CS 241 Lab 04
(Sensors and Actuators)

Benjamin Stream
Solomon Himelbloom

February 8, 2021

1 Answers to Questions

1. Benjamin Stream and Solomon Himelbloom

2. Benjamin used some old gift cards, binder clips, and paperclips to jerry-rig a stable system together. Solomon connected two flashcards and two jumper cables into a sandwich-like rectangle placed vertically on the servo control.

   We placed our servo controls 6 inches away from the ultrasonic sensor – this ensured optimal visibility for data collection. We will be examining Benjamin's data in this lab report since he got more consistent results between trials.

3. After multiple trials, we discovered that our servo control horn had to be big enough to be picked up by the ultrasonic sensor. On the other hand, it could not be too large, thus obstructing future readings and the field of view for the various data collection points.

   Our ultrasonic sensor readings became fairly repeatable once we finalized the code, attached the control horn apparatus to the servo, and stabilized the servo to a hard surface, like our desk. Since the servo was not moving between trials, we found that our total distance varied by 10-15 millimeters during each data collection.

4. See attached graph below.

5. On average, it took 72 milliseconds for the servo output to start moving after we sent a command. This value stayed consistent between data collection and has room for optimization by refactoring our Arduino code with less focus on logging outputs to the console and more concise digitalWrite statements.

6. It took Benjamin's servo 22 milliseconds to finish moving after we sent a command. This value was similar to Solomon's data collection trials, meaning that we were able to stay consistent with each other's results.

7. Plotting our data on a histogram leads to a close normal distribution for the lower time intervals, but the data collection becomes more sporadic as time and servo degrees increase. We suspect that this is the case and do not have the means and precise materials to ensure that these values are statistically significant.

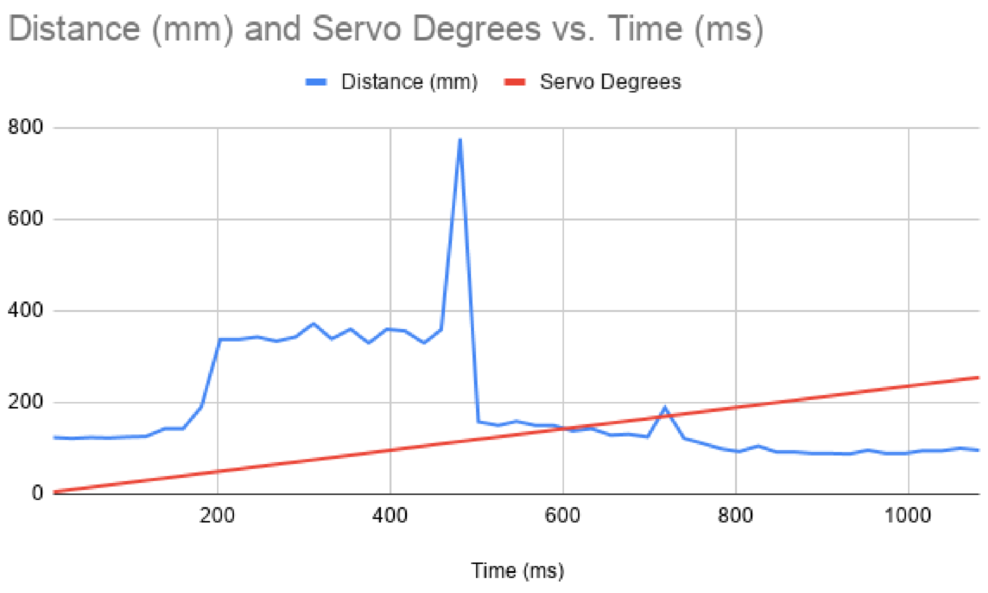
8. N/A

9. Values such as percent error are important to know because they can help us guide future decision-making while debugging and optimizing our desired Arduino output. We can use standard deviation to see the overall spread of data values and standard error to view how much they differ from a repeated average value.

10. See our attached Arduino code below.

2 Appendix

2.1 Data Collection



2.2 Source Code

```
/*
   Benjamin Stream & Solomon Himelbloom

   Credit to:
  Simple test driver program for an ultrasonic distance sensor
  Dr. Orion Lawlor, lawlor@alaska.edu, 2021-01-31 (Public Domain)
*/

/* Pins used:
   gnd - arduino ground
   echo - pin 13
   trigger - pin 11
   vcc - 5v
   servo - pin 2
*/
#include <Servo.h>

Servo serv;
const int pin_echo = 13;
const int pin_trigger = 11;
const int pin_servo = 2;
const int max_distance = 10000; // big invalid distance value

```

```
/* Return distance in millimeters, as measured by ultrasonic sensor.
   If nothing is returned from the sensor, return max_distance.
   This call takes between 8ms and 25ms.
*/

unsigned int read_distance() {
  /* Activate trigger pin for 10us */
  digitalWrite(pin_trigger, HIGH);
  delayMicroseconds(10); /* fire! */
  digitalWrite(pin_trigger, LOW);

  /* Read back echo duration from pulse length on echo pin.
     Echo lengths longer than this limit are tough to achieve in practice.
  */
  long time_us = pulseIn(pin_echo, HIGH, 25000);

  if (time_us == 0) { /* we timed out.  Report an invalid distance */
    return max_distance;
  }


  /* The sensor can't be triggered too often, or it stops returning pulses */

  int time_ms = time_us / 1024;
  const int min_ms = 8; // minimum number of milliseconds per pulse

  if (time_ms < min_ms) delay(min_ms - time_ms);
  /* Convert time to distance, via speed of sound in mm/us, and factor of 2 round trip */
  int dist_mm = (int)(time_us * (0.343 / 2));
  return dist_mm;

}

unsigned long sendServo(int degree) {
  serv.write(degree);
}

void setup()

{
  /* Set pin modes for the sensor pins */
  serv.attach(pin_servo); // servo's white wire is on pin D2
  pinMode(pin_echo, INPUT);
  pinMode(pin_trigger, OUTPUT); digitalWrite(pin_trigger, LOW);
  Serial.begin(9600); // for debug info
  Serial.println("Ultrasonic echo detector");
}

int position = 0; // Initial servo position
int count = 0;

void loop() {
    //sendServo(position);
    //delay(1000);

  while (count <= 50) {

    sendServo(position);
    position+= 5;
    if(position == 360)
    {
      position = 0;
    }

    int distance = read_distance();

    if (count <= 50) {
      Serial.print(millis()); // Time (ms)
      Serial.print("\t");
      Serial.print(distance); // Distance (mm)
      Serial.print("\t");
      Serial.println(position); // Servo command (%)
    }
    delay(10); // max 100Hz
    count++;
  }
  sendServo(position);
    delay(1000);
}
```