



İSTANBUL  
GELİŞİM  
ÜNİVERSİTESİ

# **Ai-Driven Signal Filtering For Enhanced Kalman Filter Performance in High-Noise Environment**

Abbas Ogal

Faculty Of Engineering And Architecture  
Department of Aeronautical Engineering  
Gelism University  
Istanbul, Turkey  
abbas.ogal@ogr.gelisim.edu.tr

Asst. Prof. PERİ GÜNEŞ

Faculty Of Engineering And Architecture  
Department of Aeronautical Engineering  
Gelism University  
Istanbul, Turkey  
pgunes@gelisim.edu.tr

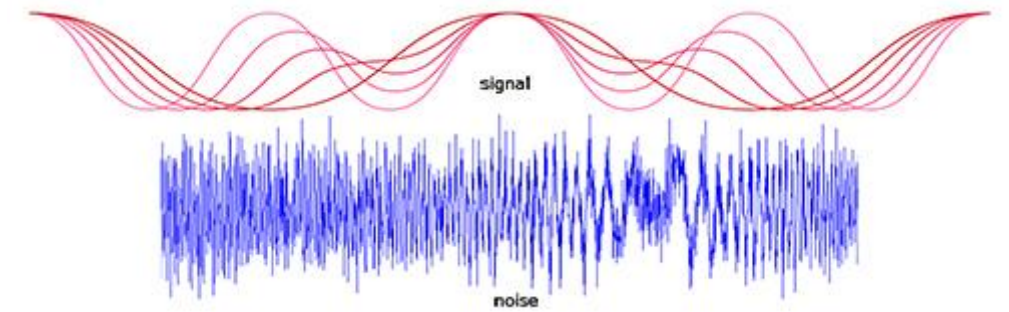
## Overview

- ▶ Kalman filter is **widely used in signal estimation**.
- ▶ **The Extended Kalman Filter** (EKF) face challenges in dynamic noisy environments.
- ▶ This research proposes using **artificial intelligent(AI)** to improve EKF performance.
- ▶ Experimental comparison with IMU sensor data.



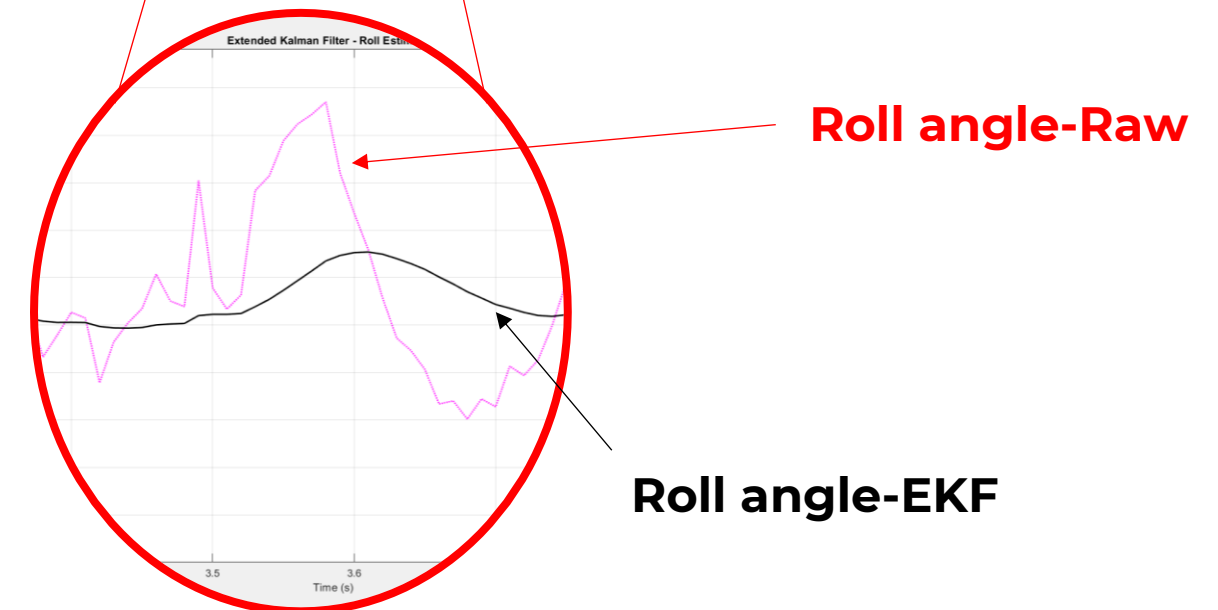
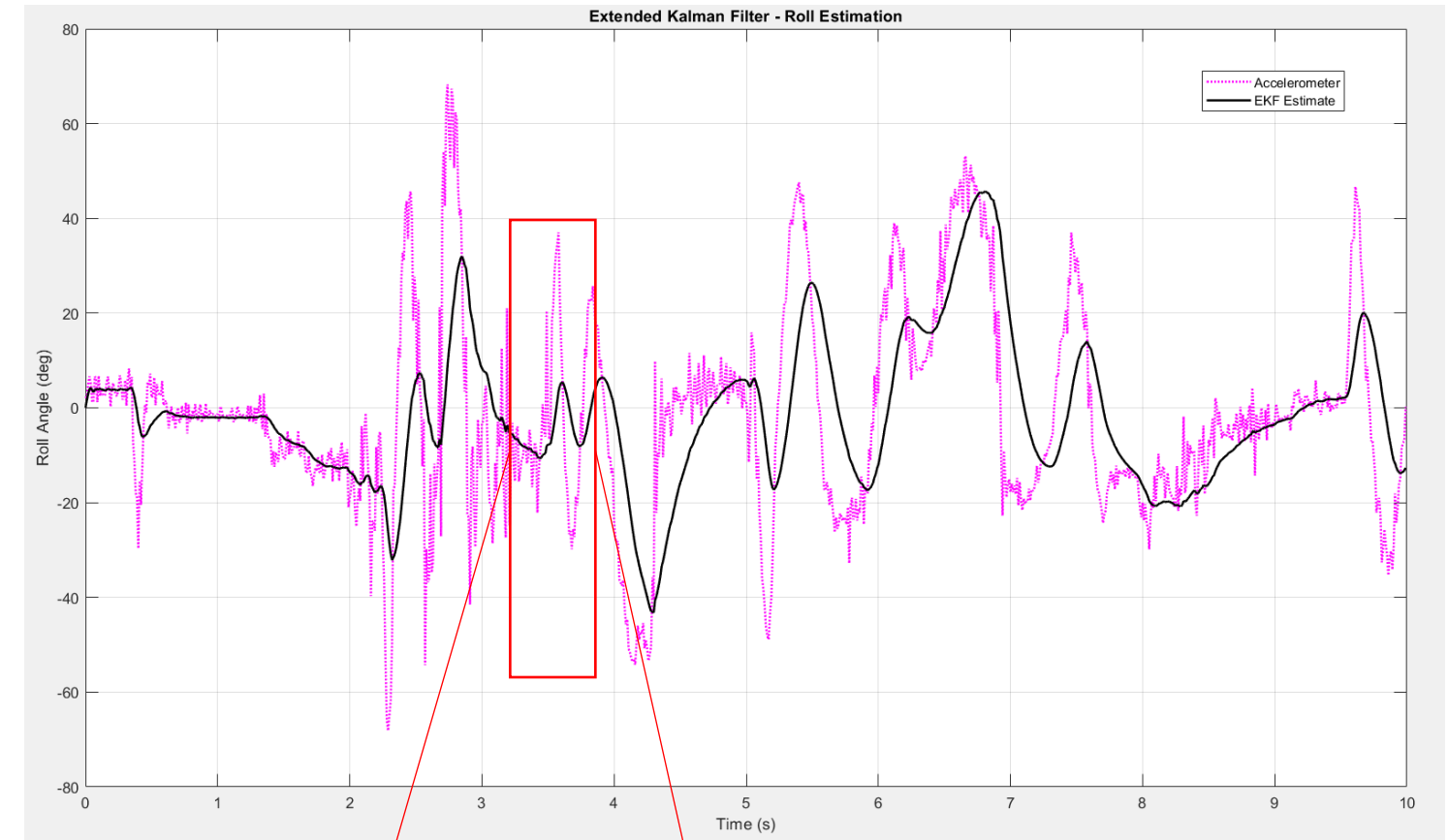
# Introduction

- ▶ Signal **noise** is a key problem in the Research.
- ▶ **Extended Kalman Filter** is popular but struggles under noise.
- ▶ **Artificial Intelligent** can help enhance EKF performance by ***Parameter tuning*** Method (Q, R).



## Research Problem

- ▶ Extended KF has limitations with non-linear data.
- ▶ Noise distorts data → high error rate.
- ▶ AI can dynamically predict/correct signal errors.



