Project 1 Report: Sensing Signal Processing

1. Signal Processing for One Ultrasonic Sensor

1.1 Problem Statement

To estimate the distance of an obstacle/object in 20-to-150 cm of range using one ultrasonic sensor and apply Kalman filter to filter the data.

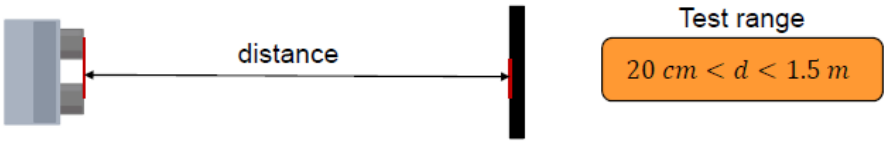

Fig1: Depiction of the problem statement

1.2 Technical Approach

Selecting the Sensor

The ultrasonic sensor selection for this test was based on the data measurements of six sensors in the test range of 20-to-150 cm (10 cm apart).

The variance and average of around 500 measurements for every test distance (for every sensor) was calculated to find its deviation from the actual distance.

The final sensor was chosen based on the least variance and the accuracy of the measured data.

Sampling

To sample the data quickly and accurately, after numerous iterations it was found that a delay of 8ms between activating and deactivating the trigPin of the ultrasonic sensor was optimal to measure any distance within 20-to-150 cm of range.

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(8);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
```

Fig2: Activating and deactivating the ultrasonic sensor

Filtering

To filter the noise from the data Kalman filter was used.

```
//****Prediction****//
Y = Yi;                         // Predicted state estimate
P = Pi;                         // Predicted error covariance

//****Correction****//
K = P / (P + R);                // Kamman gain
Yi = Y + K * (z - Y);           // Updated state estimate
Pi = (1 - K) * P;               // Updated error covariance
```

Fig3: Implementing Kalman filter

Calibration

Calibration was performed both for the measurements and the variance. To get the calibration equation 'linear fit' function in Excel and/or 'polyfit' function in MATLAB was used.

The final calibration equation for the measurements was also incorporated constants to convert the duration into distance. The calibration equation of both measurement and variance was used inside the Kalman filter loop such that the calibrated values will get updated with every iteration.

The constants of the calibration function were calculated by mapping the actual distance to the measured duration.

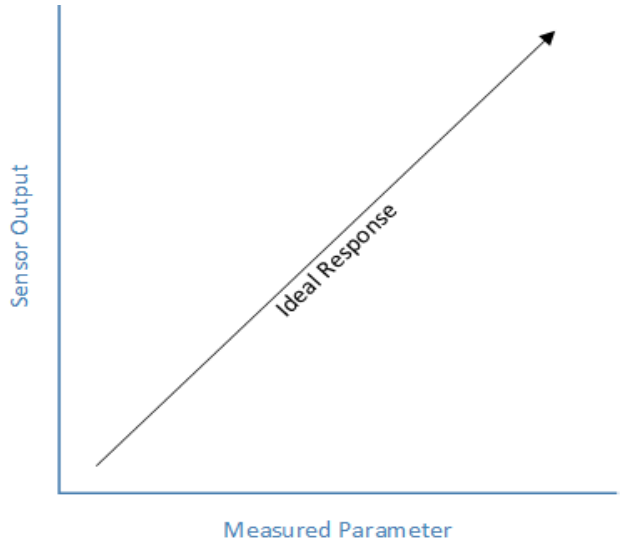

Fig4: Graph representing the basis of calibration using mapping

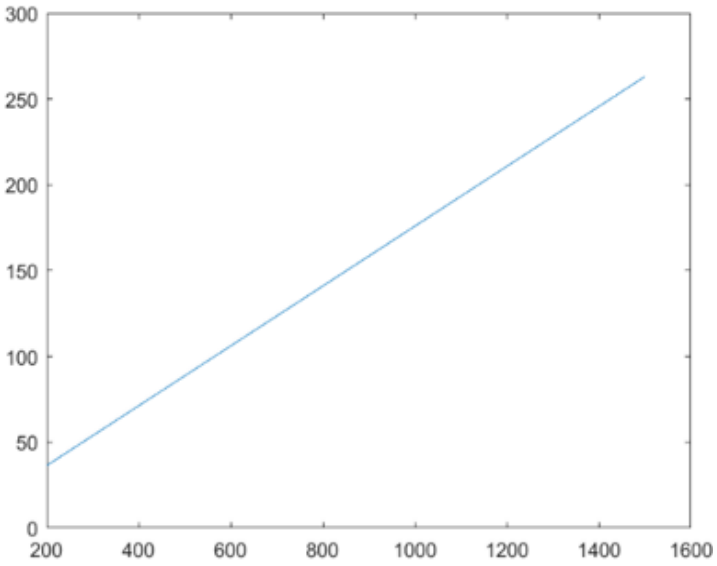

*Fig5: Calibration function for measurements assessed in MATLAB [**x-axis**: measured duration (in µs), **y-axis**: distance (in cm)]*

Error Rejection Function

To avoid erroneous data recorded from the sensor. An error rejection function was implemented in the code which takes value that falls in ±2% tolerance of the first measured data.

