1. Signal Processing for One Ultrasonic Sensor

1.1 Problem Statement:

By using the single HC-SR04 ultrasonic sensor, collecting the sample and filtering the data by Implementing the Kalman filter to reduce the noise in the time data and to calibrate the ultrasonic HC-SC04 sensor for a suitable fitting function to map the time to the distance.

1.2 Technical Approach:

a. Sampling

- To sample the time data using two ultrasonic sensors where the sensor pulse travels till the object placed and return's back.

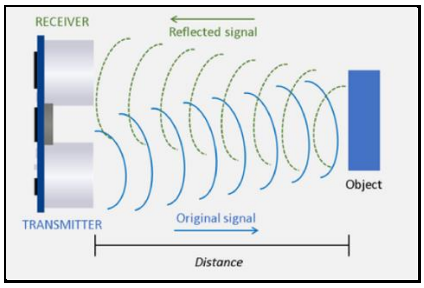

Figure 1

b. Filtering

- To design the Kalman filter and implement Kalman filter to order to reduce the noise in the time data.

<u>Design and implementation of Kalman filter:</u>

<u>Design:</u>

- A, B, H = Constant matrices
- K = discrete time
- Y = system state to be estimated
- v = measurement noise
- U = system input
- Z = system measurement
- W = process noise

$y_k = A*y_{k-1} + B*u_k + w_k$

$z_k = H*y_k + v_k$

In this project the procedure applied for the model:

$y_k = y_{k-1} + w_k$

$z_k = y_k + v_k$

• Prediction step:

- $\hat{y}_{-k} = y_{k-1}$ (predicted state estimation)

- P-k = Pk-1 + Q (predicted error covariance)

• Correction step:

- K = P-k (P-k + R)-1 (Kalman gain)
- ŷk = ŷ-k + K (zk – ŷ-k) (updated state estimate)
- PK= (I – K) P-k (updated error covariance)
- The above is the flow chart for the programming

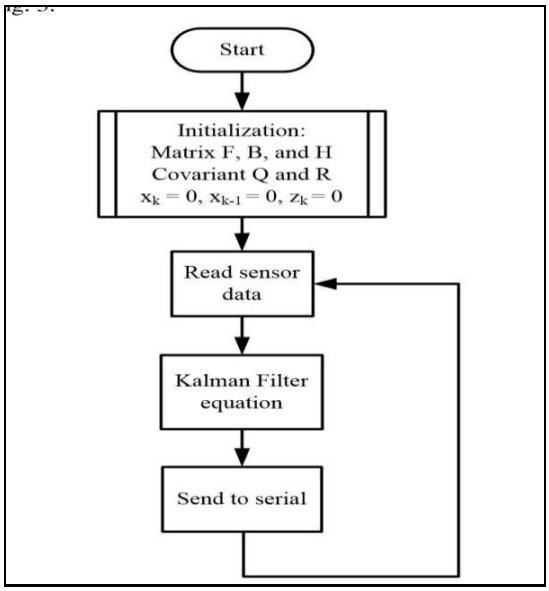

Figure 2

Reference Link: https://www.semanticscholar.org/paper/Kalman-Filter-Algorithm-Design-for-HC-SR04-Sensor-Tahtawi/c29010137f6183d5e2ed43d4c3edc6e3b942fcab

Implementation in Code:

```
kt = p1 / (p1 + r1);  // Calculating Kalman Gain
pdt = (1 - kt) * p1;  // Calculating updated process covariance
yk1 = yk + kt * (zt - yk);  // New Estimation based on Kalman Gain
h1 = yk1; // Storing the updated Estimate in a handle
yk =  h1;  // Using handle to update the filtered estimate in prediction for next iteration.
h2 = pdt;  // Storing the updated Process covarience in handle.
p1 = h2;  // Using handle to update the filtered process covarience for next iteration

distance_filter =  0.174*yk1 - 12.294;  // Calibrated Polynomial function that converts
                                           filterd time data into distance.
```