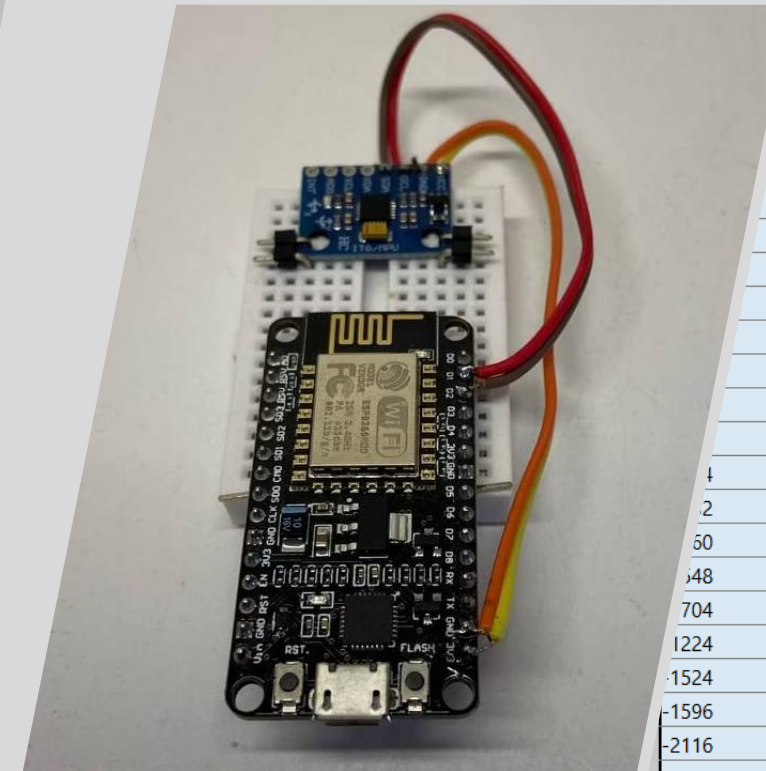


## Objectives

- ▶ Improve EKF by integrating AI-Models.
- ▶ Predict noise **Parameter**(Q,R) before filtering.
- ▶ Reduce Meas Square Error (MSE).
- ▶ Enhance filter stability.

## Implementation Details

- ▶ Data Collection.
- ▶ Data training by using Different typer of AI.
- ▶ Comparing Data
- ▶ Environment System



Number	Number	Number	Number	Number
504	13832	-551	125	-59
568	13768	336	-612	679
1284	14764	219	-1223	693
1916	16424	-670	-673	193
1472	16512	-809	458	-415
328	14872	-467	1023	-215
280	12992	-1841	253	-79
1744	15004	828	-1834	1156
1832	16804	844	0	-578
300	14236	-273	142	346
1820	15316	-378	-463	-65
652	14928	779	-452	-15
1124	16004	-1433	687	-19
692	14036	-167	-133	49
648	14496	-236	-384	45
748	15184	512	-1058	9
1560	16236	-1069	565	
1652	14260	276	-459	
1144	16772	800	-453	
72	15248	843	116	
1232	16188	-390	584	

```
main.c++ X
esp8266 > src > main.c++
1
2
3 #include <Wire.h>
4 #include <MPU6050.h>
5
6 MPU6050 imu;
7
8 void setup() {
9     Serial.begin(9600);
10    // SDA = D2
11    // SCL = D1
12    Wire.begin(D2, D1);
13    imu.initialize();
14
15
16 void loop() {
17     int16_t ax, ay, az, gx, gy, gz;
18
19     imu.getAcceleration(&ax, &ay, &az);
20     imu.getRotation(&gx, &gy, &gz);
21
22     Serial.print(ax); Serial.print(",");
23     Serial.print(ay); Serial.print(",");
24     Serial.print(az); Serial.print(",");
25     Serial.print(gx); Serial.print(",");
26     Serial.print(gy); Serial.print(",");
27     Serial.println(gz);
28
29     delay(100);
30
31 }
```



# Data Collection Process

## Connect ESP32 With Computer

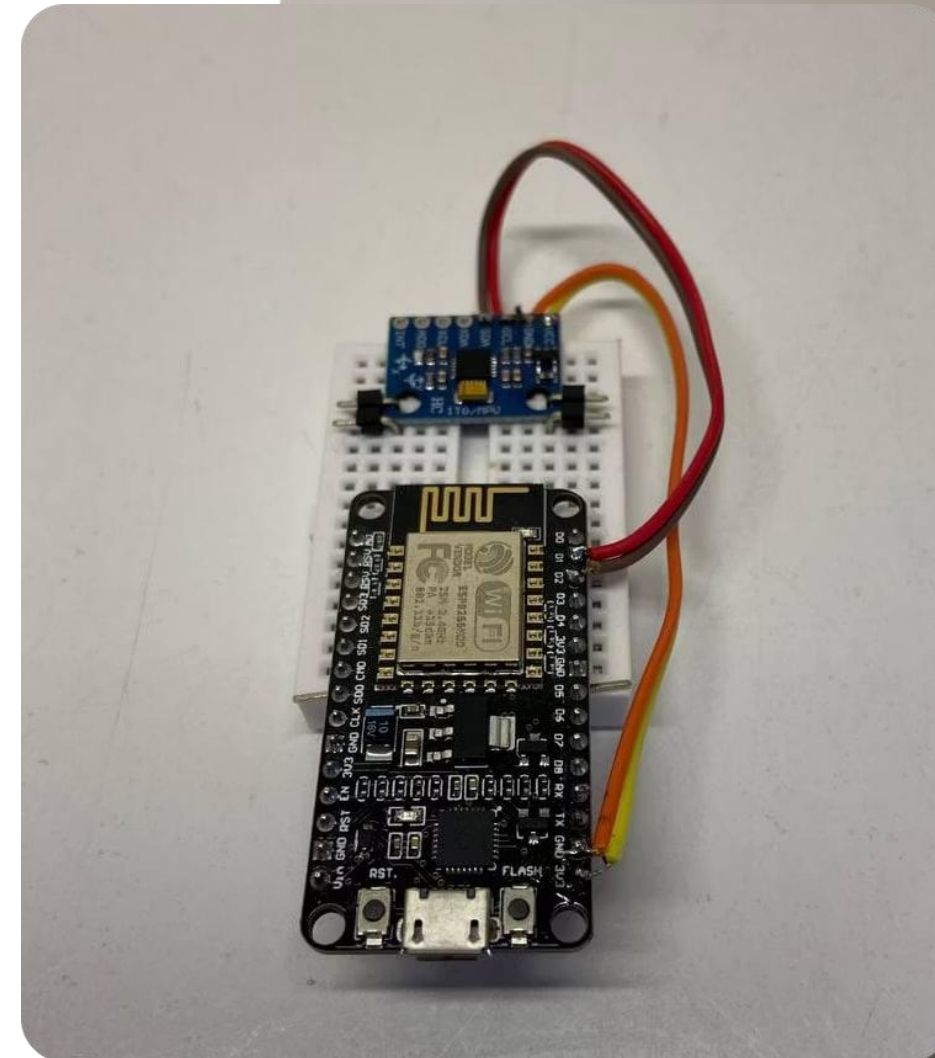
Use USB to connection **ESP32** with a computer.

## Connect ESP32 with IMU

Interface the ESP32 microcontroller with the IMU sensor for real-time data capture by using I2C protocol.