```
if (duration<(0.98*duration1st)||duration>(1.02*duration1st)){  // Error Rejection
    duration = duration1st;
}
```

Fig6: Error rejection function for measured duration from the sensor

The graph below depicts that the error rejection will ignore values which are greater than *1.02*duration1st* and less than *0.98*duration1st* and in this case the error rejection function will process the first reading from the sensor ignoring the faulty reading.

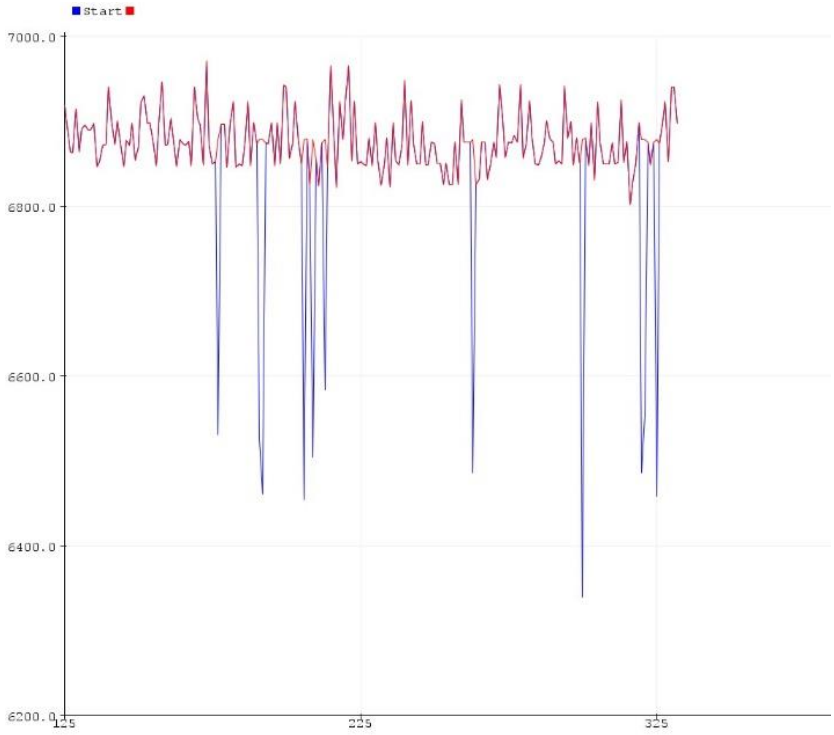

Fig7: Error rejection function to eliminate faulty readings from the sensor
[**x-axis**: measured duration (in ms), **y-axis**: measured duration (in ms)

Test

The measurement value from the ultrasonic sensor, final error covariance and the filtered distance value was printed in the end.

The buzzer gave a beep when the variance is small enough. For the stop criterion, the loop was stopped when the error covariance lies in the specified limit for every distance. In other words, the loop was stopped when the variance was small enough.

1.3 Hardware and Software Implementation

Hardware Used

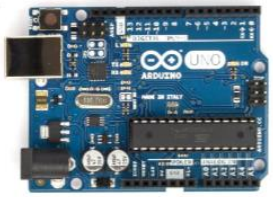

Fig8: Arduino UNO R3 controller

Fig9: Ultrasonic sensor

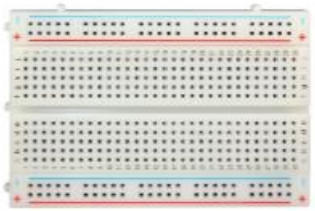

Fig10: Breadboard

Fig11: Wires for connection

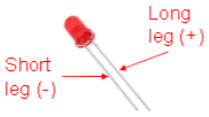

Fig12: Buzzer

Fig13: Resistor (1kohm)

Hardware	Quantity
Arduino UNO R3 controller	1
Ultrasonic sensor	1
Breadboard	1
Wires for Connection	1 (pack)
Buzzer	1
Resistor	1 (1kohm)

Software Used

-Arduino IDE 1.8.16 was used to implement the code.
-Functions in Excel and MATLAB – ‘linear fit’ and ‘polyfit’ were used to calibrate the data.