Figure 3 (Impementation in code)

c. Calibration

- To design an appropriate fitting function to map the time to distance in millimetre's and calculate using polyfit (fitting) function. We calculated with different distance and found different variance.

```
X_A = [10, 20, 30, 46, 60, 70, 87, 100, 115, 130, 150];
Y_A = [11.06711055 8.050226131 8.296457286 9.495075377 12.32552764  9.820075377 11.72037688 10.65316583 12.41987437 11.66341709 12.4210804];
T_A = [591.8757163  1213.491696     1740.00263      2742.678279 3457.808228 4079.589733 5055.744709 5762.00908  6657.164072 7551.429232 8701.354

plot(X_A,Y_A,'o')
hold on
coeff_A = polyfit(X_A,Y_A,1)
f2 = polyval(coeff_A,X_A);
plot(X_A,f2)
title('Sensor A time var vs dist')
hold off
plot(T_A,X_A,'o')
hold on
 coeff_t1 = polyfit(T_A,X_A,1)
ft2 = polyval(coeff_t1,T_A);
plot(T_A,ft2)
title('Sensor A time vs dist')
hold off

X_B = [10, 20, 30, 46, 60, 70, 87, 100, 115, 130, 150];
Y_B = [11.63175879 6.608015075  2.229045226 9.899899497 10.93577889 2.285201005 6.693869347 9.355879397 14.71949749 10.88741206 14.49758794];
T_B = [604.410441   1207.992275 1788.889381 2690.63817  3524.168456 4065.459833 5043.694535 5801.469198 6693.708906 7509.044279 8656.429167];

plot(X_B,Y_B,'o')
hold on
coeff_B = polyfit(X_B,Y_B,1)
f3 = polyval(coeff_B,X_B);
plot(X_B,f3)
title('Sensor B')
hold off
coeff_t2 = polyfit(X_B,T_B,2)
ft3 = polyval(coeff_t2,X_B);
```

Calibration Code

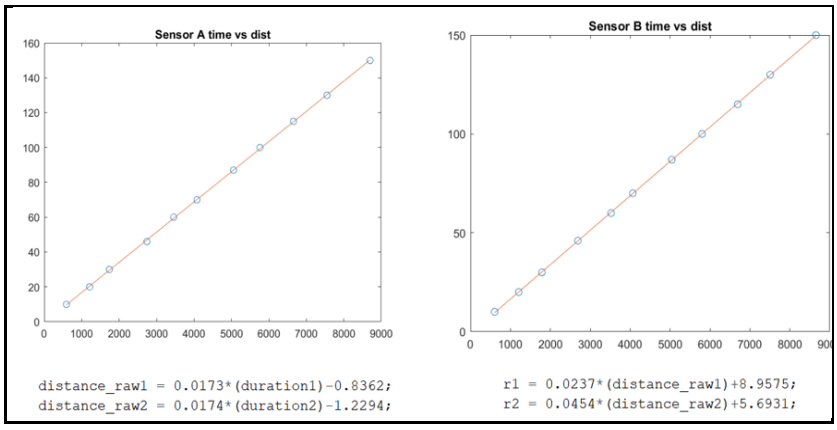

Table 1

d. Test

- After the calibration process, the distance[mm] is printed and also, the variance in real time.
- So, when the filtering is done where variance is small enough flash a LED sound a buzzer, stop printing, and output the time cost.

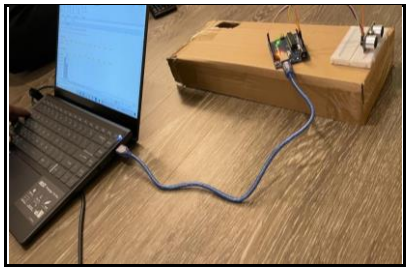

Figure 4

1.3 Hardware and Software Implementation:

Hardware:

- The hardware components used for the project are

a. Arduino UNO Rev3:

- The Arduino UNO is a microcontroller board which consists of 14 digital input/output pins which is used for variety of electronics projects.

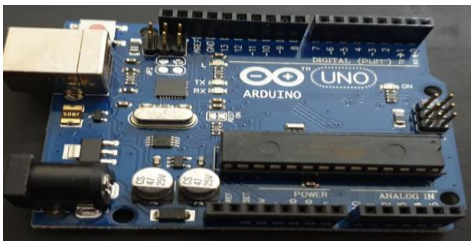

Figure 5

Reference link: https://www.elecrow.com/elecrow-uno-r3-board-atmega328p-atmega16u2-with-usb-cable-for-arduino.html

b. Bread board:

- The bread board is the construction base of electronics where it is used to build and test circuits quickly before finalizing the any circuit design further.

