Project Report on

KALMAN FILTERING FOR ACCURATE ACCELEROMETER AND GYROSCOPE MEASUREMENTS

Submitted By

Vaitla Hari Charan (21SS5A0413)

Konda Bhavana (20SS1A0428)

Kota Ishwarya (20SS1A0429)

Kadarla Nehashritha (20SS1A0424)

Submitted To

Mr. B.Srinivasa Rao

Scientist ‘F’

RESEARCH CENTRE IMARAT

DRDO

Hyderabad. Ministry of Defence

August 2023

RESEARCH CENTER IMARAT DRDO HYDERABAD

CERTIFICATE

This is to certify that this is a bonafide record of project work ____ done by

____

Under the guidance of_______________________________________

for internship completion on Embedded Systems at RCI DRDO HYD for academic session 2023-2024.

PROJECT GUIDE

Mr. B.SRINIVASA RAO Sc 'F

ACKNOWLEDGEMENTS

We avail this opportunity to express our sincere thanks to our esteemed guide ***Mr. B. Srinivasa Rao*** *for his splendid guidance, authentic supervisor, cooperation and moral support throughout this project.*

We would like to express our profound gratitude to Mr. B. Srinivasa Rao Sc 'F' for creating the required facilities to complete this project.

We would like to express our sincere thanks to our co-employees of D-HILS for giving all support and suggestions to complete our project

ABSTRACT:

The Kalman Filter, a cornerstone of modern estimation theory, has proven indispensable across an array of disciplines including control systems, signal processing, navigation, and more. Its elegant yet powerful nature lies in its capacity to seamlessly fuse imperfect measurements with predictions derived from mathematical models, thus enabling the accurate estimation of a system's underlying state. This comprehensive report delves into the heart of the Kalman Filter, elucidating its foundational theory, versatile applications, notable advantages, and inherent limitations, thereby offering a comprehensive understanding of its inner workings and real-world implications.

TABLE OF CONTENTS :

INTRODUCTION :

ABOUT PROJECT :

The Kalman Filter, developed by Rudolf E. Kálmán in the 1960s, is a state estimation algorithm that aims to provide accurate and efficient estimates of the true state of a dynamic system. It is particularly effective in scenarios where measurements are corrupted by noise and where there is a need to track the evolving state of a system in real time.

SCOPE OF PROJECT :

The Kalman filter, a versatile mathematical tool, finds application in diverse areas including navigation and tracking, control systems for improved stability, signal processing for denoising noisy data, computer vision tasks like object tracking, financial analysis, astronomy, and remote sensing, biomedical engineering for physiological signal monitoring, robotics to estimate position and orientation, aerospace applications such as satellite navigation, energy systems for monitoring grid stability, mechanical engineering for predicting system behavior, and many other fields, due to its ability to accurately estimate dynamic system states from noisy measurements while considering system behavior models.

MATHEMATICAL BASIS :

The Kalman Filter relies on two main steps: the prediction step (a priori estimation) and the update step (a posteriori estimation). These steps involve mathematical equations based on the principles of probability and linear algebra. The core equations of the Kalman Filter include:

1. ***Predict the current state of the system:***

   $S(k) = FS(k - 1) + GU(k)$

   S = state vector (Angle)
   F = state transition matrix (1)
   G = control matrix (0.004)
   U = input variable (Rate)

2. ***Calculate the uncertainty of the prediction:***

P(k) = FP (k - 1) * F ^ T + Q

P = prediction uncertainty vector (Un- certainty $_{\text{angle}}$)
Q=process uncertainty ($T_s^{2\cdot}$ 4^2)

3. ***Calculate the Kalman gain from the uncertainties on the predictions and measurements:***

L(k) = HP(k) * H ^ T + R

K = P(k) * (H ^ T)/(L(k)) = P(k) * H ^ T * L * (k) ^ - 1

L = Intermediate matrix
K = Kalman gain
H = Observation matrix (=1)
R = Measurement uncertainty ($T_s^{2\cdot}$ 3^2)

4. ***Update the predicted state of the system with the measurement of the state through the Kalman gain:***

S(k) S(k)+K (M(k)-H-S(k))