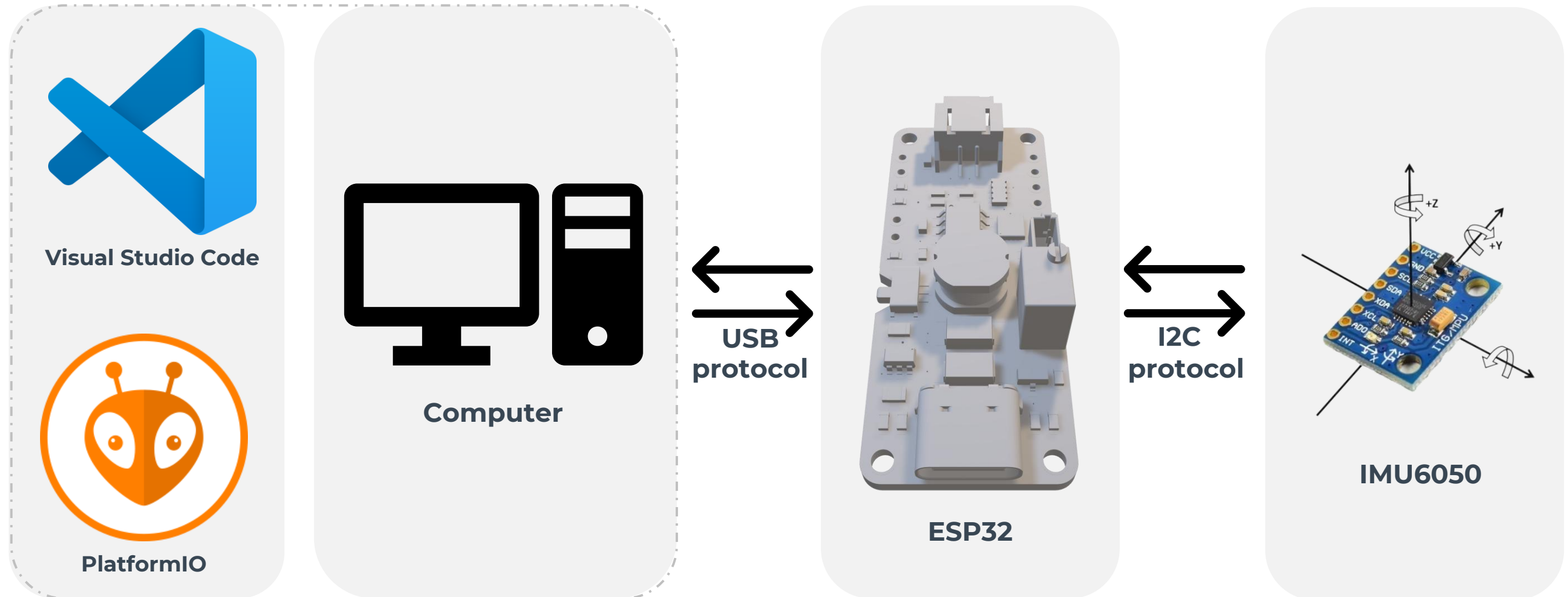
## Record Sensor Data

Capture Data from IMU conditions for analysis.



# Data Collection Process Connection system

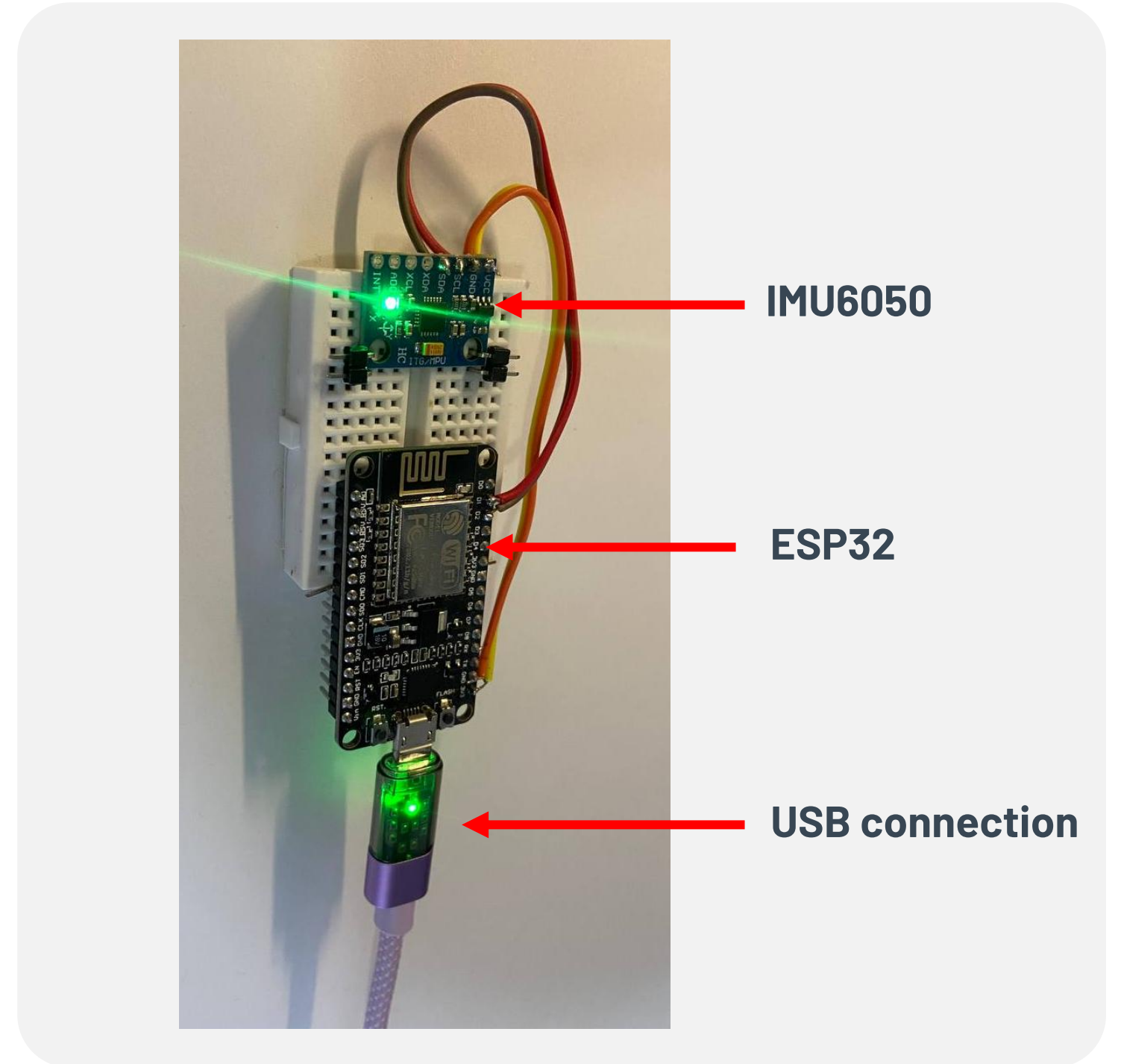
use **Visual Studio Code** and **PlatformIO** Extinction for streamlined data capture this setup enables real-time IMU sensor data recording for analysis.



# Data Collection Process

## Hardware Setup

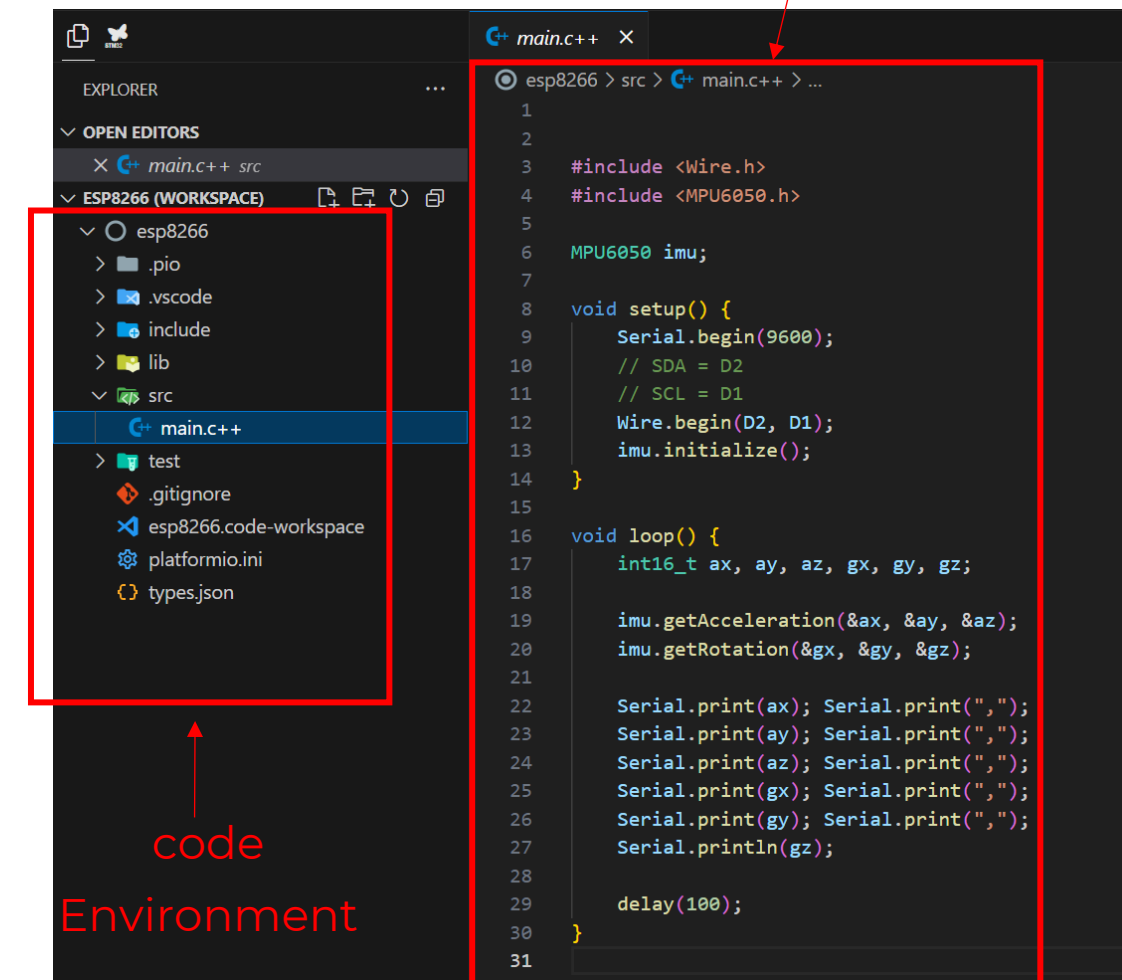
- ▶ The **IMU6050 sensor** is connected to the ESP32 board using I2C. The ESP32 is powered and programmed through a USB cable.



