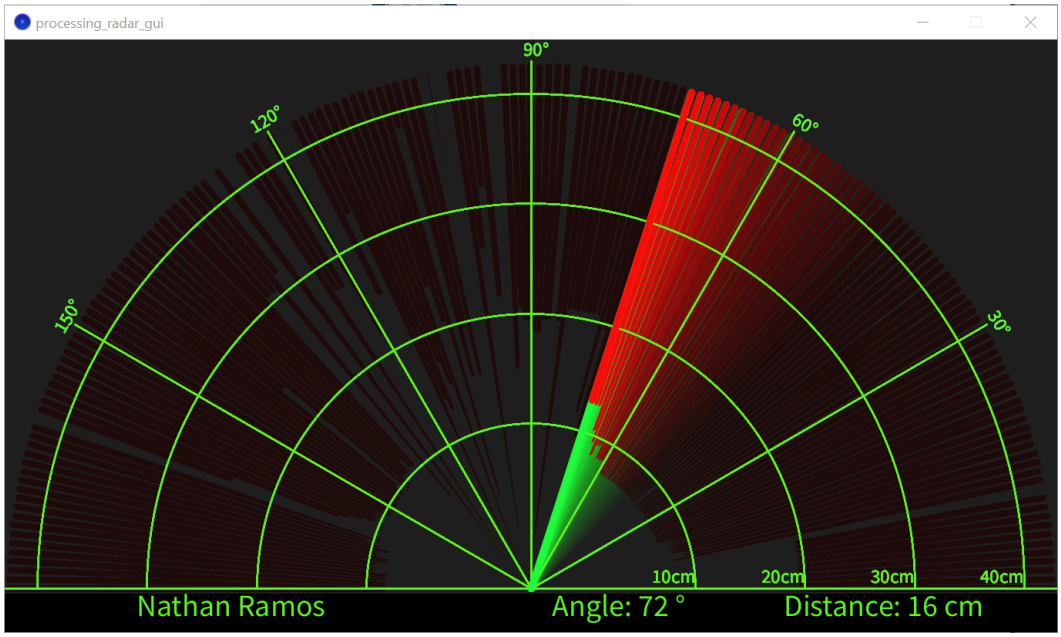


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

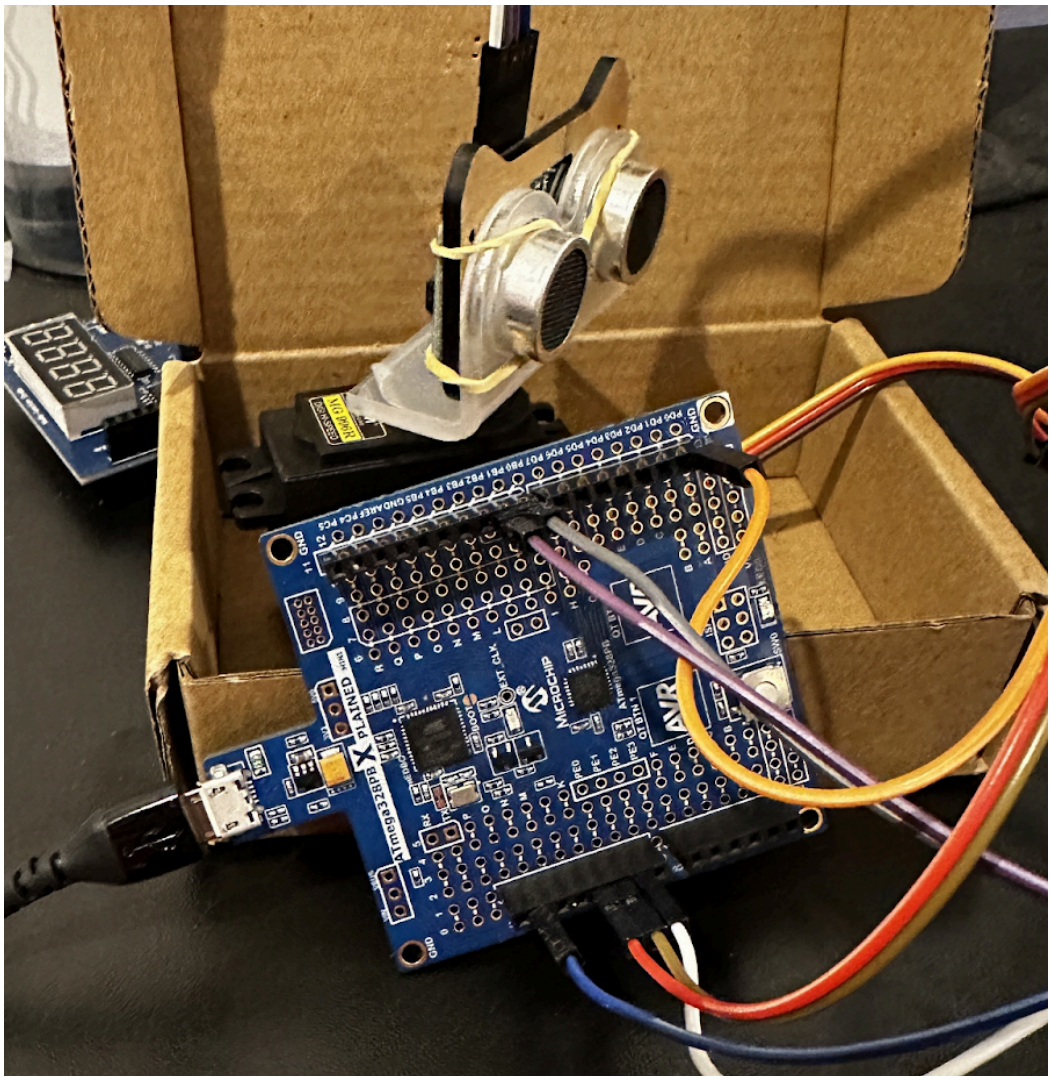
TASK 1



TASK 2



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

https://youtube.com/shorts/6ushhBqGJcY

7. GITHUB LINK OF THIS DA

https://github.com/n8ramos/atmega328pb/tree/main

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

"*This assignment submission is my own, original work*".
Nathan Ramos