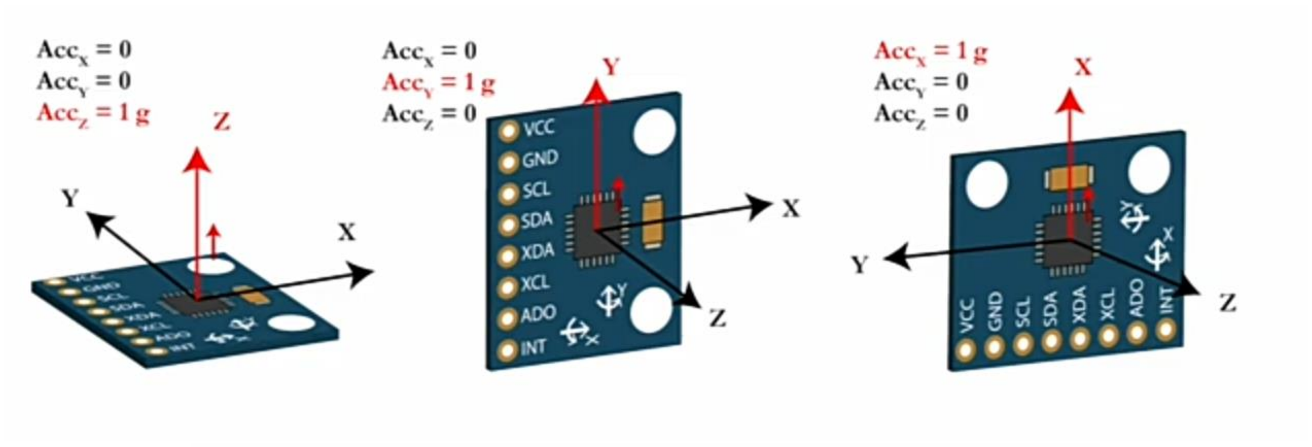
M=measurement vector (Angle)

5. ***Update the uncertainty of the predicted state:***

P(k)=(I-K-F)- P(k)

I=unity matrix (=1)

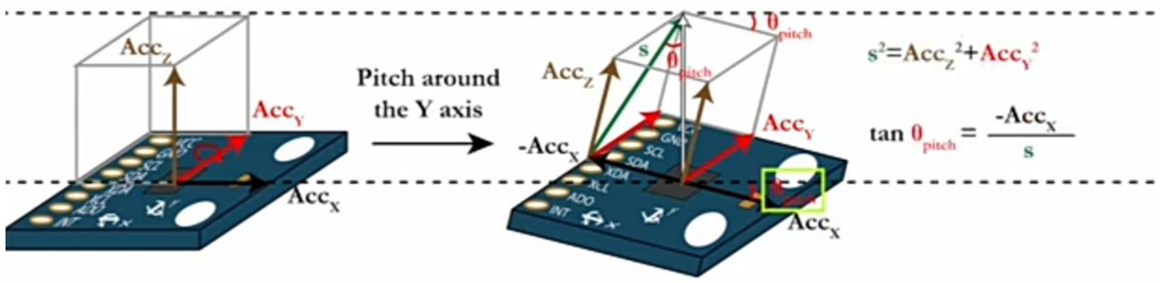
6. *To measure Pitch and Roll Angles :*



$$Angle_{pitch} = \int_0^{k\cdot T_s} Rate_{pitch}\cdot dt$$

$$Angle_{pitch}(k) = Angle_{pitch}(k-1) + Rate_{pitch}(k)\cdot T_s$$

$$Angle_{kalman}(k) = Angle_{kalman}(k-1) + T_s\cdot Rate(k)$$



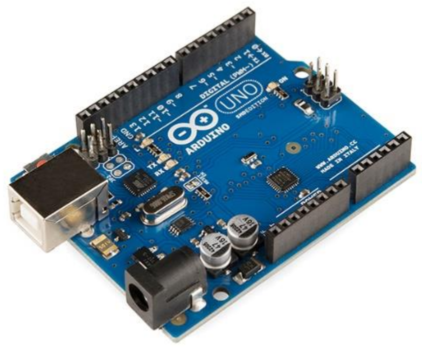
$$\theta_{pitch} = atan\left(\frac{-Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}}\right)$$

$$\theta_{roll} = atan\left(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right)$$

<u>HARDWARE REQUIREMENTS</u> :

Components: 1. ARDUINO UNO

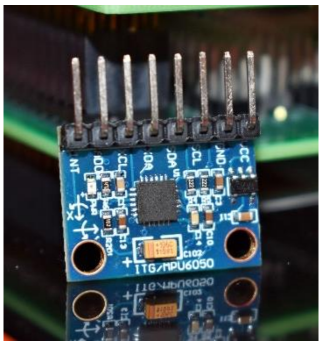
- The Arduino Uno is a popular microcontroller board renowned for its versatility in facilitating prototyping and DIY electronics projects.
- Featuring an ATmega328P microcontroller, it provides various digital and analogue I/O pins for connecting sensors, actuators, and peripherals.