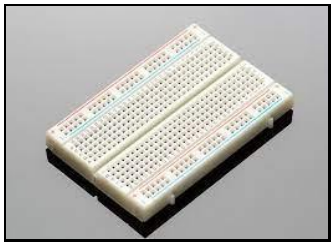

Figure 6

c. HC-SR04 ultrasonic sensor:

- The HC-SR04 ultrasonic sensor is the sensor used for detecting the distance from the object. The sensor uses the non-contact ultrasound sonar to measure the distance to the object.

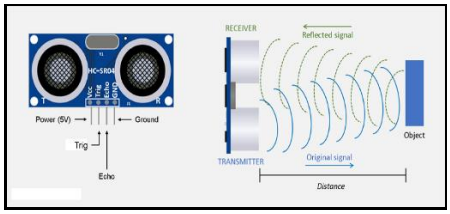

Figure 7

Reference link: https://osoyoo.com/2018/09/18/micro-bit-lesson-using-the-ultrasonic-module/

d. Jumper wires (male to male and male to female wires):

- A jumper wire is an electrical wire which is usually used for connecting the components of a breadboard for performing any kind of test or for prototyping.

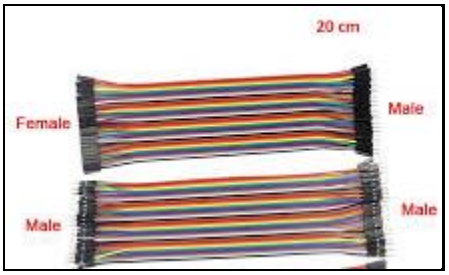

Figure 8

Reference link: https://arduinogetstarted.com/hardware/best-jumper-wires-for-arduino

e. Red led and buzzer

Software Implementation:

- Defining the parameters to execute the data.

```
#define trigPin1 5       // Trigger Pin of Ultrasonic sensor A
#define echoPin1 4       //Echo Pin of Ultrasonic sensor A

float p1, r1, yk,Time1,Time2;  // Defining Global Variables
int d=0;
const int buzz = 9, Led=10;  // For using Digital pin outputs for LED and Buzzer


void setup() {
  Serial.begin (9600);  // To Initiate Serial port data readout
  pinMode(trigPin1, OUTPUT);  // Trigger pin of Sensor1
  pinMode(echoPin1, INPUT);   // Echopin of Sensor 1
  pinMode(buzz,OUTPUT);     // Defining pin for Buzzer
  pinMode(10,OUTPUT);      // Defining Pin for LED

  Serial.println("START");    // Enmarking start of the program
  Time1 = millis();  // Recording Initializing Timestamp
```

Implementation of code

- Then, starting and initializing the timestamp in the data.
- The polynomial fitting function is found by calibrating the time output to distance.
- Calculated the covariance by fitting function.
- The polynomial function is calibrated that converts filtered (using distance filter) time in to distance.
- The condition is given such that after calculating the time cost and distance the buzzer, LED switched ON.

Experimental Results:

- The experimental result obtained by using single HC SC04 Ultrasonic sensor-

```
10:15:25.495 -> START
10:15:25.495 -> Filtered Distance 1324.96Varience5.82
10:15:25.515 -> Filtered Distance 1325.74Varience5.82
10:15:25.515 -> Filtered Distance 1325.68Varience3.90
10:15:25.590 -> Filtered Distance 1325.70Varience2.77
10:15:25.605 -> Filtered Distance 1325.50Varience2.65
10:15:25.685 -> Filtered Distance 1324.23Varience1.72
10:15:25.717 -> Filtered Distance 1324.76Varience1.65
10:15:25.760 -> Filtered Distance 1324.02Varience1.55
10:15:25.795 -> Filtered Distance 1324.35Varience1.45
10:15:25.825 -> Filtered Distance 1324.50Varience1.37
10:15:25.895 -> Filtered Distance 1324.72Varience1.23
10:15:25.910 -> Filtered Distance 1326.90Varience0.73
10:15:25.014 -> Filtered Distance 1326.98Varience0.82
10:15:25.051 -> Filtered Distance 1326.80Varience0.78
10:15:25.105 -> Filtered Distance 1326.88Varience0.83
10:15:25.155 -> Filtered Distance 1326.73Varience0.78
10:15:25.205 -> Filtered Distance 1326.61Varience0.57
10:15:25.250 -> Filtered Distance 1326.44Varience0.69
10:15:25.345 -> Filtered Distance 1326.35Varience0.71
10:15:26.305 -> Filtered Distance 1326.27Varience0.53
10:15:26.305 -> Filtered Distance 1326.20Varience0.49
10:15:26.375 -> Filtered Distance 1326.13Varience0.54
10:15:26.395 -> Filtered Distance 1323.03Varience0.43
10:15:26.465 -> Filtered Distance 1323.97Varience0.42
10:15:26.510 -> Filtered Distance 1323.92Varience0.56
10:15:26.545 -> Filtered Distance 1323.83Varience0.47
10:15:26.615 -> Filtered Distance 1322.80Varience0.39
10:15:26.290 -> 6573.00
```

