Design Assignment 5

Student Name: Samuel McCormick
Student #: 1014303276
Student Email: samuel.mccormick@gmail.com
Primary Github address: https://github.com/brokenboredom/tech-muffin.git
Directory: DesignAssignments/DA5

Write, simulate, and demonstrate using Microchip Studio 7 a C code for the AVR ATMEGA328pb microcontroller that performs the following functions:

1. Mount the HC-SR04 Ultrasonic sensor on to the servo motor using the mounting plate/horn. Scan the servo motor from 0 – 180 deg. Collect the ultrasonic distance (US) distance/raw value continuously during the scan. The resolution of scan has to be less that 2.5 deg.

The HC-SR04 sensor was mounted to the servo motor using the plastic mounting bracket. The servo motor uses Timer1 to scan between 0-180 degrees by updating OCR1A to set the servo angle +2 degrees. In this case we used 535(180 degrees) - 97 (0 degrees) divided by the number of sections we needed– in this case between 72 and 90 sections (2 to 2.5 degrees). So we chose to add 5 to OCR1A every iteration. 438/72 = 6.08. 438/90 = 4.89.

The servo motor operates on a 20ms period so we used 64 prescalar and ICR1 = 4999 for approx. 50Hz. 16MHz/64/4999 ~= 50Hz.

The data was output using USART as in previous assignments. No changes to the configuration were necessary.

C code for scanning servo motor:

```
// -- DO Servo operation
      //----------------------------------------
      //Servo motor operations

      //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)
      TIMSK1 &= ~(1 << TOIE1);                          // Disable Timer1 overflow interrupts

      ICR1=4999;                                  // 16MHz/64/4999 = fPWM=50Hz (Period = 20ms Standard).
      //----------------------------------------
      // 535 = 180 degrees
      if(servoCnt > 535) {
         servoCnt = 97; // 0 degrees
         OCR1A = servoCnt;
         degree = 0;
         _delay_ms(100);
      }
      else {
         OCR1A = servoCnt;
         /*******************************************************/
         degree = degree + 2;
         servoCnt += 5;         // 535/97 = 438. 438/(180/2) ~6. 438/(180/2.5) ~= 4. We'll use 5 for ~87 segments
         _delay_ms(500);
```

```
    }
```

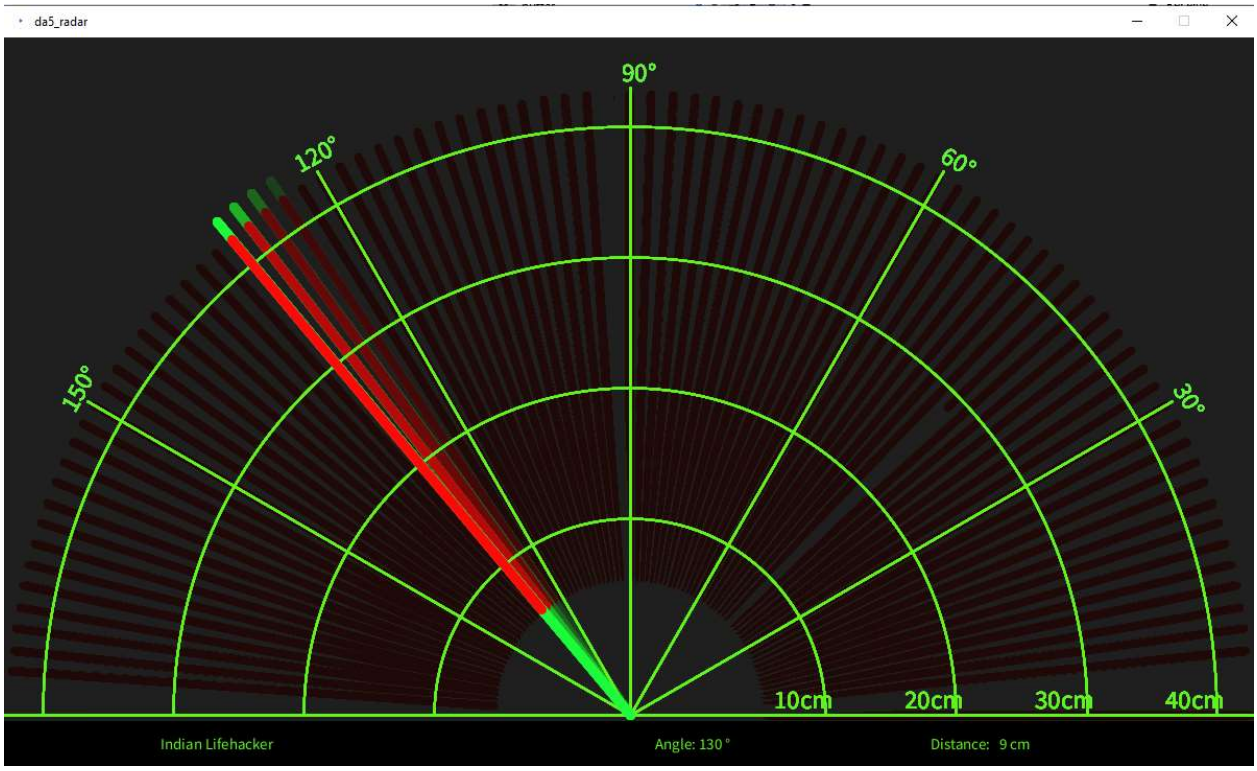
C code for parsing distance with sonic sensor:

