IMU 6 DOF Attitude Estimation with Kalman Filter Sensor Fusion

Michael Pittenger

EE 782

Fall 2024

Overview

- Introduction
- Devices
- Arduino Sensor Readings
- Kalman Filtering Without Bias
- Kalman Filtering With Bias and Sensor Fusion
- Live Arduino Kalman Filter Results
- Conclusion

Overview

Introduction

- An Inertial Measurement Unit (IMU) uses accelerometers, gyroscopes, and sometimes magnetometers to measure forces, angular rates, and orientation—enabling the calculation of state variables like attitude, angular rates, velocity, and position for navigation and maneuvering.
- The focus is of this project on fusing accelerometer and gyroscope measurements of a lost-cost, 6-DOF (degrees of freedom) IMU through Kalman filtering to achieve accurate and reliable orientation tracking (attitude estimation).
- By fusing data from accelerometers and gyroscopes (and optionally magnetometers), and applying noise reduction techniques such as Kalman Filtering, IMUs can accurately estimate tilt and orientation, despite inherent sensor noise and drift.

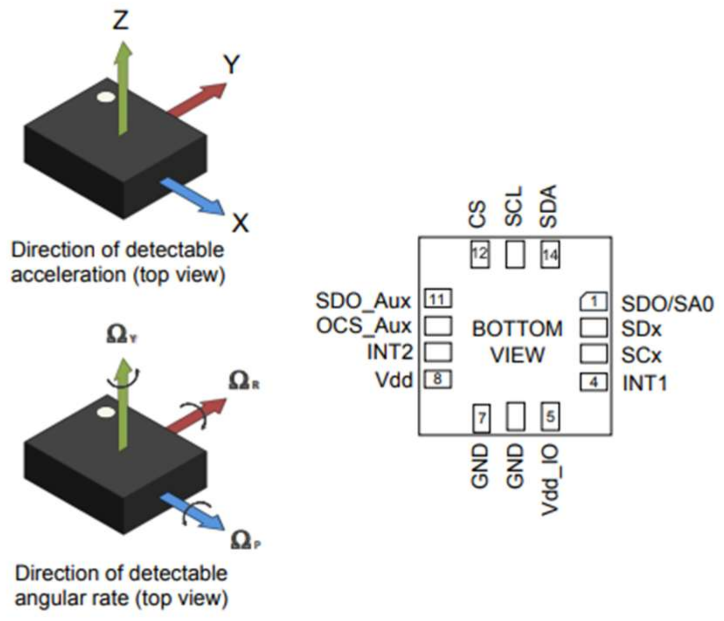

Z
Y
X
Direction of detectable acceleration (top view)
Ω_Y
Ω_R
Ω_P
Direction of detectable angular rate (top view)
CS
SCL
SDA
12
14
SDO_Aux
OCS_Aux
INT2
Vdd
11
8
BOTTOM VIEW
1
4
SDO/SA0
SDx
SCx
INT1
7
5
GND
GND
Vdd_IO

Figure 1: ISM330DHCX Pin Connections

Overview

Devices

- Adafruit ISM330DHCX 6-DoF IMU
  - A 6-DoF IMU with a 3-axis accelerometer and a 3-axis gyroscope
  - Its accelerometer range spans ±2/±4/±8/±16 g at update rates from 1.6 Hz to 6.7 KHz, and its gyroscope covers ±125 to ±4000 dps at 12.5 Hz to 6.7 KHz
  - It communicates via SPI or I2C, operates on 3V or 5V, and easily interfaces with Arduino boards
  - Free Arduino and Python libraries, along with example code and circuit diagrams, make it straightforward to extract IMU data (m/s and rad/s) and integrate into various projects
- Arduino MEGA2560
  - Support for UART, SPI, and I2C, as well as both USB and external power inputs
  - Its accessible libraries, IDE, and broad community support make it an ideal platform for prototyping and experimentation
  - Baud rate of 115200 is sufficient for this application