Figure 9 (Time VS Distance)

2. Sensor Fusion of Multiple Ultrasonic Sensors:

2.1 Problem Statement:

- By using the multi-HC-SR04 ultrasonic sensor, collecting the sample and filtering the data by Implementing the Kalman filter to reduce the noise in the time data and to calibrate the ultrasonic HC-SC04 sensor for a suitable fitting function to map the time to the distance.

2.2 Technical Approach:

a. Sampling

- To sample the time data using two ultrasonic sensors where the sensor pulse travels till the object placed and return's back.

b. Filtering

- To design the Kalman filter and implement Kalman filter to order to reduce the noise in the time data.

<u>Design and implementation of Kalman filter:</u>

<u>Design:</u>

- A, B, H = Constant matrices
- K = discrete time
- Y = system state to be estimated
- v = measurement noise
- U = system input
- Z = system measurement
- W = process noise

$yk = A*yk-1 + B*uk + wk$

$zk = H*yk + vk$

In this project the procedure applied for the model:

yk = yk-1 + wk

zk = yk + vk

• Prediction step:

- ŷ-k = yk-1 (predicted state estimation)
- P-k = Pk-1 + Q (predicted error covariance)

• Correction step:

- K = P-k (P-k + R)-1 (Kalman gain)
- ŷk = ŷ-k + K (zk – ŷ-k) (updated state estimate)
- PK= (I – K) P-k (updated error covariance)
- The above is the flow chart for the programming

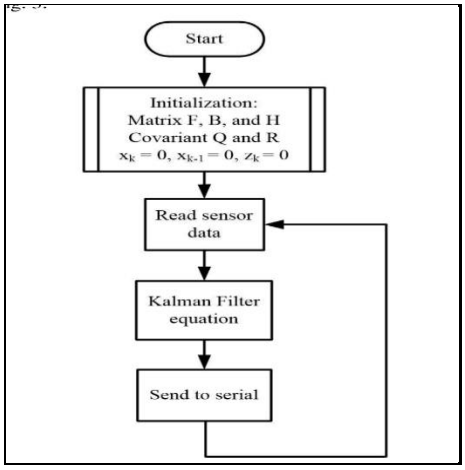

Figure 10

Reference Link: https://www.semanticscholar.org/paper/Kalman-Filter-Algorithm-Design-for-HC-SR04-Sensor-Tahtawi/c29010137f6183d5e2ed43d4c3edc6e3b942fcab

Implementation in Code:

```
kt = p1 / (p1 + r1);  // Calculating Kalman Gain
pdt = (1 - kt) * p1;  // Calculating updated process covariance
yk1 = yk + kt * (zt - yk);  // New Estimation based on Kalman Gain
h1 = yk1; // Storing the updated Estimate in a handle
yk =  h1;  // Using handle to update the filtered estimate in prediction for next iteration.
h2 = pdt;  // Storing the updated Process covarience in handle.
p1 = h2;  // Using handle to update the filtered process covarience for next iteration

distance_filter =  0.174*yk1 - 12.294;  // Calibrated Polynomial function that converts
                                           filterd time data into distance.
```