ARDUINO UNO

- Arduino Uno's open-source nature and user-friendly development environment empower both beginners and experienced developers to create a wide range of applications.
- Its compatibility with numerous shields, add-on boards, and libraries enhances its capabilities, making it a preferred choice for makers and hobbyists.
- The board supports programming through the Arduino IDE, which simplifies code development and uploading to the board for seamless execution.
- Arduino Uno is a cornerstone in educational settings, fostering hands-on learning of programming, electronics, and embedded systems.
- Despite its strengths, the limited processing power and memory of the ATmega328P restrict its suitability for complex tasks and high-performance applications.

2. MPU-6050:

- The MPU-6050 is a popular Inertial Measurement Unit (IMU) sensor module that integrates a gyroscope and an accelerometer in a single package.
- With both 3-axis gyroscope and accelerometer sensors, the MPU-6050 enables precise motion tracking and orientation estimation.
- This sensor finds applications in robotics, drones, virtual reality, and motion-controlled devices due to its ability to capture rotational and linear movements.


MPU - 6050

- MPU-6050 employs Digital Motion Processor (DMP) technology to offload sensor fusion calculations, simplifying orientation determination for developers.
- The sensor communicates over the I2C interface, allowing it to be easily integrated with microcontrollers like Arduino Uno.
- Inertial sensors like MPU-6050 are essential components for achieving accurate motion detection and stabilization in a variety of projects.
- While the MPU-6050 is highly useful, its measurements can be affected by noise and drift, requiring calibration and filtering techniques for optimal accuracy.

3. Connecting Wires

4. Breadboard

<u>SOFTWARE REQUIREMENTS</u> :

SIMULATION SOFTWARE :

- **Wowki Online simulator**

  Used to simulate Arduino, ESP32, STM32, and many other popular boards, parts, and sensors.

Real-Time Software:

- **Arduino IDE :**

  The Arduino Integrated Development Environment (IDE) is a software platform designed to simplify the process of programming and uploading code to Arduino boards. It provides a user-friendly interface for writing, compiling, and uploading code to Arduino microcontrollers.

CIRCUIT DIAGRAM :