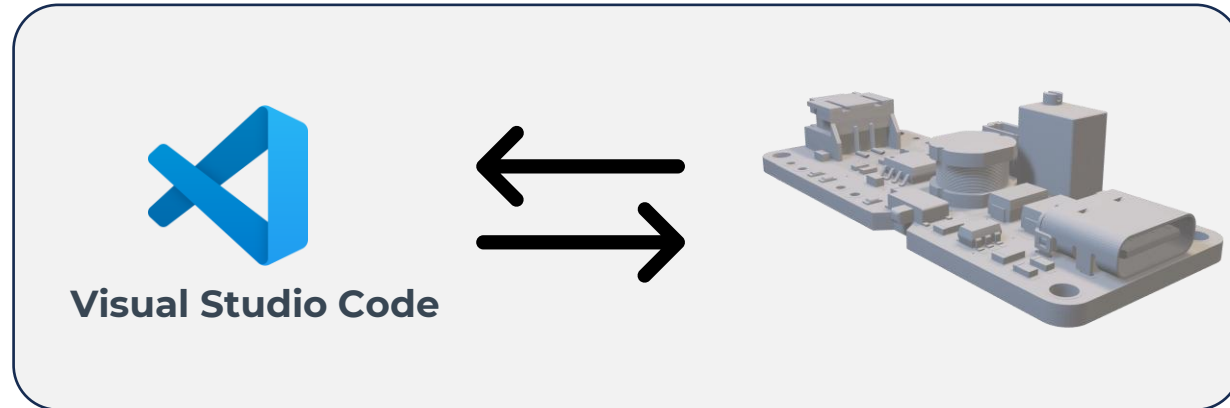


# Data Collection Process Environment PlatformIO with VSCode.

- ▶ **Code Environment:** This shows the folder structure of the **project**, including essential directories like (src, lib, and *configuration files*) such as **platformio.ini**. The main code file **main.cpp** is located in the src folder.
- ▶ **Code Window:** This displays the actual C++/C code used to collect data from the **MPU6050 sensor (IMU)**. The code initializes the sensor and collects accelerometer and gyroscope data.



# Data Collection Process



## Connect ESP32 with IMU

- ▶ Using **Mode Function** inside terminal in Project **Workspace**
- ▶ To Checking Connection Between **system** (Computer) and **subsystem** (ESP32)

```
main.c++ x
esp8266 > src > main.c++ > loop()
6 MPU6050 imu;
7
8 void setup() {
9     Serial.begin(9600);
10    // SDA = D2
11    // SCL = D1
12    Wire.begin(D2, D1);
13    imu.initialize();
14 }
15
16 void loop() {
17     int16_t ax, ay, az, gx, gy, gz;
18
19     imu.getAcceleration(&ax, &ay, &az);
20     imu.getRotation(&gx, &gy, &gz);
21
22     Serial.print(ax); Serial.print(",");
23     Serial.print(ay); Serial.print(",");
24     Serial.print(az); Serial.print(",");
25     Serial.print(gx); Serial.print(",");
26     Serial.print(gy); Serial.print(",");
27     Serial.println(gz);
28
29     delay(100);
30 }
31
```

MEMORY XRTOS TERMINAL SERIAL MONITOR OUTPUT PROBLEMS PORTS SPELL CHECKER DEBUG CONSOLE

PS C:\Users\ASUS\Documents\PlatformIO\Projects\esp8266> mode

Status for device COM3:

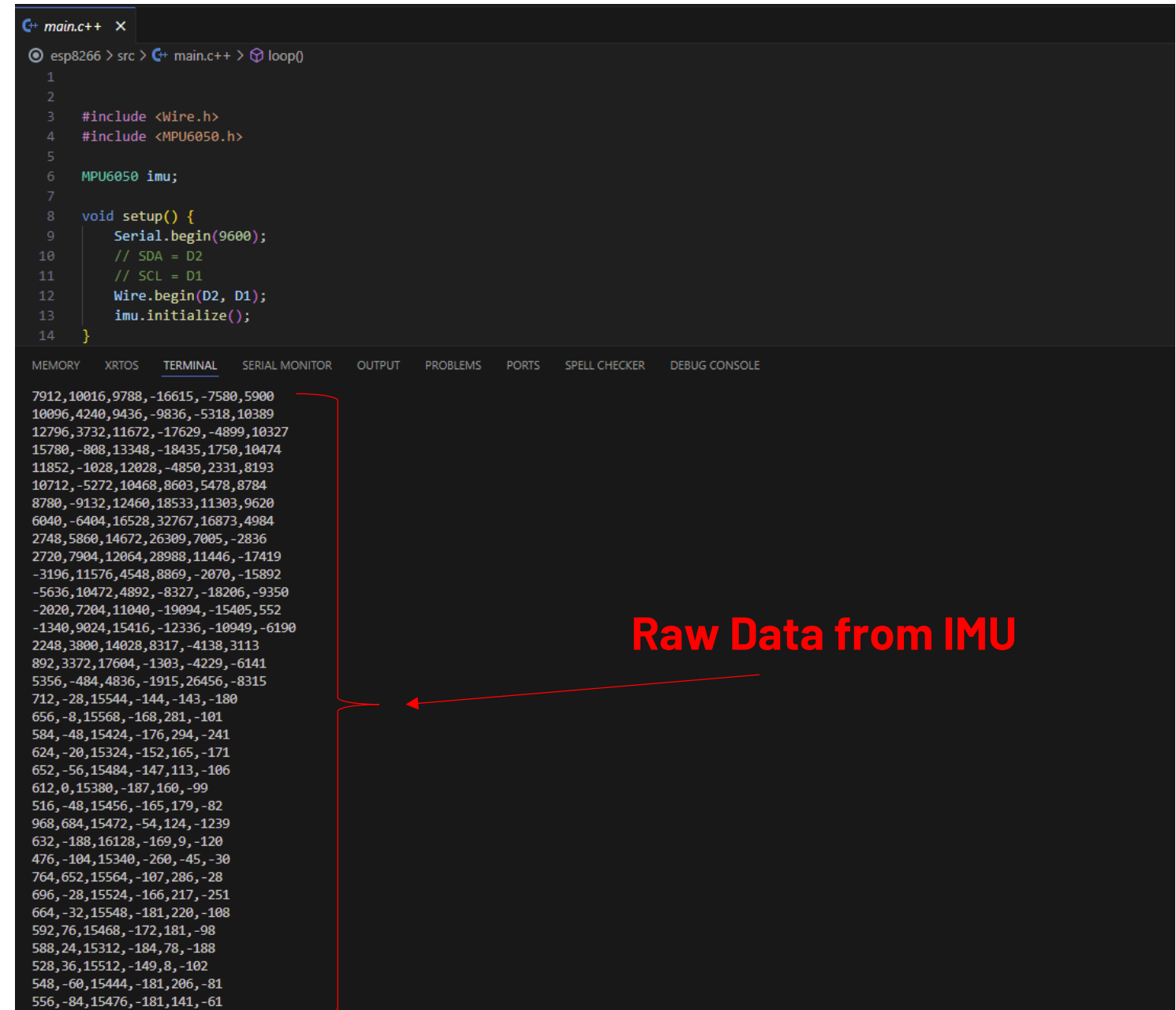
Baud:	115200
Parity:	None
Data Bits:	8
Stop Bits:	1
Timeout:	OFF
XON/XOFF:	OFF
CTS handshaking:	OFF
DSR handshaking:	OFF
DSR sensitivity:	OFF
DTR circuit:	OFF
RTS circuit:	OFF

Mode Function

Baud Rate  
Data Bits size

# Data Collection

- **Raw Data** Output The ESP32 receives raw data from the IMU6050 and sends it via serial communication. The data includes acceleration and gyroscope values displayed in the **Serial Monitor**.