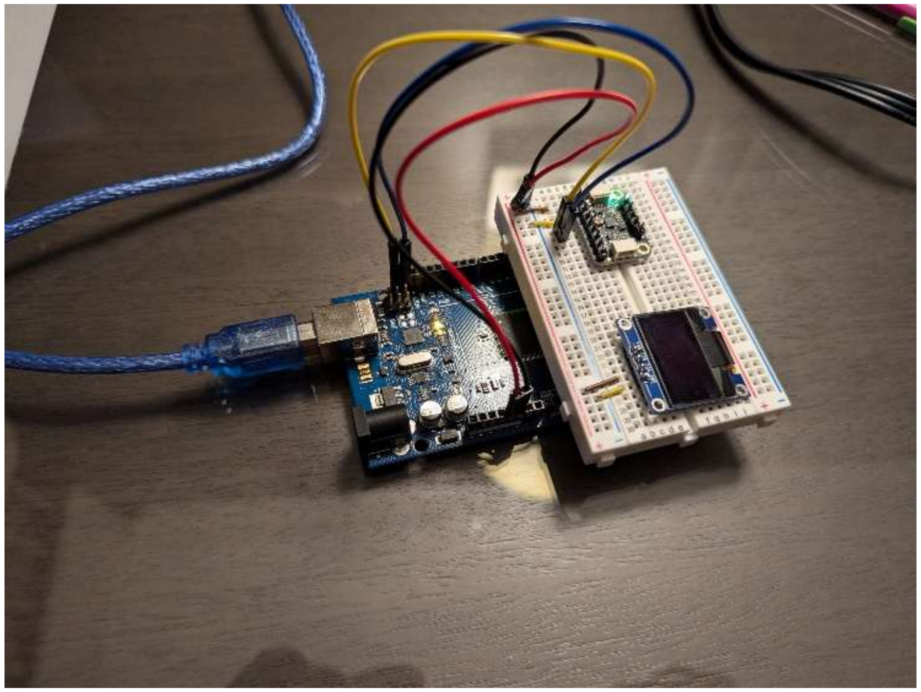


Figure 2: Arduino + IMU Setup

Overview

Arduino Sensor Readings

- Orientation calculation:

$$\theta_{AccPitch} = atan2(AccData_z, AccData_x)(1)$$

$$\theta_{AccRoll} = atan2(AccData_z, AccData_y)(2)$$

  - To calculate the orientation of the IMU, we use the known equations of the Euler angles to convert the raw sensor readings (m/s) into the pitch and roll of the device

- Results
  - Angles appear to write to Arduino IDE serial monitor correctly

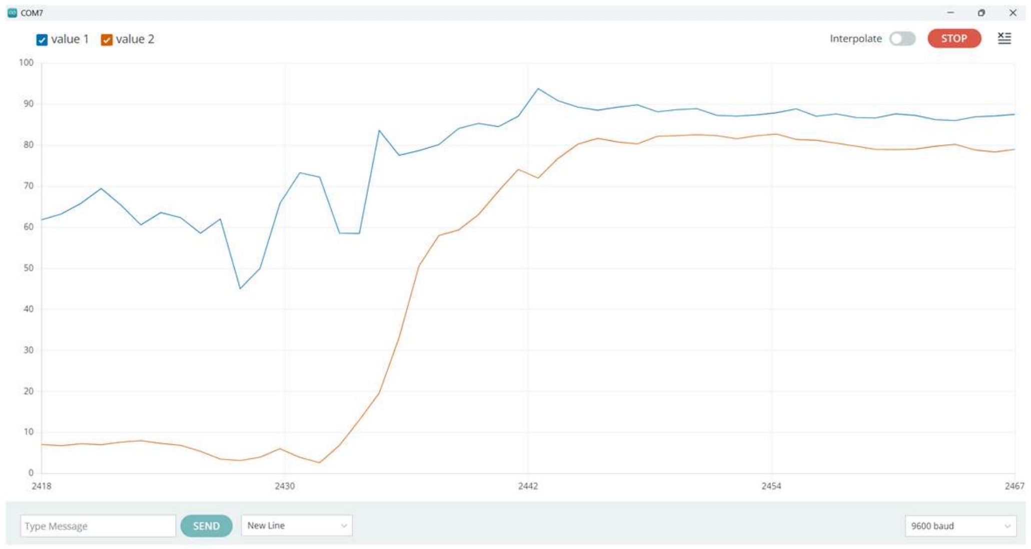


Figure 3: Live Arduino Roll from 0 to 90 Degrees

Arduino Sensor Readings

- Adafruit provides 3D model viewer that takes Euler angles and outputs the orientation through a model of a rabbit
- While the noise was very visible, the rabbit model moved along with the IMU in real space, showing that the IMU readings and Euler angle calculations were valid