Figure 11 (Impementation in code)

e. Calibration

- To design an appropriate fitting function to map the time to distance in millimetre's and calculate using polyfit (fitting) function. We calculated with different distance and found different variance.

```
X_A = [10, 20, 30, 46, 60, 70, 87, 100, 115, 130, 150];
Y_A = [11.06711055 8.050226131 8.296457286 9.495075377 12.32552764  9.820075377 11.72037688 10.65316583 12.41987437 11.66341709 12.4210804];
T_A = [591.8757163  1213.491696     1740.00263      2742.678279 3457.808228 4079.589733 5055.744709 5762.00908  6657.164072 7551.429232 8701.354

plot(X_A,Y_A,'o')
hold on
coeff_A = polyfit(X_A,Y_A,1)
f2 = polyval(coeff_A,X_A);
plot(X_A,f2)
title('Sensor A time var vs dist')
hold off
plot(T_A,X_A,'o')
hold on
 coeff_t1 = polyfit(T_A,X_A,1)
ft2 = polyval(coeff_t1,T_A);
plot(T_A,ft2)
title('Sensor A time vs dist')
hold off

X_B = [10, 20, 30, 46, 60, 70, 87, 100, 115, 130, 150];
Y_B = [11.63175879 6.608015075  2.229045226 9.899899497 10.93577889 2.285201005 6.693869347 9.355879397 14.71949749 10.88741206 14.49758794];
T_B = [604.410441   1207.992275 1788.889381 2690.63817  3524.168456 4065.459833 5043.694535 5801.469198 6693.708906 7509.044279 8656.429167];

plot(X_B,Y_B,'o')
hold on
coeff_B = polyfit(X_B,Y_B,1)
f3 = polyval(coeff_B,X_B);
plot(X_B,f3)
title('Sensor B')
hold off
coeff_t2 = polyfit(X_B,T_B,2)
ft3 = polyval(coeff_t2,X_B);
```

Calibration Code

X1	Var. S1	t1	T1.Var			t2	var.T2		var.T3	t4	var.T4
26.93	0.003266	1570	12.57666	26.93	1.93	949	7.912663	3220	159.6259	1912	4.41907
26.98		1571		26.98	1.98	950		3219		1913	
26.86		1564		26.86	1.86	949		3218		1912	
26.84		1569		26.84	1.84	955		3245		1912	
26.83		1571		26.83	1.83	950		3219		1914	
26.95		1564		26.95	1.95	951		3220		1913	
27.03		1564		27.03	2.03	951		3221		1913	
26.86		1571		26.86	1.86	956		3219		1913	
26.84		1565		26.84	1.84	949		3220		1913	
26.96		1564		26.96	1.96	950		3220		1920	
26.95		1575		26.95	1.95	951		3245		1913	
26.83		1571		26.83	1.83	951		3220		1912	
26.86		1564		26.86	1.86	951		3219		1912	
26.95		1572		26.95	1.95	949		3219		1919	
26.95		1571		26.95	1.95	950		3245		1912	
26.84		1564	AVG	26.84	1.84	949		3219		1913	
26.84		1564		26.84	1.84	949		3245		1912	
26.83		1571		26.83	1.83	950		3245		1912	
26.95		1564		26.95	1.95	949		3218		1912	
26.96		1564		26.96	1.96	949		3239		1912	
26.84		1570		26.84	1.84	950		3212		1914	
26.84		1574		26.84	1.84	951		3212		1912	
26.96		1564		26.96	1.96	951		3213		1912	
26.95		1572		26.95	1.95	951		3245		1912	
26.84		1571		26.84	1.84	950		3221		1919	

Table 1

f. Test

- After the calibration process, the distance[mm] is printed and also, the variance in real time.
- So, when the filtering is done where variance is small enough flash a LED sound a buzzer, stop printing, and output the time cost.

