GROUP NUMBER: 13

ROLL NUMBER 1: 200260021 NAME 1: JVS Shreya
ROLL NUMBER 2: 200260051 NAME 2: Shreya Shrivastava

Ultrasonic Pong

Two player pong game on laptop, played by controlling the paddles with a physical object or hand (detected by ultrasonic sensor)
[note: the final project as built may be quite different from your initial proposal. *Everything in this report must refer to the final project as presented in the demo.*]

ABSTRACT:

We use ultrasonic sensors to measure the distance of an object which will be used to control the paddle in the pong game.

The sensor data is interfaced by an Arduino and is filtered to eliminate noise using a simple Kalman Filter algorithm.

The filtered data is sent to a USB port of the laptop connected to the Arduino. We then recognize the port as a serial port in Python using a library called pyserial.

We redefine the Readline function so that the rate at which data is read by the Python code is fast enough for the game to run smoothly.

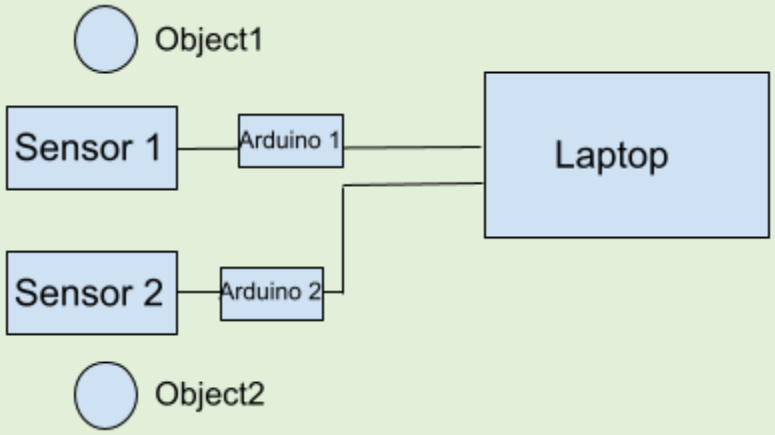
We incorporate the received data into the paddle controls of the game.

< describe your project idea in 5 lines or less >

PROJECT DETAILS:

< Give a detailed description of your project idea >

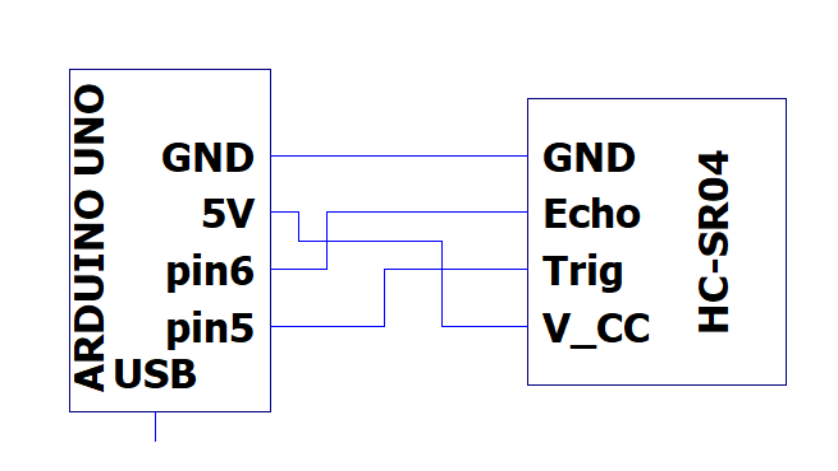
You must provide a block diagram of the major components of the project.
[1]