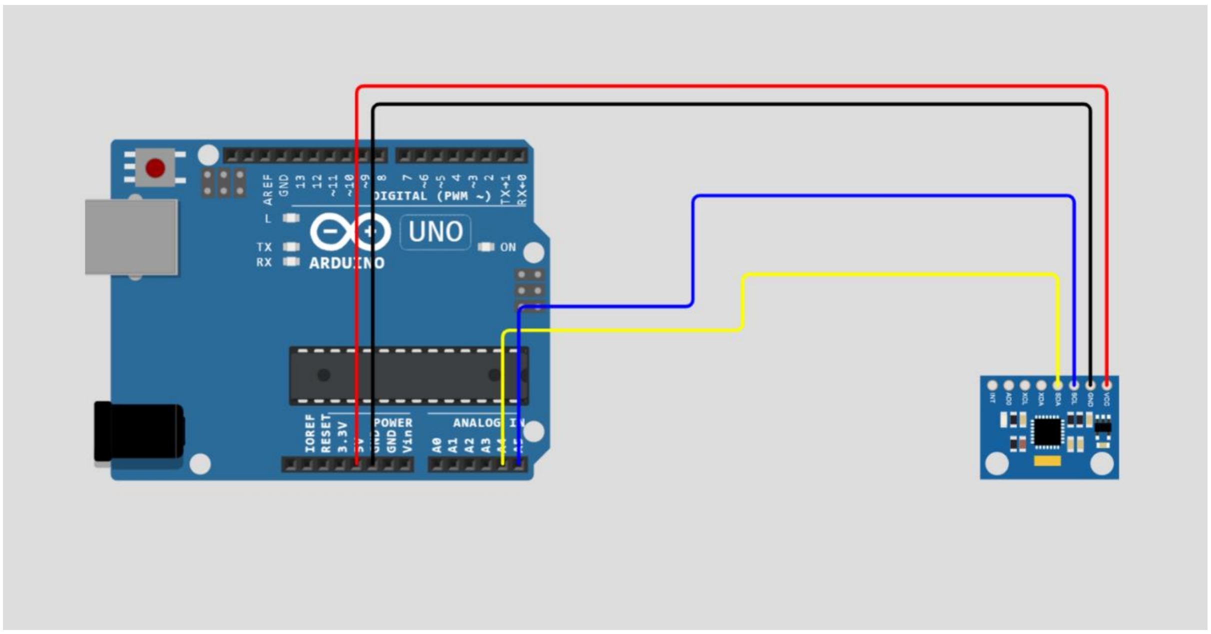
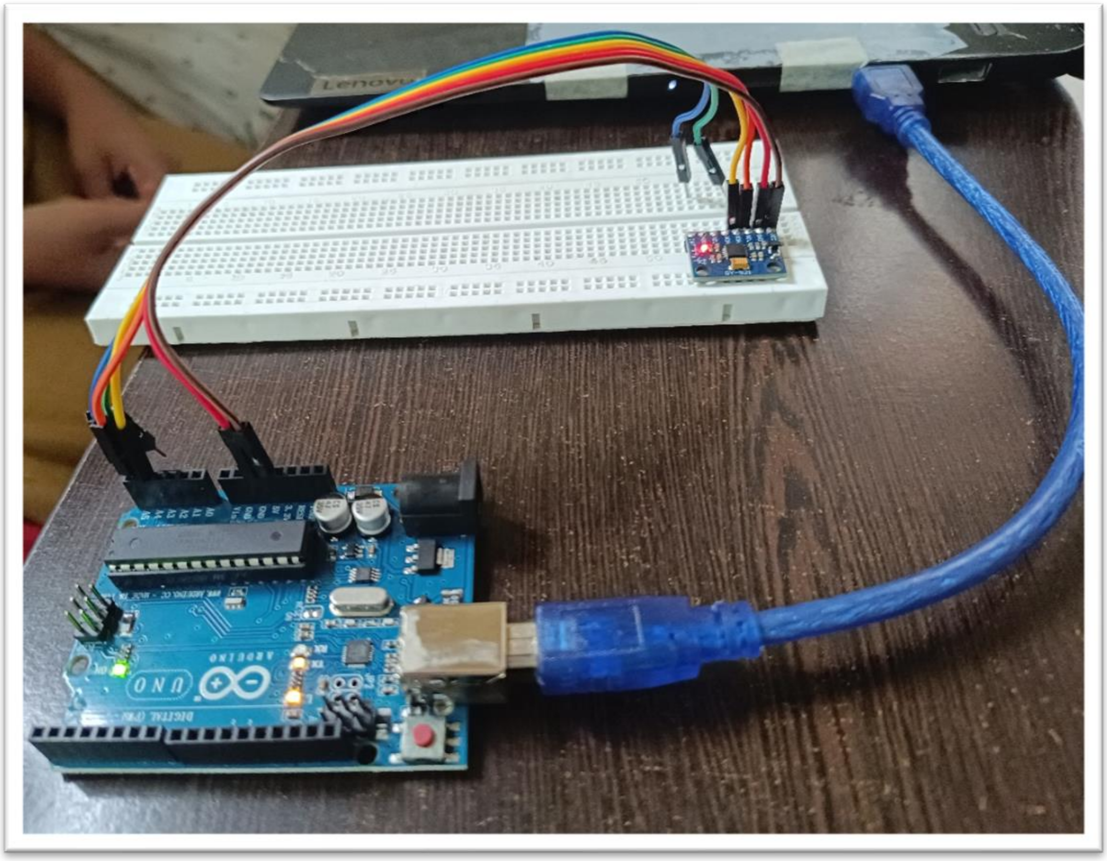