The image shows a screenshot of an IDE with a C++ file named `main.cpp` open. The code is for an ESP32 microcontroller and includes the `Wire` and `MPU6050` libraries. It initializes an `MPU6050` object named `imu` and sets up a serial connection at 9600 baud rate. The `setup` function configures the SDA and SCL pins and initializes the `imu` object. The `loop` function is empty, indicating that the data output is handled by the `imu` object's default behavior.

Below the code editor, the `TERMINAL` tab is active, displaying a stream of raw data from the IMU. The data is presented as a series of comma-separated integers, representing acceleration and gyroscope values. A red arrow points from the text "Raw Data from IMU" to the terminal output.

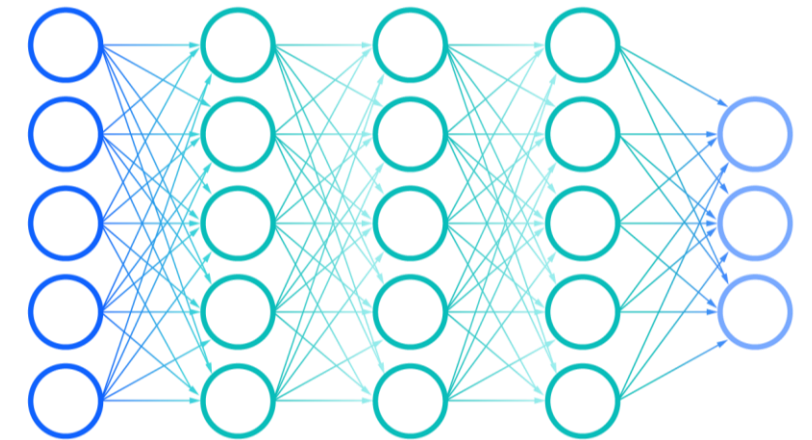
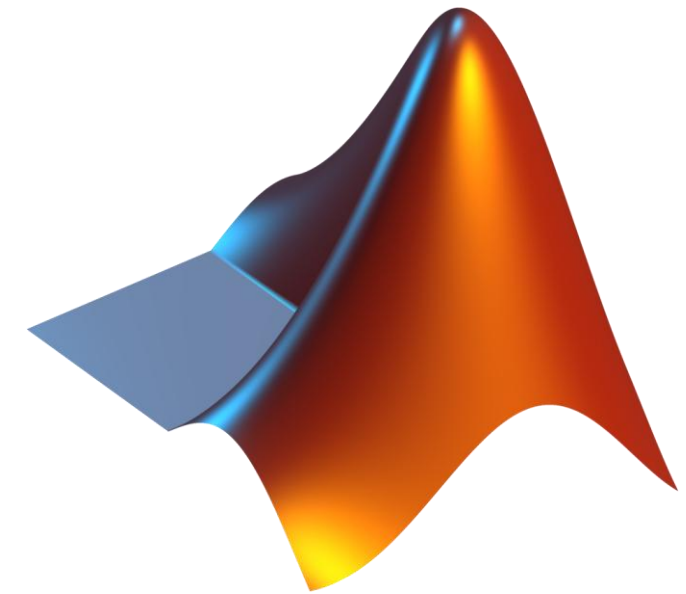
```
main.cpp X
esp8266 > src > main.cpp > loop()
1
2
3 #include <Wire.h>
4 #include <MPU6050.h>
5
6 MPU6050 imu;
7
8 void setup() {
9     Serial.begin(9600);
10     // SDA = D2
11     // SCL = D1
12     Wire.begin(D2, D1);
13     imu.initialize();
14 }

MEMORY  XRTOS  TERMINAL  SERIAL MONITOR  OUTPUT  PROBLEMS  PORTS  SPELL CHECKER  DEBUG CONSOLE

7912,10016,9788,-16615,-7580,5900
10096,4240,9436,-9836,-5318,10389
12796,3732,11672,-17629,-4899,10327
15780,-808,13348,-18435,1750,10474
11852,-1028,12028,-4850,2331,8193
10712,-5272,10468,8603,5478,8784
8780,-9132,12460,18533,11303,9620
6040,-6404,16528,32767,16873,4984
2748,5860,14672,26309,7005,-2836
2720,7904,12064,28988,11446,-17419
-3196,11576,4548,8869,-2070,-15892
-5636,10472,4892,-8327,-18206,-9350
-2020,7204,11040,-19094,-15405,552
-1340,9024,15416,-12336,-10949,-6190
2248,3800,14028,8317,-4138,3113
892,3372,17604,-1303,-4229,-6141
5356,-484,4836,-1915,26456,-8315
712,-28,15544,-144,-143,-180
656,-8,15568,-168,281,-101
584,-48,15424,-176,294,-241
624,-20,15324,-152,165,-171
652,-56,15484,-147,113,-106
612,0,15380,-187,160,-99
516,-48,15456,-165,179,-82
968,684,15472,-54,124,-1239
632,-188,16128,-169,9,-120
476,-104,15340,-260,-45,-30
764,652,15564,-107,286,-28
696,-28,15524,-166,217,-251
664,-32,15548,-181,220,-108
592,76,15468,-172,181,-98
588,24,15312,-184,78,-188
528,36,15512,-149,8,-102
548,-60,15444,-181,206,-81
556,-84,15476,-181,141,-61
```

# Processing Data

- ▶ Data Processing By using **MATLAB Software**.
- ▶ Using Raw Data to training **AI models** to prediction  $(Q,R)^*$  value in Extend Kalman Filter **Formulation**.
- ▶ Compare Data from EKF with Data from EKF-AI models.





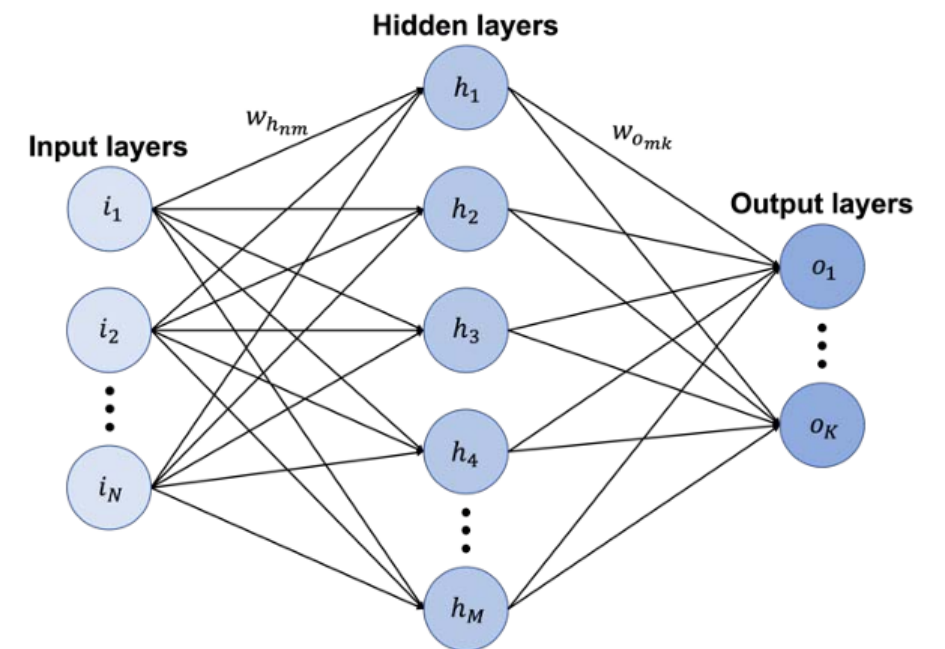
# Artificial Intelligent (AI)

Several **AI models** were tested to improve the ***Parameter tuning*** filtering process and enhance **state estimation** accuracy. These models include:

1. **ANN (Artificial Neural Network):** Basic model used to **learn patterns** in noisy signals.
2. **LSTM (Long Short-Term Memory):** Handles time-series data and remembers long-term dependencies.
3. **GRU (Gated Recurrent Unit):** Similar to LSTM but **simpler** and **faster**, used for real-time estimation.
4. **BiLSTM (Bidirectional LSTM):** Reads data **forward** and **backward** for better context understanding.
5. **Fuzzy Logic Model:** Uses fuzzy rules to handle **uncertainty** and **smooth** out noisy signals.