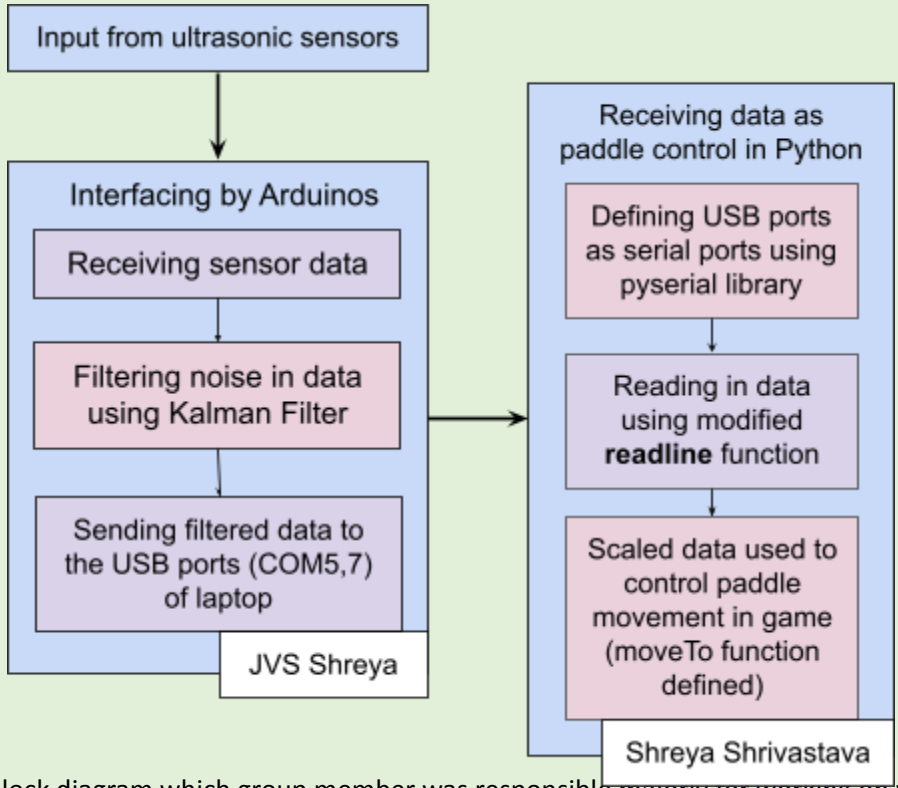
Sensor 1, 2: Ultrasonic sensors [HC-SR04]

Use blocks to specify all sensors/input/output elements used. Give the details of the parts used, especially if they are commercially purchased parts (like ultrasonic sensors, LCD screens etc)

[2]
External bread-boarded circuits should be discrete blocks – give the circuit diagram corresponding to those circuits as sub-figures. Hand-drawn circuit diagrams are *not* acceptable – use a proper software like circuitlab or ltspice etc you have learned in earlier labs.



[3]
The Arduino with the required algorithm should be in a separate block.
The **algorithm flowchart** must be shown here. Program code is to be included as an appendix to the report (see below for format)



Mark on your block diagram which group member was responsible majorly for working on which block. Writing 'both did everything' is not acceptable – surely you must have shared workload among the group members. The TA's have been tracking your progress.

[4]
If your project has significant analog circuits as part of the design, include LTSpice simulation circuit diagrams and simulation result plots of the analog component.

MAIN COMPONENTS NEEDED TO BUILD THE PROJECT:

- Two Ultrasonic Sensors (HC-SR04)
- Two Arduino Unos, USB connectors
- Two breadboards, long wires

*Also took two servo motors for testing direct arduino to other hardware outputs in the last week but did not include in final project.

Give an inventory of all the components you used to complete the project. If you needed to purchase some components other than the ones provided in your kit, please mention them separately.

RESULTS:

Summarize the results of your project.
Provide photos or links to video recording of your working project output. In the project demo and viva we expect a fully working end-result of your project work. But sometimes a last minute part failure may cause problems. In that case, we would like to see that your project worked at *some* time!

