Aryan Gupta
Partner: Anthony Samagaio

Video Link: https://youtu.be/6Vm6OOLjLMY

Objective: The video will demonstrate a program that rotates a ultrasonic sensor at 0, 90, and 180 degrees using a servo. The ultrasonic sensor takes five measurements and calculates the median of the data points and assumes that to be the distance.

Commentary:

- 0:01: Little box at the closest distance
- 0:02: Medium sized box at medium distance
- 0:04: Large box farthest away from sensor
- 0:09: Measuring at 0° (little box)
- 0:16: Measuring at 90° (medium box)
- 0:22: Measuring at 180° (large box)
- 0:30: Measuring at 90° again (medium box)

Reflection: A lot of hot glue was used in this lab. Learned that cheap servos aren't very accurate when it comes to angles. Even though the uC sent the PWM signal for zero degrees, it didn't rotate to exactly zero degrees, more like to five degrees. No major problems encountered.

Code:

```
//*****************************************************************
// ServoExample - Run an inexpensive Servo Motor
// James Conrad, 2020-06-10
// Aryan Gupta, 2020-06-13
//*****************************************************************

#undef min
#undef max
#include <algorithm>
#include <Servo.h>

#define ARRAY_LEN(X) (sizeof(X) / sizeof(X[0]))

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

constexpr int DEFAULT_SERVO_POS = 90;
constexpr size_t NUM_SAMPLES = 5;

const int trigPin = 32; //This is Port Pin 3.5 on the MSP432 Launchpad
const int echoPin = 33; //This is Port Pin 5.1 on the MSP432 Launchpad
```

```
void setup() {    // put your setup code here, to run once:
   // initialize two digital pins as outputs.
   pinMode(76, OUTPUT);  //RGB LED - P2.1 GREEN_LED
   pinMode(77, OUTPUT);  //RGB LED - P2.2 BLUE_LED
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);
   Serial.begin(9600);

   myservo.attach(38);  // attaches the servo on Port 2.4 to the servo object
   myservo.write(DEFAULT_SERVO_POS);    // Send it to the default position
   delay(500); // delay so the servo has time to get to default position

   Serial.println("Starting HC-SR04 Test...");
}

long get_sonar_val() {
  long samples[NUM_SAMPLES];
  long inches;
  long centimeters;

  for (int i = 0; i < NUM_SAMPLES; ++i) {
    digitalWrite(trigPin, LOW);            // send low to get a clean pulse
    delayMicroseconds(5);                  // let it settle
    digitalWrite(trigPin, HIGH);           // send high to trigger device
    delayMicroseconds(10);                 // let it settle

    samples[i] = pulseIn(echoPin, HIGH);

    inches = samples[i] / 148;
    centimeters = samples[i] / 58;

    Serial.print("Distance = ");
    Serial.print(centimeters);
    Serial.println(" centimeters");

    delay(1000);
  }

  std::sort(samples, samples + NUM_SAMPLES);

  return samples[(NUM_SAMPLES / 2) + 1];
}

void servo_goto(int new_pos) {
  static int pos = DEFAULT_SERVO_POS;

  int direction = (new_pos < pos)? -1 : 1;

  for (; pos != new_pos; pos += direction) {
```

```
    myservo.write(pos);  // tell servo to go to position in variable 'pos'
    delay(15);           // waits 15ms for the servo to reach the position
  }
}

void loop() {    // put your main code here, to run repeatedly:
  int sweep_angles[] = { 90, 0, 90, 180 };
  static int current_angle_idx = 0;

  long pulseLength;
  long inches;
  long centimeters;

  servo_goto(sweep_angles[current_angle_idx]);

  Serial.print("Measuring at ");
  Serial.print(sweep_angles[current_angle_idx]);
  Serial.println(" degrees");

  pulseLength = get_sonar_val();
  inches = pulseLength / 148;
  centimeters = pulseLength / 58;

  Serial.print("Median distance at ");
  Serial.print(sweep_angles[current_angle_idx]);
  Serial.print(" degrees = ");
  Serial.print(centimeters);
  Serial.println(" centimeters");

  current_angle_idx++;
  if (current_angle_idx >= ARRAY_LEN(sweep_angles)) {
    current_angle_idx = 0;
  }
}
```