# Artificial Intelligent (AI)

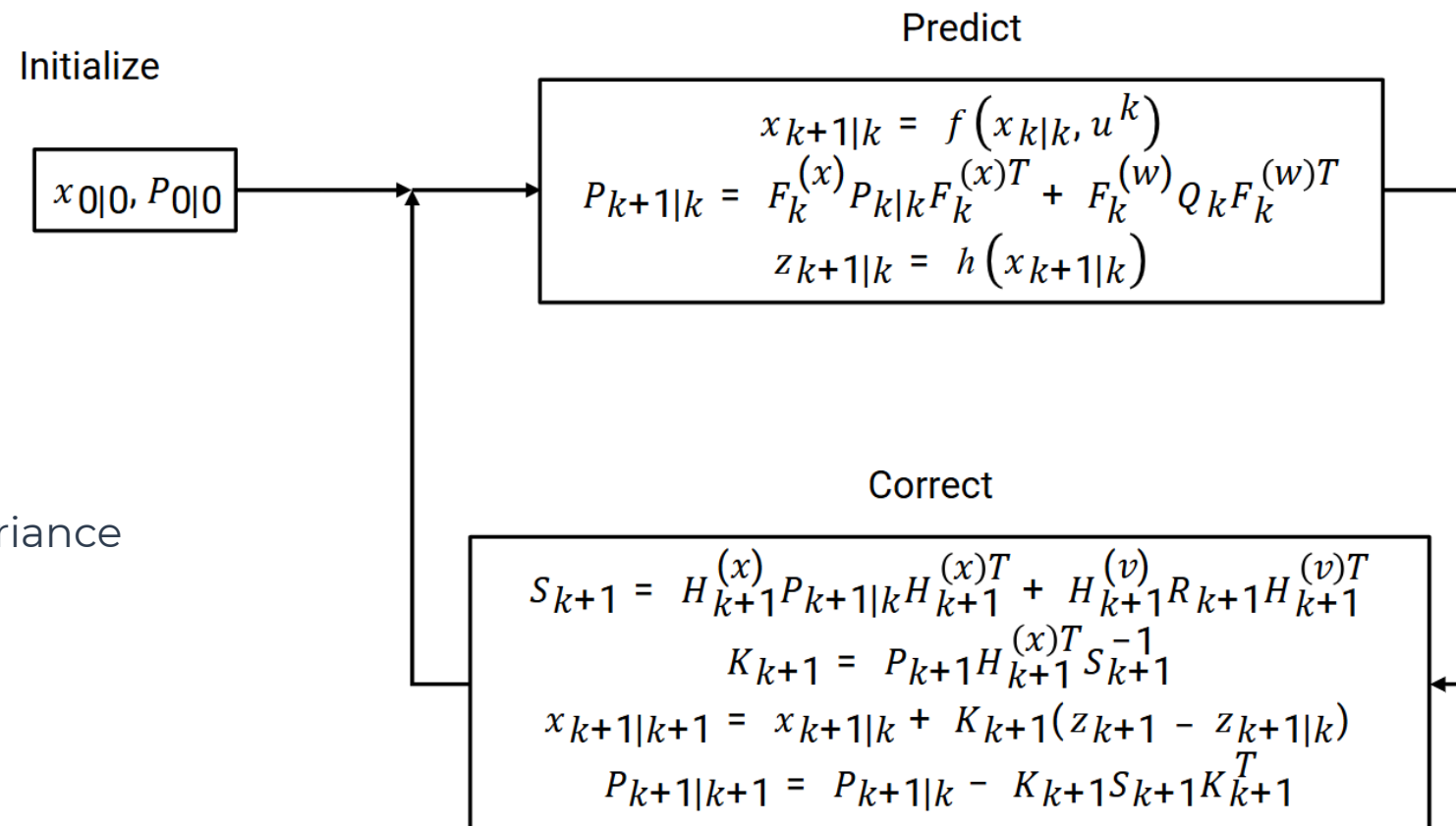
- ▶ General AI structure (Input → Hidden → Output).
- ▶ Learns patterns and noise.
- ▶ Adapts weights using gradient descent.



# Extend Kalman Filter

- Recursive estimation method, Prediction & Update steps(correct).

- Equations :



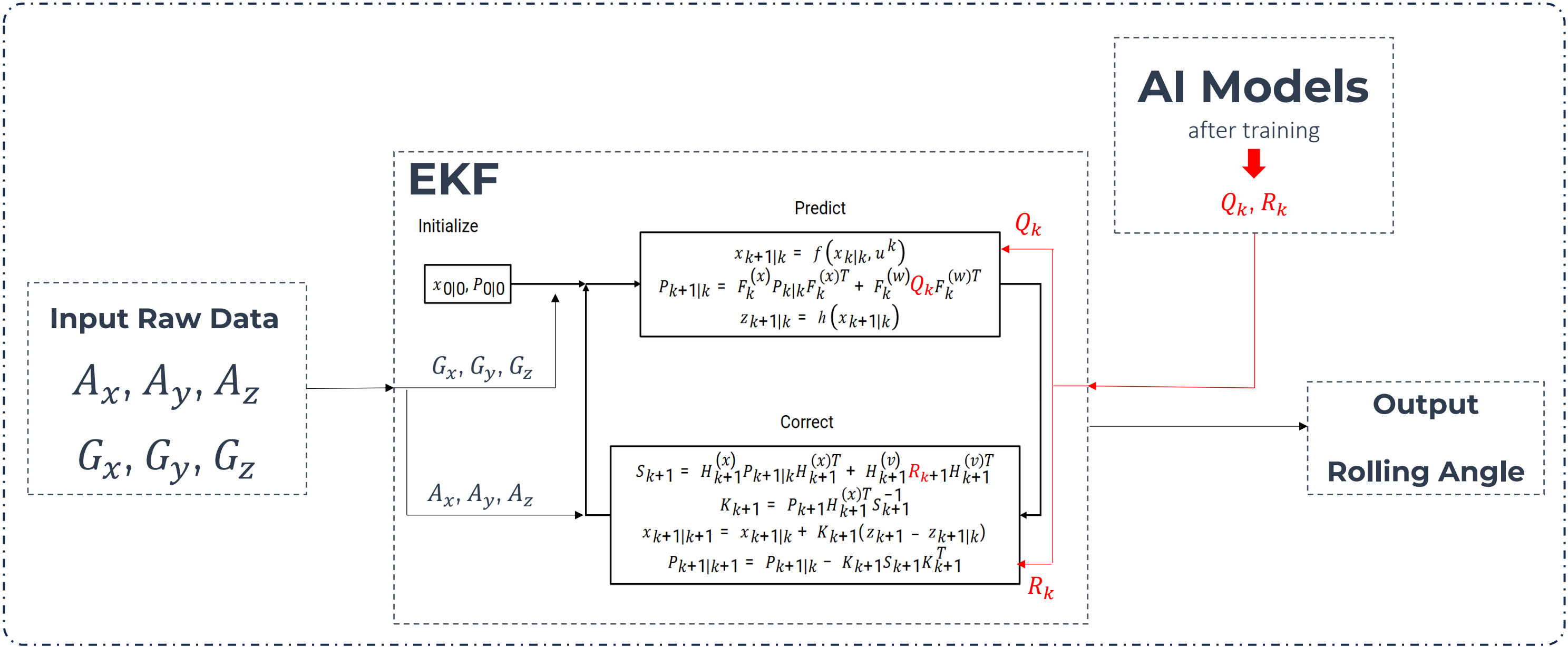
$R_K$  = Measurement Noise Covariance

$Q_K$  = Process Noise Covariance

$H_K$  = Measurement Matrix

$F_K$  = State Transition Matrix

# Extend Kalman Filter With AI



$A_x, A_y, A_z$  = Acceleration in Three Axis

$G_x, G_y, G_z$  = gyroscope in Three Axis



## Discussion – AI Model-EKF Advantages

- ▶ AI-Models improves sensor accuracy.
- ▶ Reduces sensor drift (Acceleration), sliding(gyroscope).
- ▶ decreases **RMSE** , **MSE** compared to standard EKF.
- ▶ Adapts to high-noise and dynamic Data.