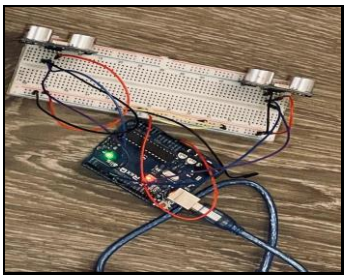

Figure (Test module)

2.3 Hardware and Software Implementation:

Hardware:

- The hardware components used for the project are

a. Arduino UNO Rev3:
b. Bread board
c. HC SR04 Ultrasonic Sensor
d. Jumper wires
e. 3D-printed case
f. External battery
g. Red led and buzzer

Software Implementation:

- Defining the parameters to execute the data.

```
#define trigPin1 5       // Trigger Pin of Ultrasonic sensor A
#define echoPin1 4       //Echo Pin of Ultrasonic sensor A

float p1, r1, yk,Time1,Time2;  // Defining Global Variables
int d=0;
const int buzz = 9, Led=10;  // For using Digital pin outputs for LED and Buzzer


void setup() {
  Serial.begin (9600);  // To Initiate Serial port data readout
  pinMode(trigPin1, OUTPUT);  // Trigger pin of Sensor1
  pinMode(echoPin1, INPUT);   // Echopin of Sensor 1
  pinMode(buzz,OUTPUT);     // Defining pin for Buzzer
  pinMode(10,OUTPUT);      // Defining Pin for LED

  Serial.println("START");    // Enmarking start of the program
  Time1 = millis();  // Recording Initializing Timestamp
```

Implementation of code

- Then, starting and initializing the timestamp in the data.
- The polynomial fitting function is found by calibrating the time output to distance.
- Calculated the covariance by fitting function.
- The polynomial function is calibrated that converts filtered (using distance filter) time in to distance.
- The condition is given such that after calculating the time cost and distance the buzzer, LED switched ON.

2.4 Experimental Results:

- The experimental result obtained by using single HC SC04 Ultrasonic sensor-

```
10:30:05.495 -> START
10:30:05.495 -> Filtered Distance 1101.15Varience 1.37
10:30:05.517 -> Filtered Distance 1101.57Varience 1.64
10:30:05.545 -> Filtered Distance 1115.31Varience 1.76
10:30:05.573 -> Filtered Distance 1112.25varience 1.12
10:30:06.573 -> Filtered Distance 1109.87Varience 1.08
10:30:06.595 -> Filtered Distance 1101.98Varience 1.01
10:30:06.616 -> Filtered Distance 1101.05Varience 0.89
10:30:06.695 -> Filtered Distance 1006.99Varience 0.79
10:30:06.695 -> Filtered Distance 1006.89Varience 0.68
10:30:06.683 -> Filtered Distance 1006.78Varience 0.63
10:30:06.941 -> Filtered Distance 1006.69Varience 0.59
10:30:07.953 -> Filtered Distance 1006.15Varience 0.43
10:30:07.932 -> Filtered Distance 1007.96Varience 0.40
10:30:07.101 -> Filtered Distance 1006.11Varience 0.37
10:30:07.112 -> Filtered Distance 1007.22Varience 0.35
10:30:07.125 -> Filtered Distance 1008.30Varience 0.33
10:30:07.145 -> Filtered Distance 1006.38Varience 0.31
10:30:07.179 -> Filtered Distance 1006.48Varience 0.29
10:30:07.201 -> Filtered Distance 1008.54Varience 0.28
10:30:07.225 -> Filtered Distance 1008.62Varience 0.26
10:30:07.257 -> Filtered Distance 1008.59Varience 0.25
10:30:07.295 -> Filtered Distance 1008.64Varience 0.24
10:30:07.245 -> Filtered Distance 1008.62Varience 0.23
10:31:07.234 -> Filtered Distance 1008.66Varience 0.22
10:31:07.539 -> Filtered Distance 1008.62Varience 0.21
10:31:07.521 -> Filtered Distance 1008.62Varience 0.20
10:31:07.615 -> Filtered Distance 1008.59Varience 0.20
11:49:17.025 -> 4279.00
```

Figure 12 (Time VS Distance)

3. Conclusions and Discussions:

3.1 Conclusions (a summary the results for both the approaches)

- They are different approach to analyse and to perform the task. The main aim of the project was to identify the different distances accurately placed away from the object in less time.
- Finally, the test was performed be placing away from the object and the Kalman filter applied is working and the results showed faster in less time.

3.2 Discussions (a comparison of different approaches, and potential future work to further improve each approach)

- While performing the process the application of Kalman filter has been learnt.
- During performing different test's, the following are been observed.
    - There was a temperature difference while performing the test in different environments.
    - While performing the tests it has been observed that, there was a constant noise and false grounding in circuitry.
    - We performed different tests to cut off the noise disturbance errors and get the distances in less time without any errors.