1.4 Experimental Results

After the test, it was found that the sensor was able to detect distances between 20-to-150 cm with under ± 3 mm of error.

The graph below shows the test results for measuring 40 cm and 80 cm of distance. The convergence time for 40 cm distance was about 230 milli seconds. While the convergence time for 80 cm distance was about 244 milli seconds.

Also, the ‘Blue’ line in the graph represents the unfiltered raw measurement from the sensor. While, the ‘Red’ line represents the filtered data after passing through the Kalman filter.

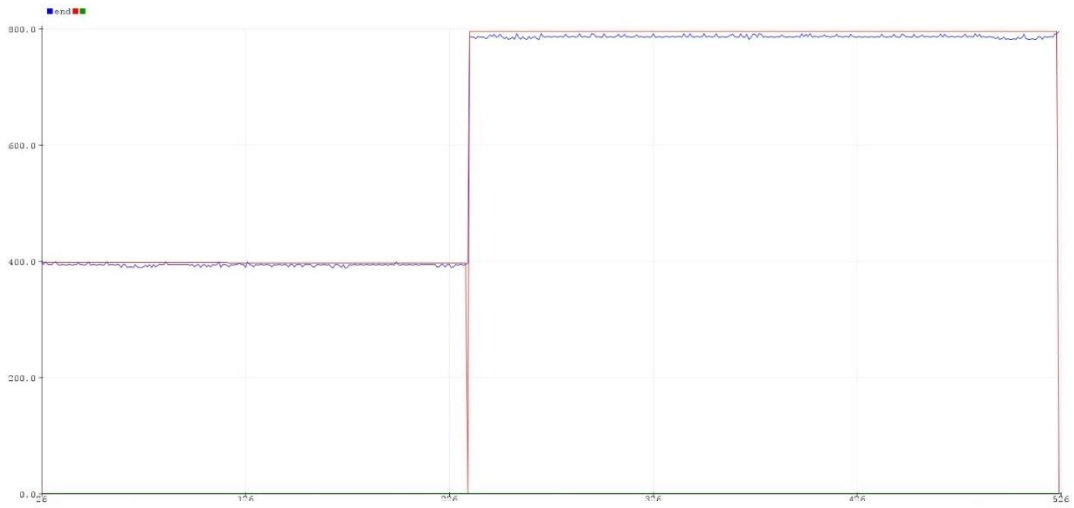

Fig14: Final results showing measure distance 40 and 80 cm.
*[**x-axis**: convergence time (in ms), **y-axis**: measured distance (in mm)]*

---

2. Sensor Fusion of Multiple Ultrasonic Sensors

2.1 Problem Statement

To estimate the distance of an obstacle/object in 20-to-150 cm of range by fusing the output from two ultrasonic sensors and apply Kalman filter to filter the data.

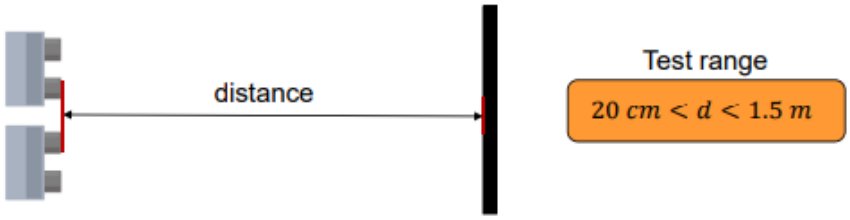

Fig15: Depiction of the problem statement

2.2 Technical Approach

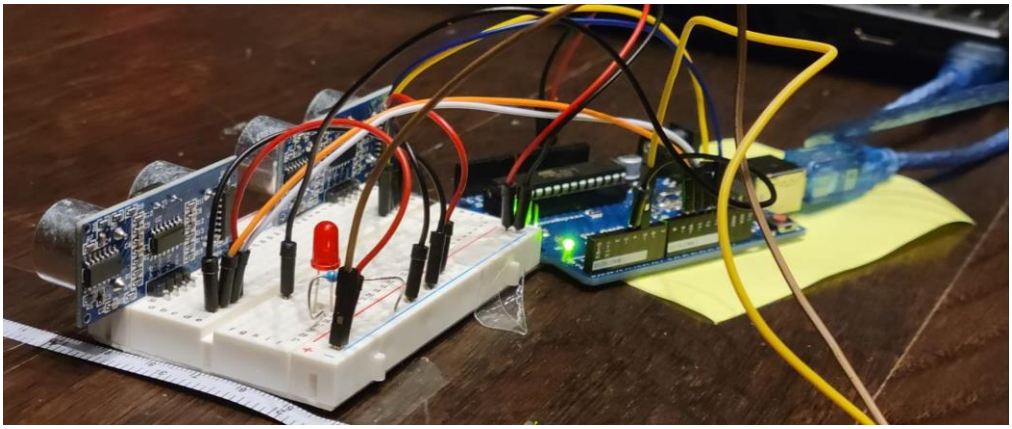

Fig16: Setup for sensor fusion

Sampling

To sample the time data as quick as possible, the accuracy of all the six sensors was tested to select the best sensors having least error and variance.

To sample the data quickly and accurately, after numerous iterations it was found that a delay of 8ms between activating and deactivating the trigPin of the ultrasonic sensor was optimal to measure any distance within 20-to-150 cm of range.