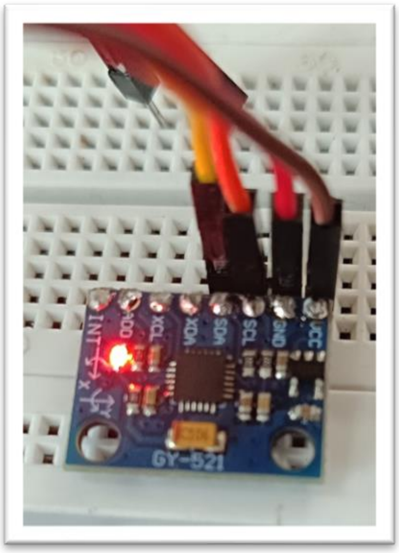
Figure 1Arduino Uno connected with MPU 6050

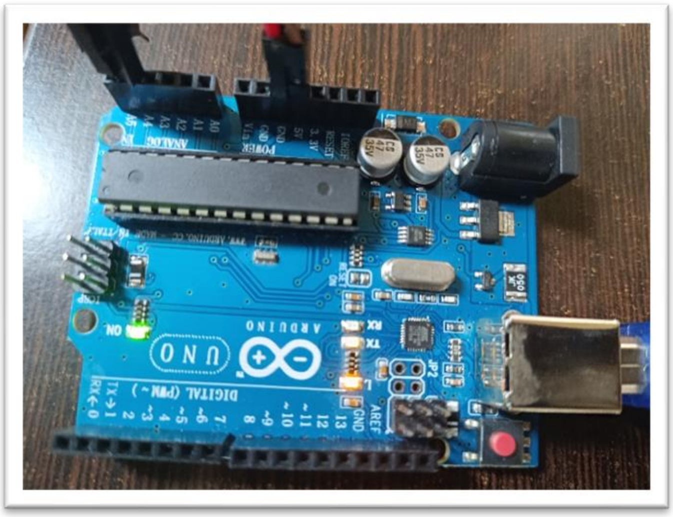
PIN CONNECTIONS :

Arduino UNO	MPU 6050
5V	Vcc
GND	GND
A5	SCL
A4	SDA

REAL-TIME HARDWARE SETUP :







Unfiltered MPU-6050 Output Analysis: Exploring Raw Data from Inertial Sensors

```
#include <Wire.h>
float RateRoll, RatePitch, RateYaw;
float AccX, AccY, AccZ;
float AngleRoll, AnglePitch;
float LoopTimer;
void gyro_signals(void) {
  Wire.beginTransmission(0x68);
  Wire.write(0x1A);
  Wire.write(0x05);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission();
  Wire.requestFrom(0x68,6);
  int16_t AccXLSB = Wire.read() << 8 | Wire.read();
  int16_t AccYLSB = Wire.read() << 8 | Wire.read();
  int16_t AccZLSB = Wire.read() << 8 | Wire.read();
  Wire.beginTransmission(0x68);
  Wire.write(0x1B);
  Wire.write(0x8);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x43);
  Wire.endTransmission();
  Wire.requestFrom(0x68,6);
  int16_t GyroX=Wire.read()<<8 | Wire.read();
  int16_t GyroY=Wire.read()<<8 | Wire.read();
  int16_t GyroZ=Wire.read()<<8 | Wire.read();
  RateRoll=(float)GyroX/65.5;
  RatePitch=(float)GyroY/65.5;
  RateYaw=(float)GyroZ/65.5;
  AccX=(float)AccXLSB/4096;
  AccY=(float)AccYLSB/4096;
```

```
 AccZ=(float)AccZLSB/4096;
 AngleRoll=atan(AccY/sqrt(AccX*AccX+AccZ*AccZ))*1/(3.142/180);
 AnglePitch=-atan(AccX/sqrt(AccY*AccY+AccZ*AccZ))*1/(3.142/180);
}
void setup() {
 Serial.begin(57600);
 pinMode(13, OUTPUT);
 digitalWrite(13, HIGH);
 Wire.setClock(400000);
 Wire.begin();
 delay(250);
 Wire.beginTransmission(0x68);
 Wire.write(0x6B);
 Wire.write(0x00);
 Wire.endTransmission();
}
void loop() {
 gyro_signals();
  Serial.print("Roll Angle [°]: ");
 Serial.print(AngleRoll);
 Serial.print(" Pitch Angle [°]: ");
 Serial.println(AnglePitch);
 delay(50);
}
```