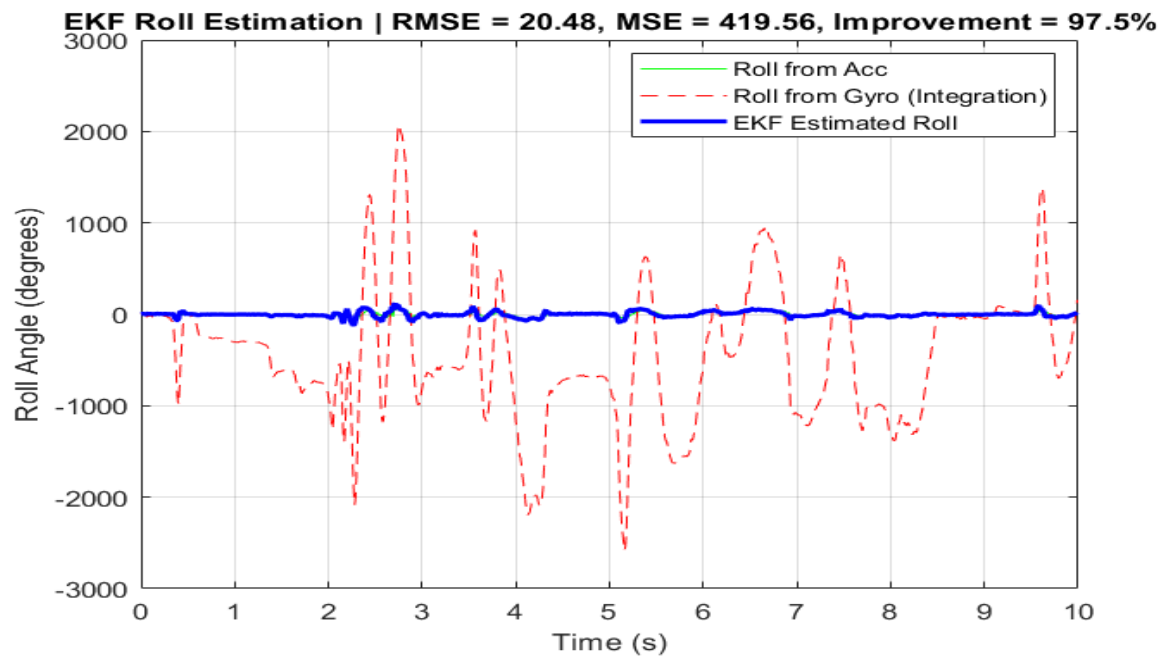
# Results.

Model	RMSE (deg)	MSE (deg <sup>2</sup> )	Improvement (%)
EKF	406.58	165309.96	51.10%
<b>GRU_EKF</b>	<b>9.2</b>	<b>84.59</b>	<b>98.90%</b>
FUZZY_EKF	15.44	238.27	98.10%
LSTM_EKF	20.48	419.56	97.50%
BILSTM_EKF	66.62	4438.6	92.00%
ANN_EKF	61.43	3773.77	92.60%

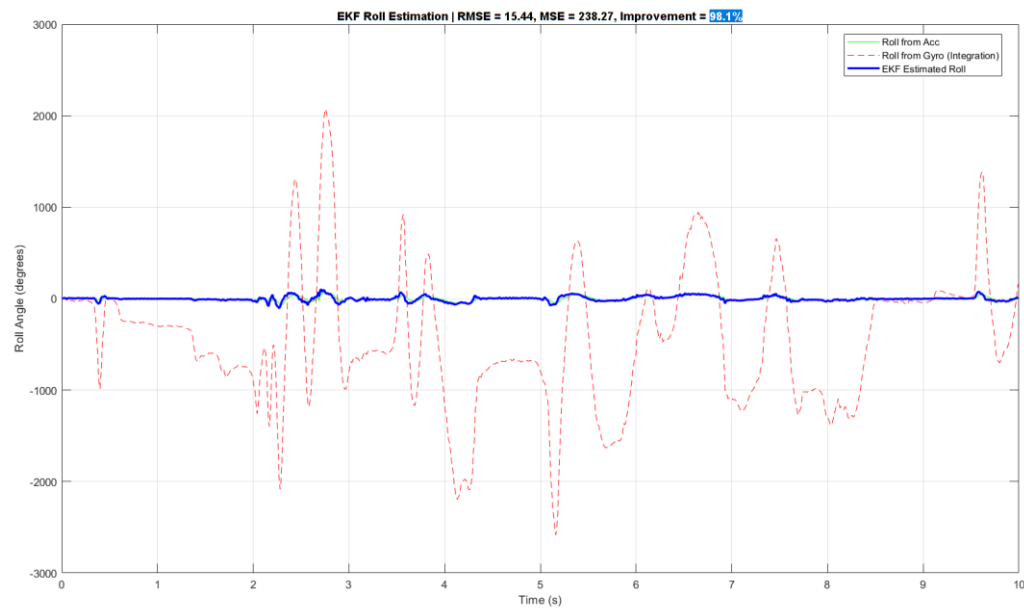
Several AI models were tested to enhance the performance of the Extended Kalman Filter (EKF) when dealing with noisy IMU data. As shown in the table, we evaluated each model using RMSE and MSE. Lower values mean better prediction accuracy.