```
digitalWrite(trigPin1, HIGH);
delayMicroseconds(8);
digitalWrite(trigPin1, LOW);
duration1 = pulseIn(echoPin1, HIGH);
```

Fig17: Activating and deactivating the 1st ultrasonic sensor

```
digitalWrite(trigPin2, HIGH);
delayMicroseconds(8);
digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH);
```

Fig18: Activating and deactivating the 2nd ultrasonic sensor

Sensor Fusion and Kalman Filter

After sampling, the two different outputs from the ultrasonic sensors need to be fused to get more accurate values. This fusion was performed by applying the Kalman filter. First, the output from one sensor was corrected in the Kalman filter loop. Next, the output from the previous loop was taken as the input to the second Kalman filter loop, where the second sensor value was taken as a measurement value.

In this way, both sensors were fused to achieve more accurate output.

Calibration

Calibration was performed both for the measurements and the variance.

To calibrate the measurement values; the measured data was mapped with actual test distances using ‘polyfit’ function in MATLAB.

To calibrate variance; variances for different test distances was calculated and mapped with the corresponding test distances using ‘polyfit’ and ‘polyval’ function in MATLAB.

Calibration equations for measurement from the sensors:

```
z1 = 0.00000155951305722164*(pow(distance1,2)) -0.00046645789307297 8*(pow(distance1,1)) +1.01137694207455*(distance1) -1.3270719729626845;
```

```
z2 = 0.00000155951305722164*(pow(distance2,2)) -0.00046645789307297 8*(pow(distance2,1)) +1.01137694207455*(distance2) -1.3270719729626845;
```

Calibration equations for variance:

```
R1 = distance1*0.00110903209401274 -0.0127805363643312;
```

```
R2 = distance2*0.00110903209401274 -0.0127805363643312;
```

Test

The measurement value from both ultrasonic sensors, final error covariance and the filtered distance value were printed.

The buzzer gave a beep when the variance is small enough. For the stop criterion, the loop was stopped when the error covariance lies in the specified limit for every distance. In other words, the loop was stopped when the variance was small enough.

2.3 Hardware and Software Implementation

Fig19: Arduino UNO R3 controller

Fig20: Ultrasonic sensor

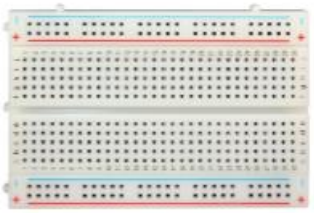

Fig21: Breadboard

Fig22: Wires for connection

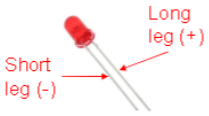

Fig23: Buzzer

Fig24: Resistor (1kohm)

Hardware	Quantity
Arduino UNO R3 controller	1
Ultrasonic sensor	2
Breadboard	1
Wires for Connection	1 (pack)
Buzzer	1
Resistor	1 (1kohm)

2.4 Experimental Results

After the test, it was found that the sensor fusion setup was able to detect distances between 20-to-150 cm with under ± 3 mm of error.

The graph below shows the test results for measuring 20, 40, 90, 120, 150 cm of distances.

The ‘Blue’ line in the graph represents the unfiltered raw measurement from the sensor1. The ‘Red’ line in the graph represents the unfiltered raw measurement from the sensor2. The ‘Green’ line represents the final filtered data after passing through the Kalman filter.

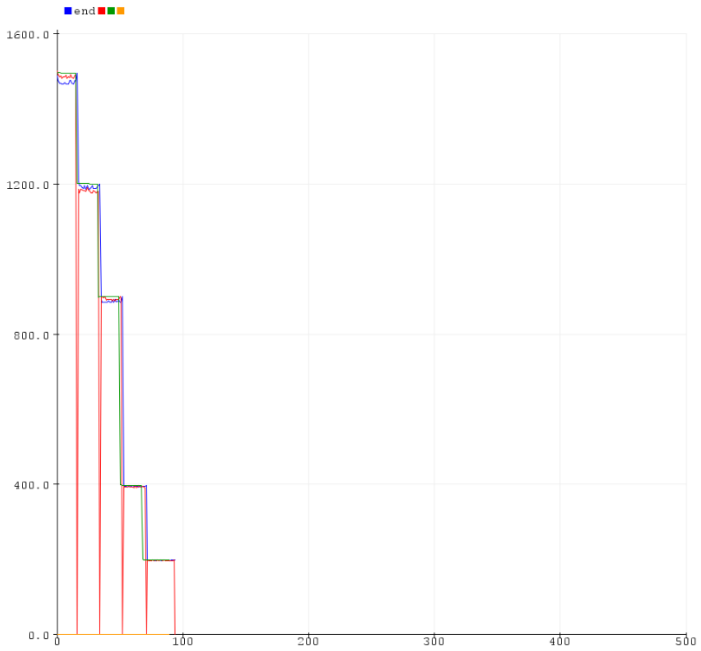

Fig14: Graph showing the measure distances 20, 40, 90, 120, 150 cm
*[**x-axis**: data points, **y-axis**: distance (in mm)]*

3. Conclusions and Discussions

3.1 Conclusions (a summary the results of different approaches)

The main aim of both tasks was the implementation of the Kalman filter to eradicate noises from the sensor readings.

In Task 1.1, it was found that the convergence time was more, while in Task 1.2, with the sensor fusion, it was found that the convergence time was less. Also, the final readings from Task 1.2 were much more accurate than the accuracy of data from Task 1.1.

Furthermore, the error covariance in the case of sensor fusion was much less than that in the single sensor setup.

Overall, this project bolstered my understanding of signal processing and sensor fusion, and it was an excellent hands-on experience of the theoretical knowledge I gained in the class.

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

In both the tasks, the Kalman filter was used to filter noises. A much better approach for Task 1.1 would be to use a time filter first on measured duration to eliminate any unwanted significant errors that could shift the final readings. While in Task 1.2, more than two sensors can be fused to achieve much higher accuracy and less convergence time.

Furthermore, much better and accurate error rejection functions can be included in the code to avoid noisy signals from the sensors. This would also lead to improved convergence time.

Notice: The data in an individual report should be recorded individually. Students in one group can use the same hardware and software but cannot use the same data in the report.