Filtered MPU-6050 Output Analysis: Enhancing Inertial Sensor Data with Kalman Filtering

```
#include <Wire.h>

float RateRoll, RatePitch, RateYaw;
float RateCalibrationRoll, RateCalibrationPitch, RateCalibrationYaw;
int RateCalibrationNumber;
float AccX, AccY, AccZ;
float AngleRoll, AnglePitch;
uint32_t LoopTimer;
float KalmanAngleRoll = 0;
float KalmanUncertaintyAngleRoll = 2 * 2;
float KalmanAnglePitch = 0;
float KalmanUncertaintyAnglePitch = 2 * 2;
float Kalman1DOutput[] = {0, 0};
void kalman_1d(float &KalmanState, float &KalmanUncertainty, float
   KalmanInput, float KalmanMeasurement) {
  KalmanState = KalmanState + 0.004 * KalmanInput;
  KalmanUncertainty = KalmanUncertainty + 0.004 * 0.004 * 4 * 4;
  float KalmanGain = KalmanUncertainty / (KalmanUncertainty + 3 * 3);
  KalmanState = KalmanState + KalmanGain * (KalmanMeasurement -
   KalmanState);
  KalmanUncertainty = (1 - KalmanGain) * KalmanUncertainty;
  Kalman1DOutput[0] = KalmanState;
  Kalman1DOutput[1] = KalmanUncertainty;
}
void gyro_signals(void) {
  Wire.beginTransmission(0x68);
  Wire.write(0x1A);
  Wire.write(0x05);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 6);
  int16_t AccXLSB = Wire.read() << 8 | Wire.read();
  int16_t AccYLSB = Wire.read() << 8 | Wire.read();
  int16_t AccZLSB = Wire.read() << 8 | Wire.read();
  Wire.beginTransmission(0x68);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();
  Wire.beginTransmission(0x68);
  Wire.write(0x43);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 6);
  int16_t GyroX = Wire.read() << 8 | Wire.read();
  int16_t GyroY = Wire.read() << 8 | Wire.read();
  int16_t GyroZ = Wire.read() << 8 | Wire.read();
  RateRoll = (float)GyroX / 65.5;
  RatePitch = (float)GyroY / 65.5;
  RateYaw = (float)GyroZ / 65.5;
  AccX = (float)AccXLSB / 4096;
  AccY = (float)AccYLSB / 4096;
  AccZ = (float)AccZLSB / 4096;
  AngleRoll = atan(AccY / sqrt(AccX * AccX + AccZ * AccZ)) * (180 / 3.141);
  AnglePitch = -atan(AccX / sqrt(AccY * AccY + AccZ * AccZ)) * (180 / 3.141);
}
void setup() {
  Serial.begin(57600);
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  Wire.setClock(400000);
  Wire.begin();
  delay(250);
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();
for (RateCalibrationNumber = 0; RateCalibrationNumber < 2000;
    RateCalibrationNumber++) {
    gyro_signals();
    RateCalibrationRoll += RateRoll;
    RateCalibrationPitch += RatePitch;
    RateCalibrationYaw += RateYaw;
    delay(1);
   }

  RateCalibrationRoll /= 2000;
  RateCalibrationPitch /= 2000;
  RateCalibrationYaw /= 2000;

  LoopTimer = micros();
}
void loop() {
  gyro_signals();
  RateRoll -= RateCalibrationRoll;
  RatePitch -= RateCalibrationPitch;
  RateYaw -= RateCalibrationYaw;
  kalman_1d(KalmanAngleRoll, KalmanUncertaintyAngleRoll, RateRoll, AngleRoll);
  KalmanAngleRoll = Kalman1DOutput[0];
  KalmanUncertaintyAngleRoll = Kalman1DOutput[1];
  kalman_1d(KalmanAnglePitch, KalmanUncertaintyAnglePitch, RatePitch,
    AnglePitch);
  KalmanAnglePitch = Kalman1DOutput[0];
  KalmanUncertaintyAnglePitch = Kalman1DOutput[1];
  Serial.print("Roll Angle [°]: ");
  Serial.print(KalmanAngleRoll);
  Serial.print(" Pitch Angle [°]: ");
Serial.println(KalmanAnglePitch);
  while (micros() - LoopTimer < 4000);
  LoopTimer = micros();
}
```

A Study of Simulation vs. Real-Time Outputs: Performance Evaluation and Insights :

1. **SIMULATION OUTPUTS**:

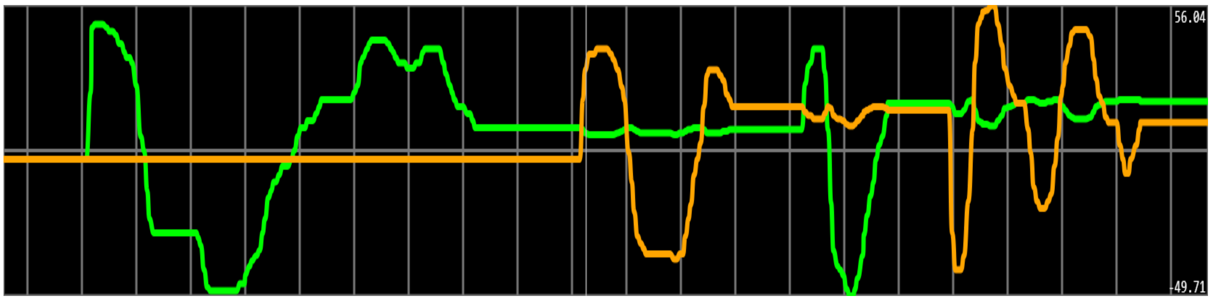


Figure 2Output Without using the Kalman filter

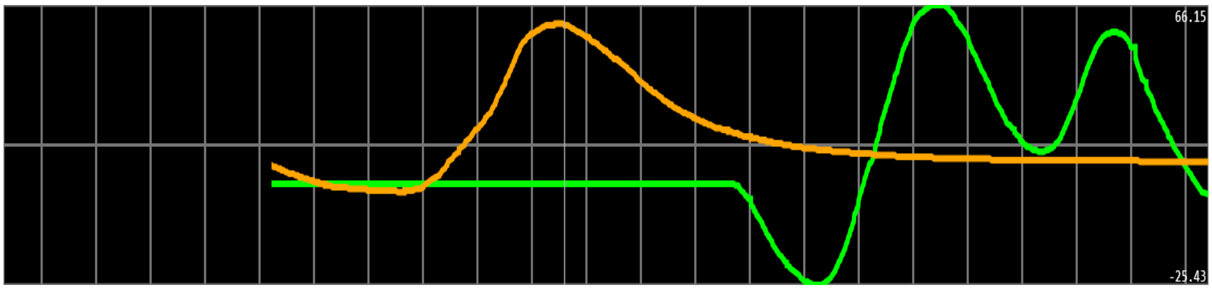


Figure 3: Output With using Kalman filter

2. **Real-Time Outputs:**

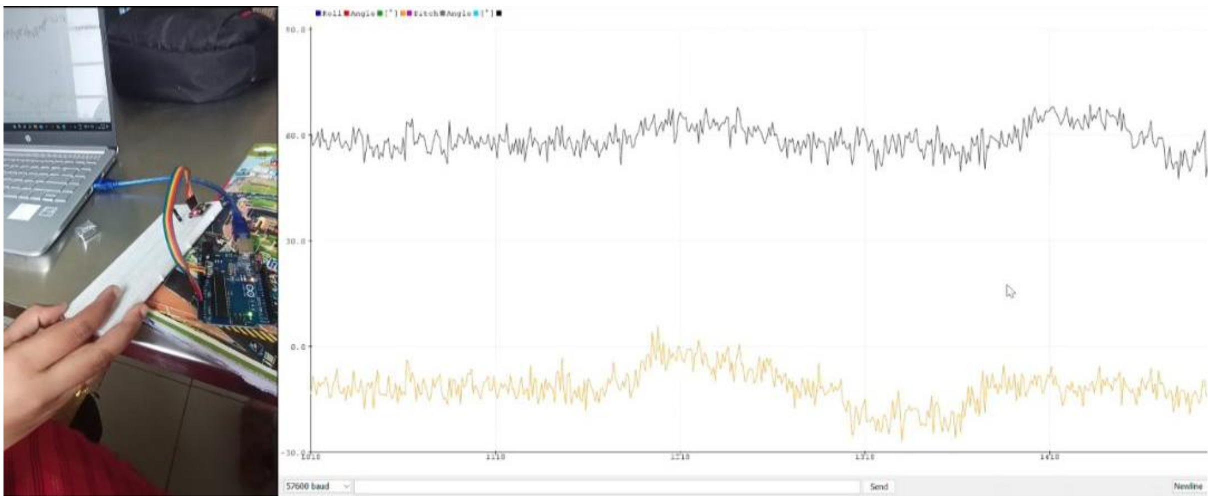


Figure 4Output Without using the Kalman filter

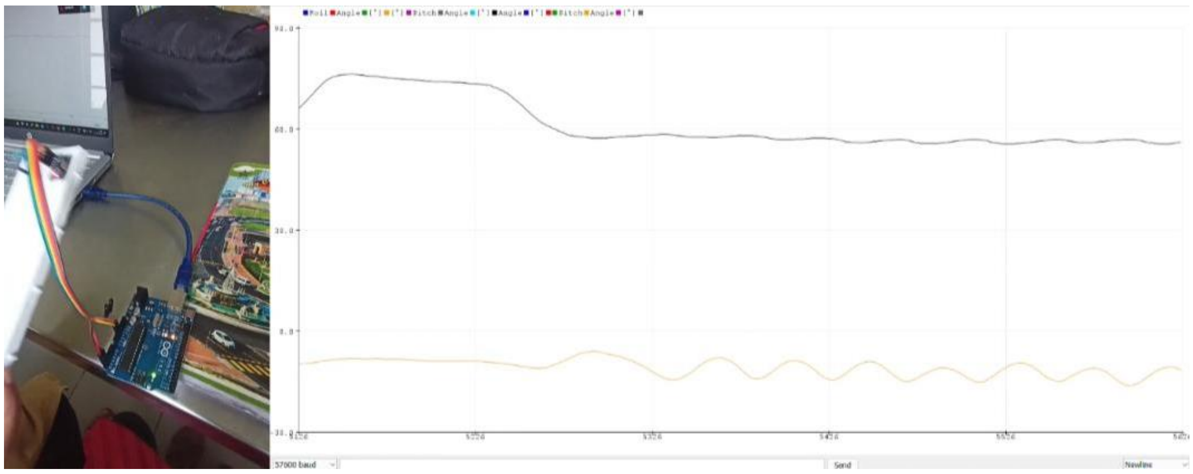


Figure 5Output With using Kalman filter

APPLICATION :

The Kalman Filter finds applications in various domains, including:

- **Navigation:** GPS systems, autonomous vehicles, aircraft navigation.
- **Control Systems:** Real-time control of industrial processes, robotics, and aerospace systems.
- **Signal Processing:** Speech and image processing, radar tracking, sensor fusion.
- **Finance:** Time-series analysis, stock market prediction.

LIMITATIONS :

- **Linearity and Gaussian Assumption:** The Kalman Filter assumes linearity and Gaussian noise, which might not hold in all scenarios.
- **Complexity for Nonlinear Systems:** For nonlinear systems, extended variants like the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) are used, but they can be more complex and computationally intensive.
- **Sensitivity to Model Errors:** Errors in the system model or noise assumptions can lead to suboptimal estimates.

CONCLUSION :