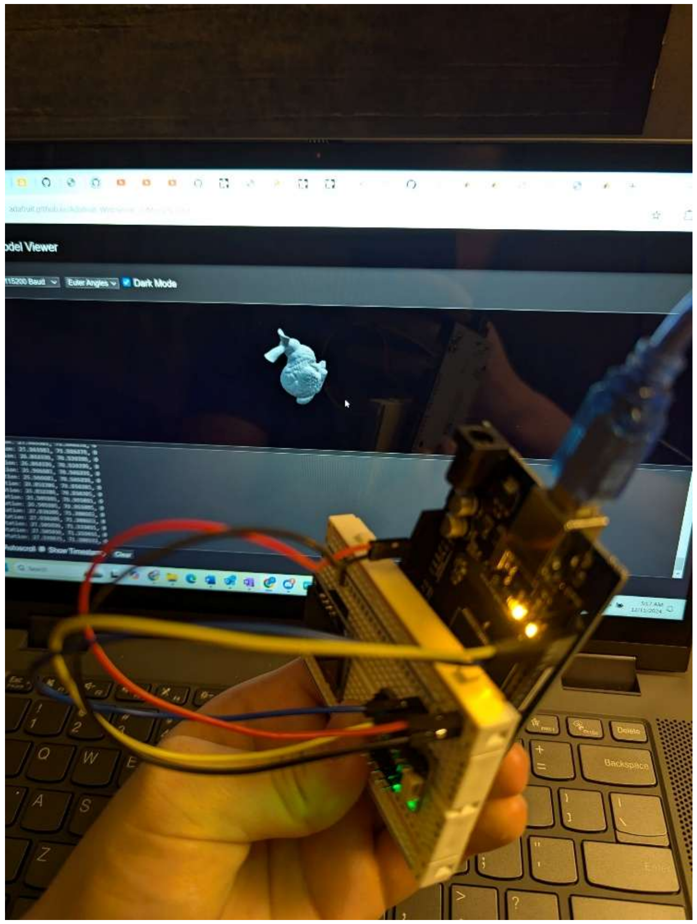


Figure 4: Adafruit 3D Model Viewer

Overview

Kalman Filtering Without Bias

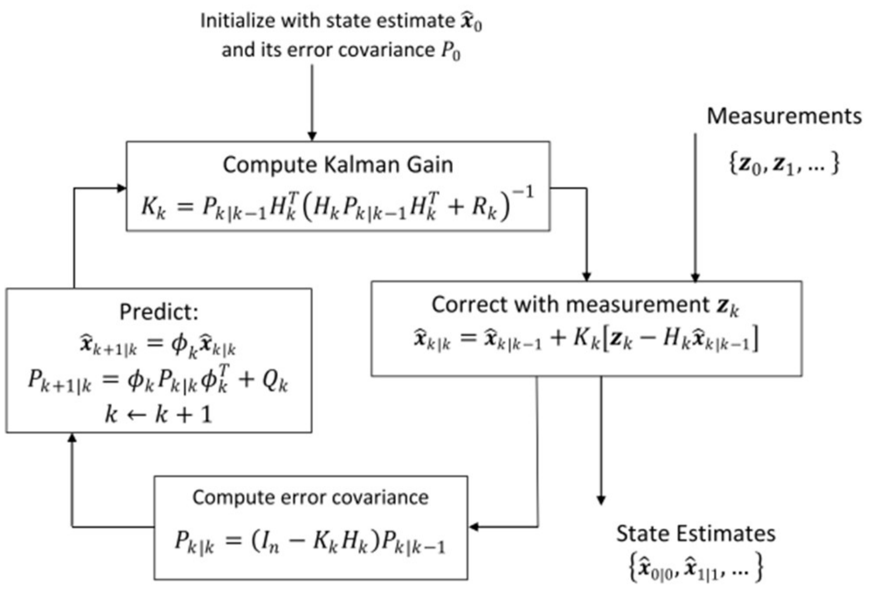
- Kalman filter equations:

$$\boldsymbol{\alpha_{Acc} = \alpha_{extra} - g + b_a + n_a}$$
$$\boldsymbol{\omega_{gyro} = \omega + b_g + n_g}$$

$$\boldsymbol{x_k = Fx_{k-1} + Bu_k + w_k}$$
$$\boldsymbol{z_k = Hx_k + v_k}$$

$$\boldsymbol{x_k = [\theta]_k}$$
$$\boldsymbol{F = [1]}$$
$$\boldsymbol{B = [0]}$$
$$\boldsymbol{H = [1]}$$

- Where $\boldsymbol{x_k}$ is the angle at time $\boldsymbol{k}$ in degrees, $\boldsymbol{F}$ is the state transition model, $\boldsymbol{B}$ is the control-input model, $\boldsymbol{u_k}$ is the gyroscope measurement in degrees/second, $\boldsymbol{w_k}$ is the process noise, $\boldsymbol{z_k}$ is the measurement, $\boldsymbol{H}$ is the observation model, and $\boldsymbol{v_k}$ is the measurement noise.