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

      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);
      // Output should look like: "angle,distance."
      itoa(degree, angle, 10);          // String to Int angle for Processor graph
      USART_putstring(angle);
      USART_putstring(",");             // add the comma
      itoa(distance, string, 10);       // String to int distance
      USART_putstring(string);
      USART_putstring(".");             // add ending period
      _delay_ms(10);
```

2. Display your results as a two dimensional, 0 -180 deg distance graph. Update your scan after every scan range. Use the reference provided (https://tinyurl.com/2oph9dmk) to plot the two-dimensional US data.

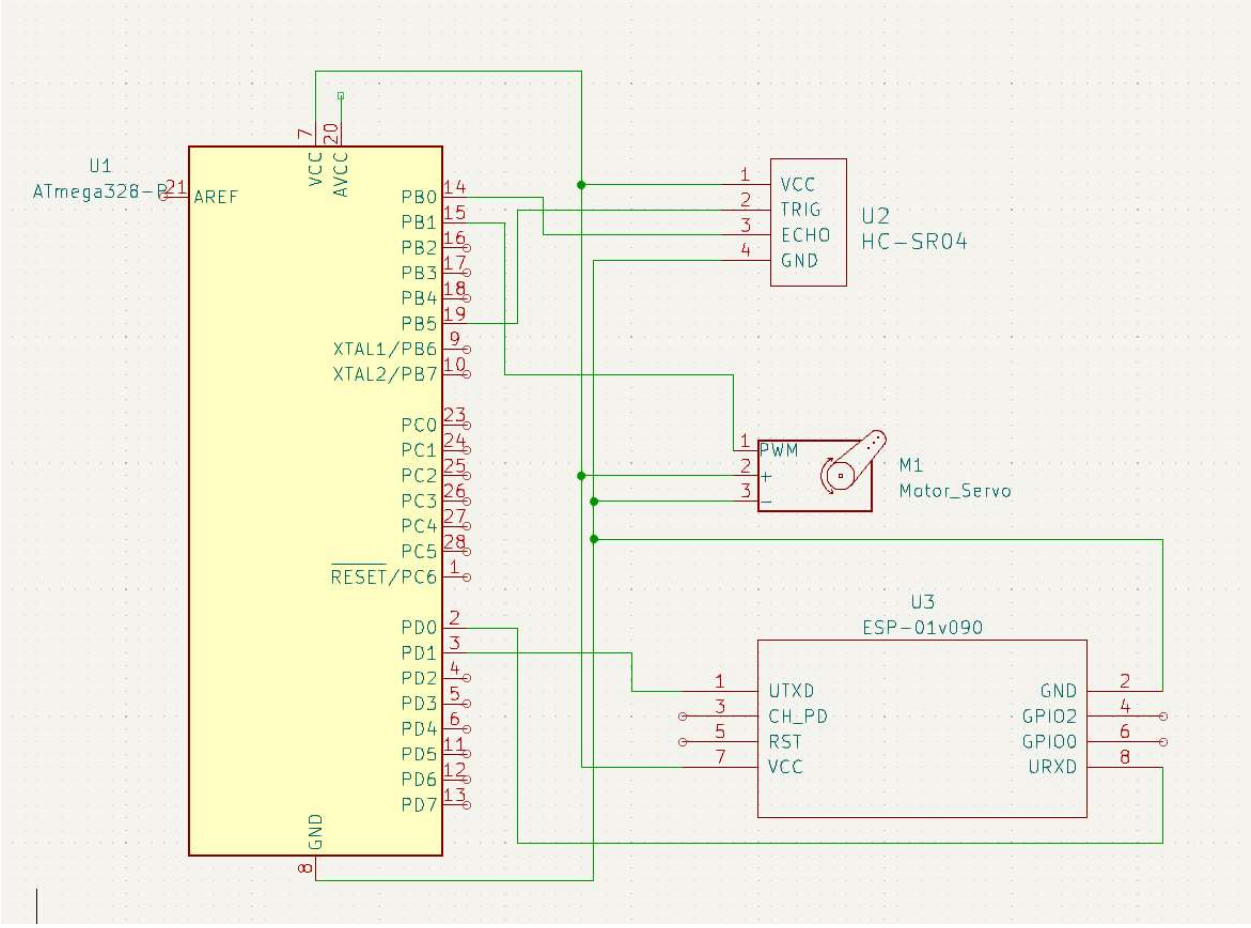
The string output from USART was updated to match the output required in the Processor sketch– expected “angle,string.” so that the radar graph would updated properly.