The realm of applications where the Kalman Filter shines is vast and impactful. From the precision-guided navigation of GPS systems and the seamless coordination of autonomous vehicles to the intricate control of industrial processes and the awe-inspiring capabilities of robotics, the Kalman Filter stands as a cornerstone of modern technology. Its versatility extends to aerospace systems, ensuring the flawless navigation of aircraft through the skies. Beyond the physical realm, its prowess in signal processing becomes evident in its adept handling of speech and image processing tasks, making it an unsung hero in the enhancement of sensory data. The radar tracking domain benefits immensely from its ability to sift through noisy input and provide crystal-clear insights. In the world of finance, where the fluctuations of the stock market often seem unpredictable, the Kalman Filter emerges as a guiding light, aiding in time-series analysis and even influencing stock market predictions.

In every corner it touches, the Kalman Filter transforms chaos into order, noise into clarity, and uncertainty into confidence. Its influence spans industries and technologies, shaping our modern world and propelling us towards new heights of precision, efficiency, and innovation. As we navigate the complex web of dynamic systems, the Kalman Filter stands as an essential ally, an algorithmic maestro orchestrating the symphony of data and dynamics.

REFERENCES :

1. ***YOUTUBE Channel:*** **Carbon Aeronautics** ***(https://youtu.be/5HuN9iL-zxU)***
2. ***Wikipedia (https://en.wikipedia.org/wiki/Kalman_filter)***
3. ***ChatGPT, etc,.***