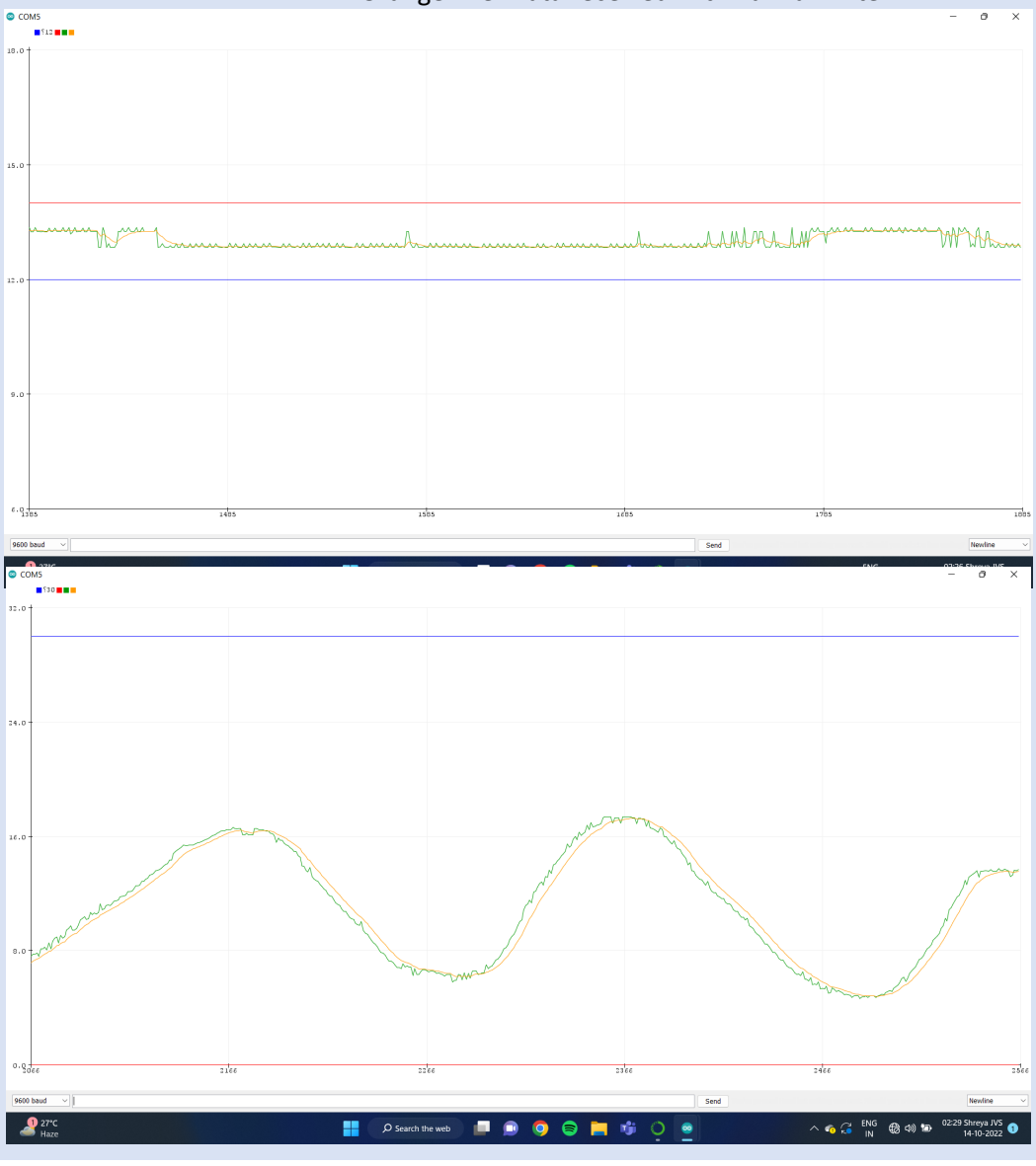
Plus, a photogenic project output gets you a chance to get on the 'Projects Hall-of-Fame' poster board.

PHOTOS

Initially there was a lot of noise in the measured data. We finally improved this by implementing a Kalman Filter algorithm and smoothened out the values as one can see in the below images.

1. Serial Plotter(Distance vs time)-Green line: Data received without Kalman Filter

Orange line: Data received with Kalman FIlter



VIDEOS

Arduino data with kalman filter (to reduce noise) results:
https://drive.google.com/file/d/1PJHI7oLmr85ZZ6siKYq0kJIzwWgbc7yt/view?usp=share_link

Removed time delay (excluding kalman filter):
https://drive.google.com/file/d/1Oh89jLZgpNgIt6gYaj0GzFp6XcbuQ4sw/view?usp=share_link

We were facing a huge time delay issue which was exponentially increasing as time passed on. This was because the default readline function of the pyserial library was very slow. We fixed this by modifying the Readline function which is mentioned in the appendix. The final result can be seen below:

Final set up after fixing time delay:
https://drive.google.com/file/d/1PAi1E8Qn1KetunZoM8J_WDHAas2s3-TK/view?usp=share_link

APPENDIX:

Program code is to be put here in the following fixed width font. Note that the code must be well commented and self-explanatory. The following is a shining example of well-written code:
(the color coding of functions is just a cosmetic add-on for readability)