# Results.

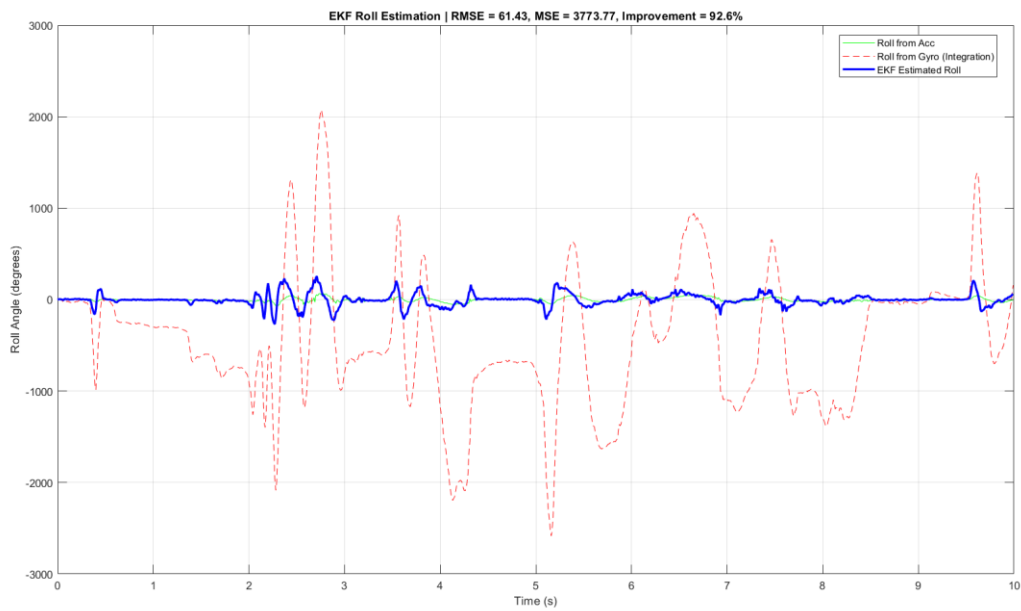
## LSTM-EKF



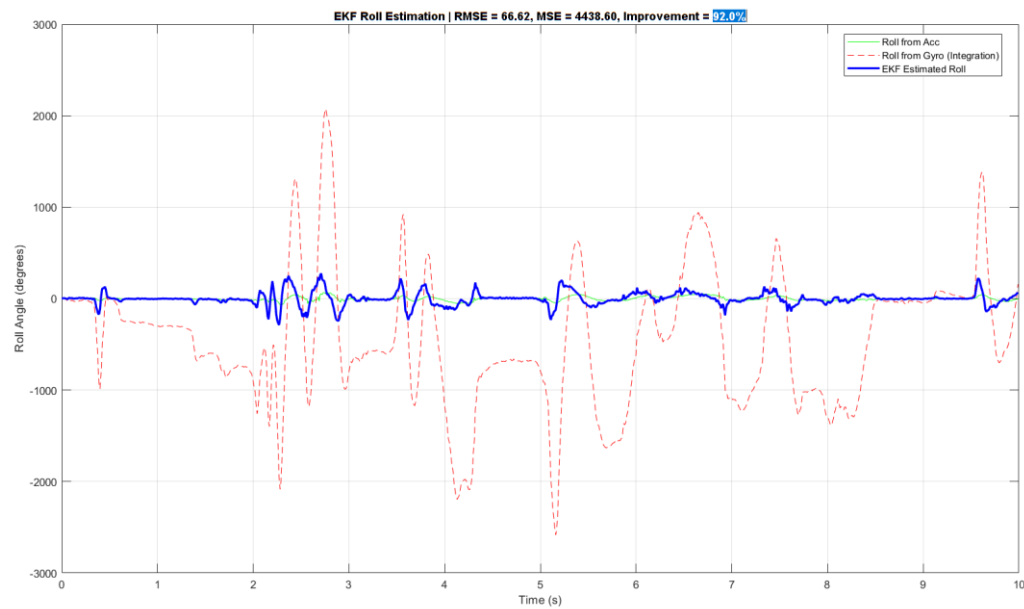
## Fuzzy-EKF



## ANN-EKF

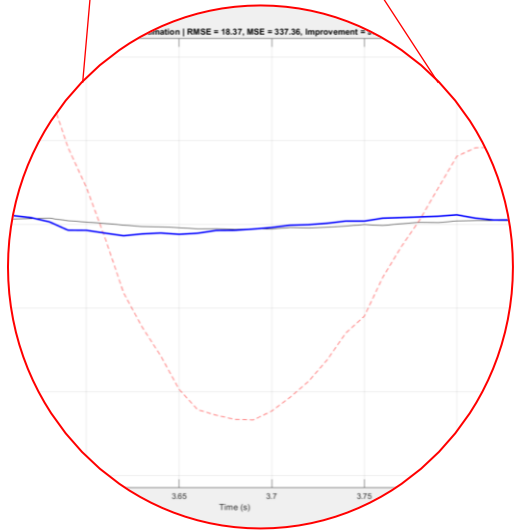
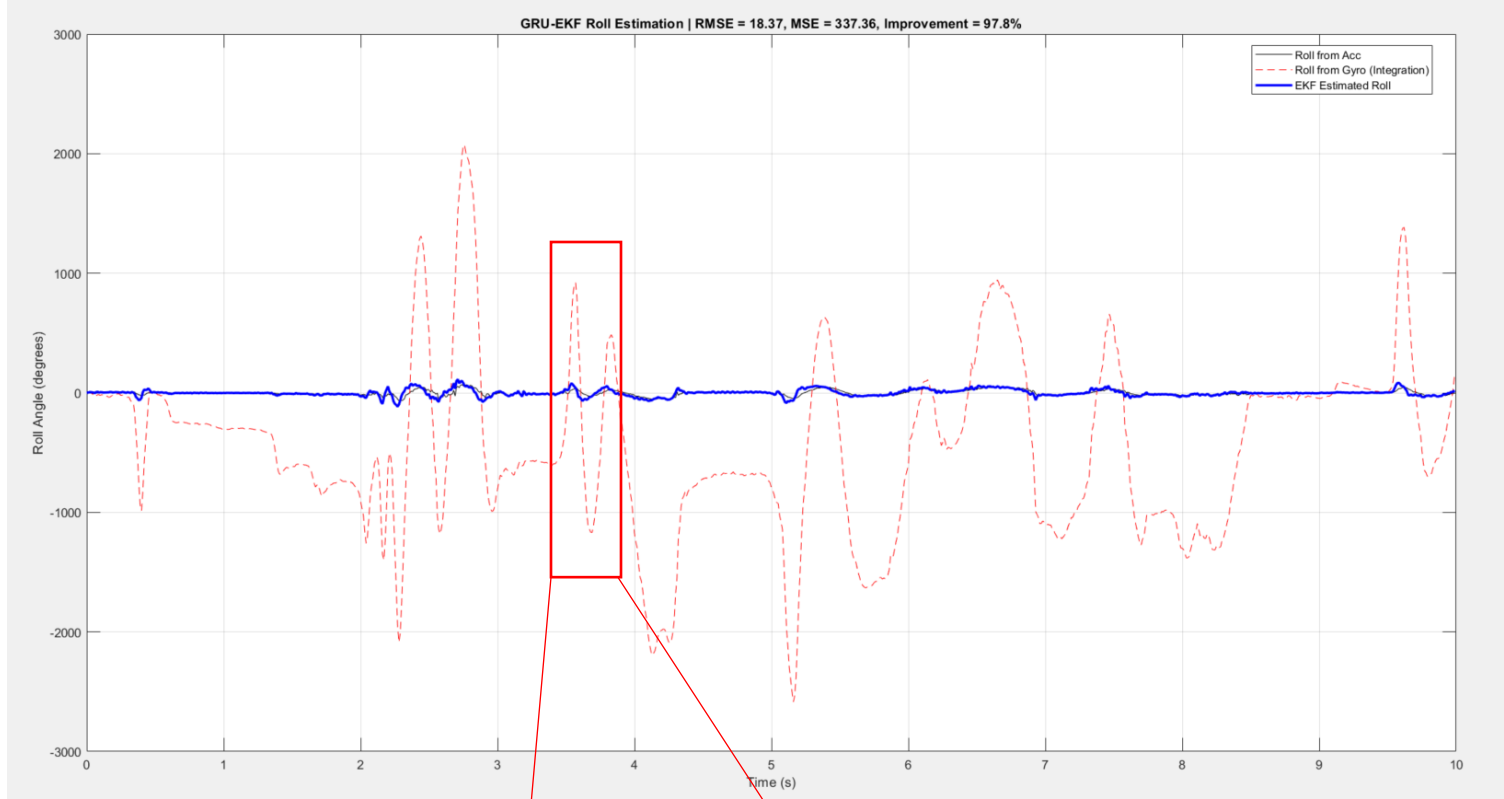


## BiLSTM-EKF

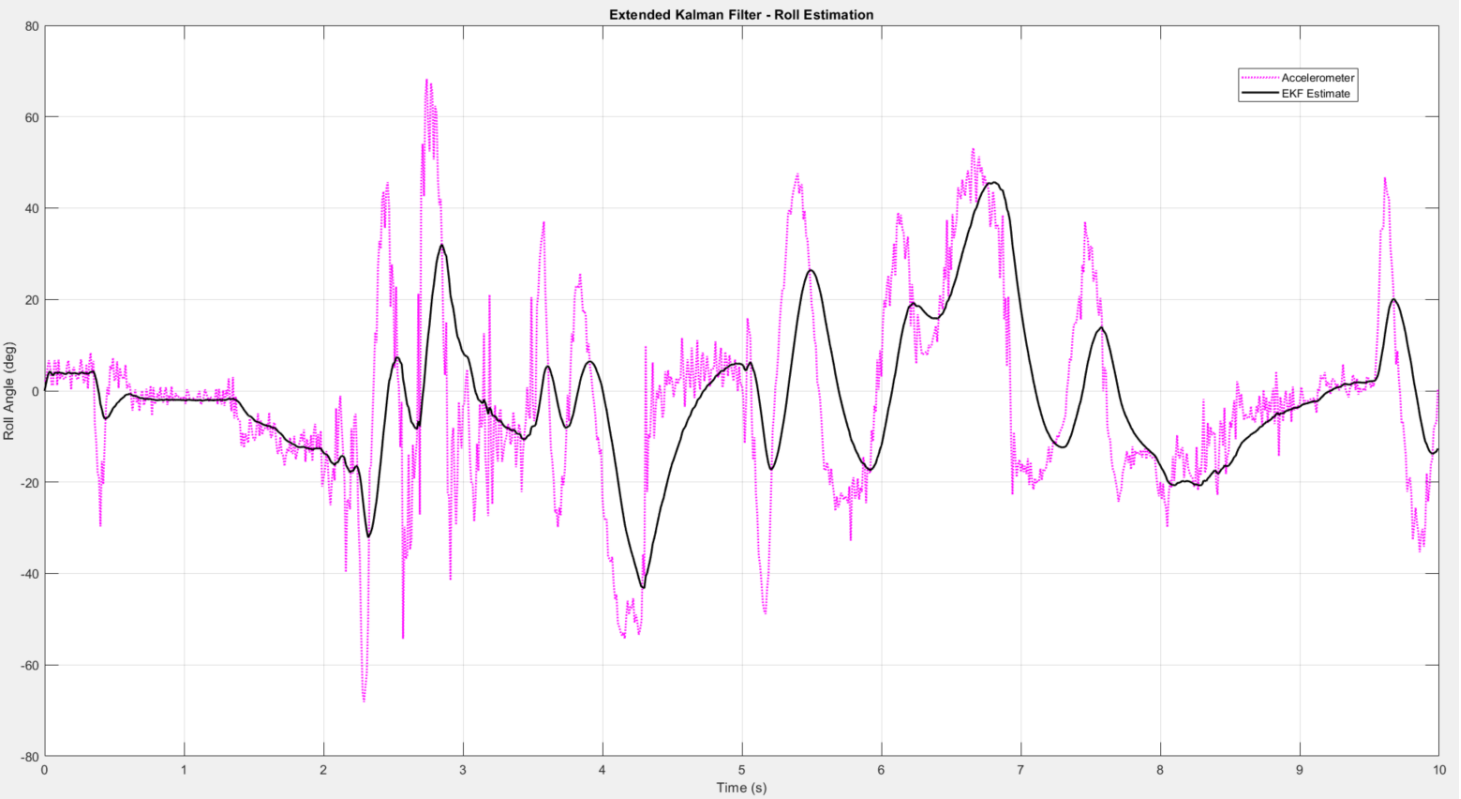


# Results.

## GRU-EKF



## EKF





# Conclusion

- ▶ **GRU + EKF** = Better Accuracy & Stability and Less Meas Square Error.
- ▶ Adaptive to real-time noise.
- ▶ Suitable for UAVs & navigation systems(GPS, IMU).
- ▶ The best performance was achieved by GRU\_EKF, which significantly reduced the error and achieved 98.9% improvement compared to the standard EKF.
- ▶ Other models like FUZZY\_EKF, LSTM\_EKF, and ANN\_EKF also showed strong improvements, but GRU had the lowest error overall.

## Future Work & Challenges

- ▶ More advanced AI models.
- ▶ Reduce **computation cost**.
- ▶ Collecting data from a real aircraft

**Thank You for Listening.**