```
// 8MHz Timer freq, sound speed =343 m/s, calculation mentioned in doc.
      distance = (double)count / (58*16);
      // Output should look like: "angle,distance."
      itoa(degree, angle, 10);          // String to Int angle for Processor graph
      USART_putstring(angle);
      USART_putstring(",");             // add the comma
      itoa(distance, string, 10);       // String to int distance
      USART_putstring(string);
      USART_putstring(".");             // add ending period
      _delay_ms(10);
```

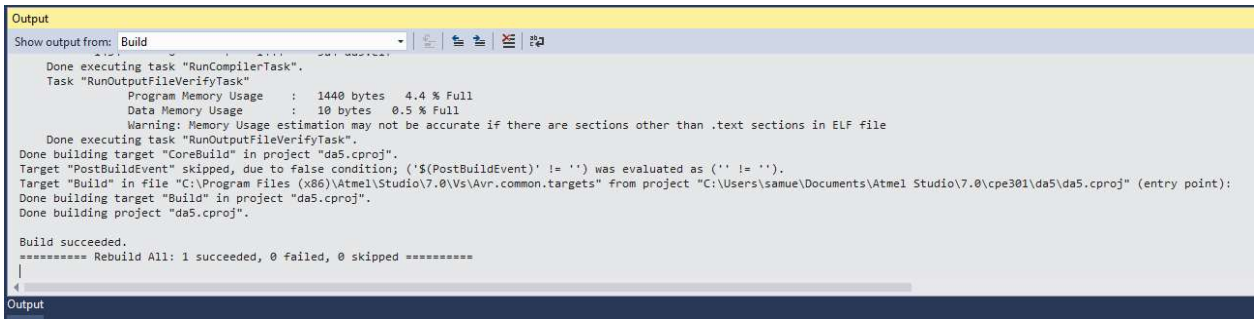


Screen capture of running Processor graph.

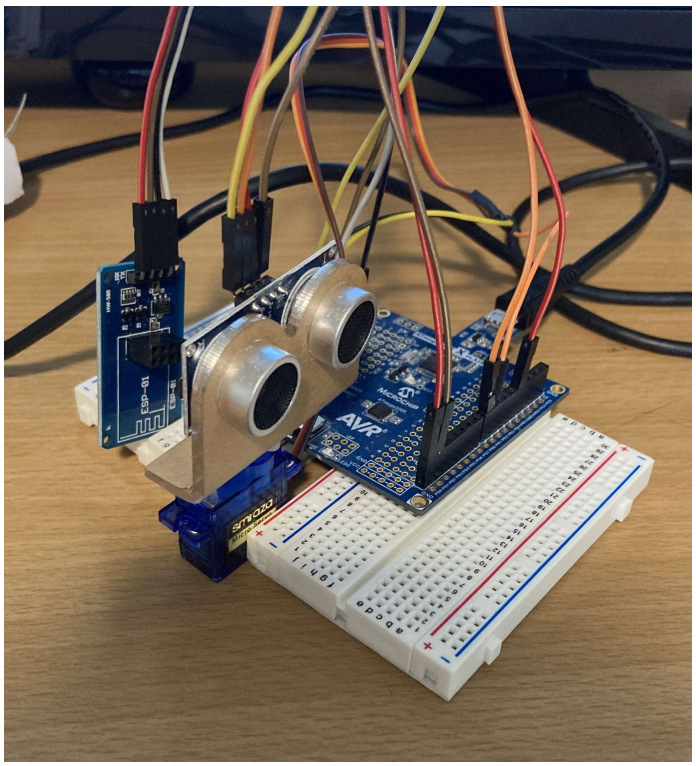
Schematics:


U1
ATmega328-P
AREF
VCC
AVCC
PB0
PB1
PB2
PB3
PB4
PB5
XTAL1/PB6
XTAL2/PB7
PC0
PC1
PC2
PC3
PC4
PC5
RESET/PC6
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
GND
VCC
TRIG
ECHO
GND
U2
HC-SR04
PWM
+
-
M1
Motor_Servo
U3
ESP-01v090
UTXD
CH_PD
RST
VCC
GND
GPIO2
GPIO0
URXD

Captures:



Successful compilation capture



Demo circuit.

Youtube Demos:

https://youtube.com/shorts/MIvUdnOHmuA?feature=share