To view our code, go to this github repository

Arduino Code

```
// To measure distance of object placed in front of ultrasonic sensor

const int trigPin2 = 5;
const int echoPin2 = 6;

float duration2, distance2; // duration as measured by the sensor, distance which is scaled
// according to duration

float x_k_p, P_k_p, K_k; // variables required to run Kalman Filter algorithm
float x_k = 15; // variable storing updated distance value
float P_k = 1;  // variable storing updated error in distance
float Q = 1e-5; // covariance of environmental noise (value taken from paper)
float R = 2.92e-4; // covariance of measurement noise of HC-SR04 sensor

void setup() {

  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  Serial.begin(9600);

}

void loop() {

  // sending ultrasonic wave through speakers by controlling voltage of trigger pin
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);

  // scaling measured duration with speed of sound to get distance in centimeters
  duration2 = pulseIn(echoPin2, HIGH);
  distance2 = (duration2*.0343)/2;

  // Kalman Filter algorithm(for details, refer to the end of report)
  x_k_p = x_k;
  P_k_p = P_k + Q;
  K_k = P_k_p/(P_k_p + R);
  x_k = x_k_p + K_k*(distance2 - x_k_p);
  P_k = (1 - K_k)*P_k_p;
```

```
  // Sendoing data as a string variable to the Serial monitor(or USB port of the laptop)
  String dataToSend2 = String(x_k);
  Serial.println(dataToSend1);
  // delay finetuned to match the delay taken by Python game to run
  delay(40);

}
```

Pyhton code snippets

ReadLine class (faster than the standard readline function of pyserial library)

```
class ReadLine:
    def __init__(self, s):
        self.buf = bytearray()
        self.s = s

    def readline(self):
        i = self.buf.find(b"\n")
        if i >= 0:
            r = self.buf[:i+1]
            self.buf = self.buf[i+1:]
            return r

        while True:
            i = max(1, min(2048, self.s.in_waiting))
            data = self.s.read(i)
            i = data.find(b"\n")

            if i >= 0:
                r = self.buf + data[:i+1]
                self.buf[0:] = data[i+1:]
                return r

            else:
                self.buf.extend(data)
```

moveTo function

```
def moveTo(self, pixels):
        self.rect.y = pixels
       #Check that you are not going too far (off the screen)

        if self.rect.y < 0:
          self.rect.y = 0
        if self.rect.y > 400:
          self.rect.y = 400
```

Defining serial ports

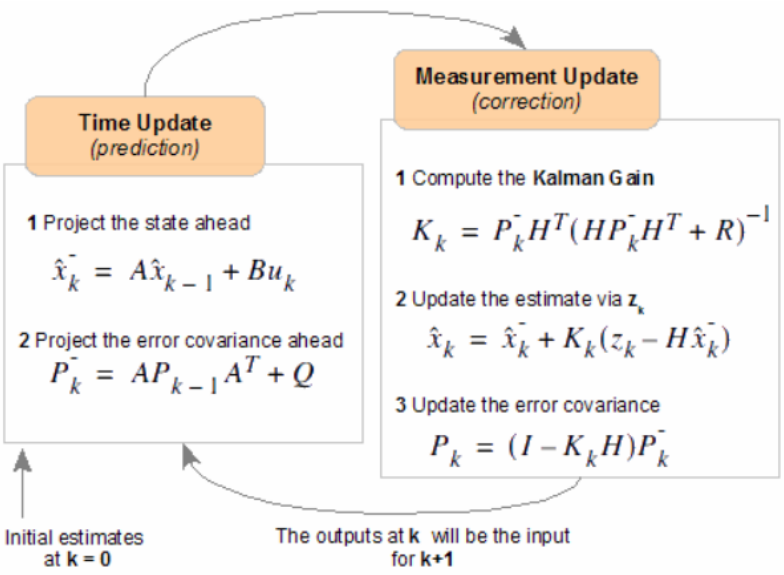
```
ser1 = serial.Serial('COM7',9600)
ser1.close()
ser1.open()

ser2 = serial.Serial('COM5',9600)
ser2.close()
ser2.open()
```

Giving paddle controls

```
some1 = int(20*float(data1.decode()))
some2 = int(20*float(data2.decode()))
```

paddleA.moveTo(some1 - 50.)
paddleB.moveTo(some2 - 50.)

Kalman Filter algorithm



The traditional Kalman Filter algorithm flowchart

$$\hat{x}_k^{(-)} = \hat{x}_{k-1}$$
$$P_k^{(-)} = P_{k-1} + [1e-5]$$
$$K_k = P_k^{(-)}\left(P_k^{(-)} + [2{,}92e-4]\right)^{-1}$$
$$\hat{x}_k = \hat{x}_k^{(-)} + K_k\left(z_k - \hat{x}_k^{(-)}\right)$$
$$P_k = (1 - K_k)P_k^{(-)}$$

Above are the equations obtained for our particular case