Initialize with state estimate $\hat{x}_0$ and its error covariance P_0
Measurements
$\{z_0, z_1, \dots\}$
Compute Kalman Gain
$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1}$
Predict:
$\hat{x}_{k+1|k} = \phi_k\hat{x}_{k|k}$
$P_{k+1|k} = \phi_kP_{k|k}\phi_k^T + Q_k$
$k \leftarrow k+1$
Correct with measurement z_k
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k[z_k - H_k\hat{x}_{k|k-1}]$
Compute error covariance
$P_{k|k} = (I_n - K_kH_k)P_{k|k-1}$
State Estimates
$\{\hat{x}_{0|0}, \hat{x}_{1|1}, \dots\}$

Figure 5: Block Diagram for Discrete Time Kalman Filter

Kalman Filtering Without Bias

- The IMU’s angular data was processed with a Kalman filter, using an R value of 0.3 to balance noise reduction and response lag
- Tests involved smooth rotations along pitch and roll axes, as well as shaking along x, y, and z axes to simulate vibration
- The filter significantly reduced noise, but without bias correction, the device still experiences drift over time

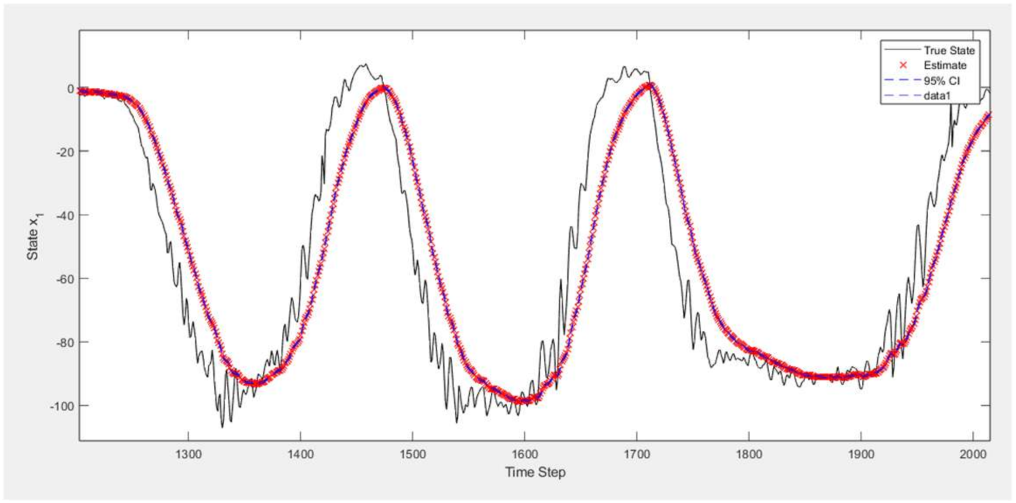


Figure 6: Pitch Angle Kalman Filter Rotation

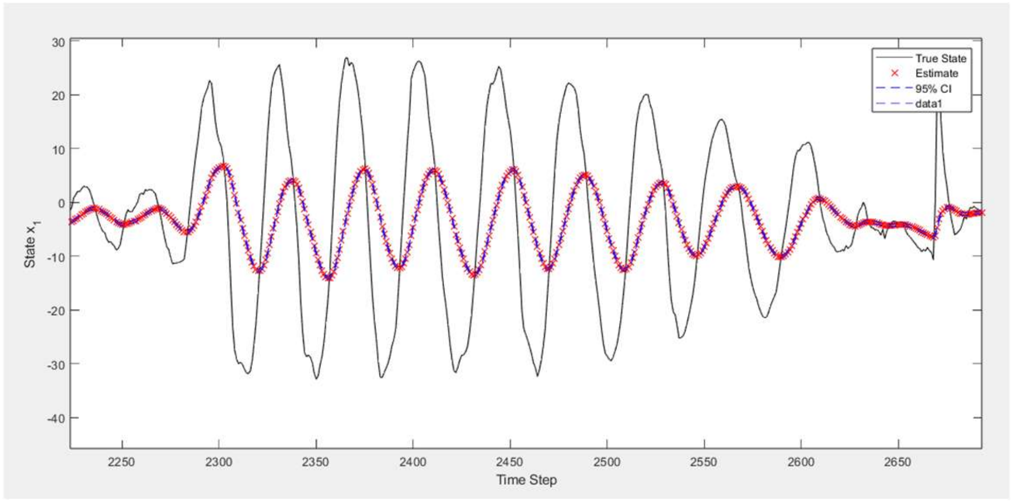


Figure 7: Pitch Angle Kalman Filter Shake in X-Direction

Overview

Kalman Filtering With Bias and Sensor Fusion

- Kalman filter equations:

$$\boldsymbol{x_k = Fx_{k-1} + Bu_k + w_k}$$

$$\boldsymbol{z_k = Hx_k + v_k}$$

$$\boldsymbol{x_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k}$$

$$\boldsymbol{F = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}}$$

$$\boldsymbol{B = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}}$$

$$\boldsymbol{H = [1 \quad 0]}$$

$$\boldsymbol{x_k = Fx_{k-1} + B\dot{\theta}_k + w_k}$$

$$\boldsymbol{w_k \sim N(0, Q_k)}$$

$$\boldsymbol{v_k \sim N(0, R)}$$

$$\boldsymbol{R = E[v_k \quad v_k^T] = var(v_k)}$$

- Where $\boldsymbol{x_k}$ is the angle at time $\boldsymbol{k}$ in degrees, $\boldsymbol{F}$ is the state transition model, $\boldsymbol{B}$ is the control-input model, $\boldsymbol{u_k}$ is the gyroscope measurement in degrees/second, $\boldsymbol{w_k}$ is the process noise, $\boldsymbol{z_k}$ is the measurement, $\boldsymbol{H}$ is the observation model, and $\boldsymbol{v_k}$ is the measurement noise.

- Kalman filter steps:
- Predict:

$$\boldsymbol{\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + B\dot{\theta}_k}$$

$$\boldsymbol{P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k}$$

- Update:

$$\boldsymbol{y_k = z_k - H\hat{x}_{k|k-1}}$$

$$\boldsymbol{S_k = HP_{k|k-1}H^T + R}$$

$$\boldsymbol{K_k = P_{k|k-1}H^T S_k^{-1}}$$

$$\boldsymbol{\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k}$$

$$\boldsymbol{P_{k|k} = (I - K_k H)P_{k|k-1}}$$

Kalman Filtering With Bias and Sensor Fusion

- The updated Kalman filter incorporates both accelerometer and gyroscope data, introducing a bias term through the Kalman gain, K
- This method applies to both pitch and roll angles and is designed to better handle long-term noise and drift
- While initial results show only marginal improvement, the bias and gyroscope data become increasingly beneficial over time, preventing long-term drift compared to accelerometer-only methods

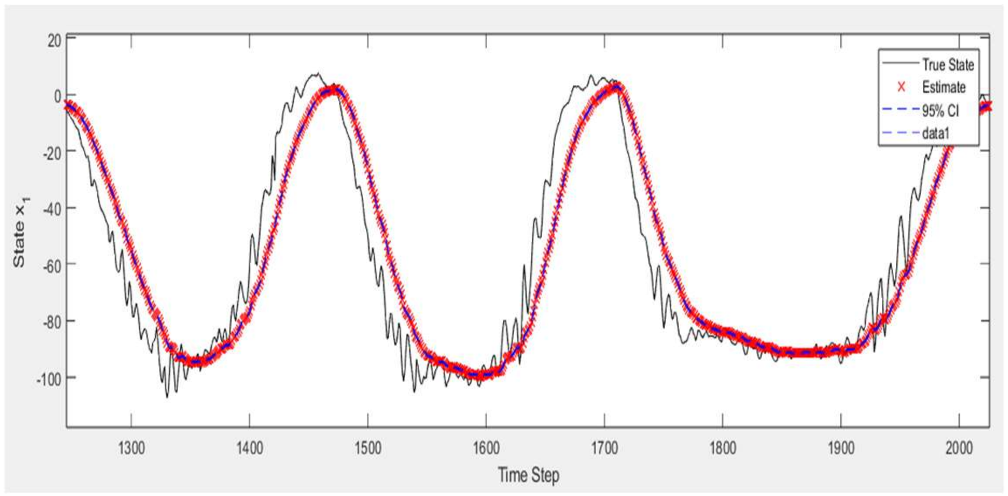


Figure 8: Pitch Angle Kalman Filter Rotation with Bias

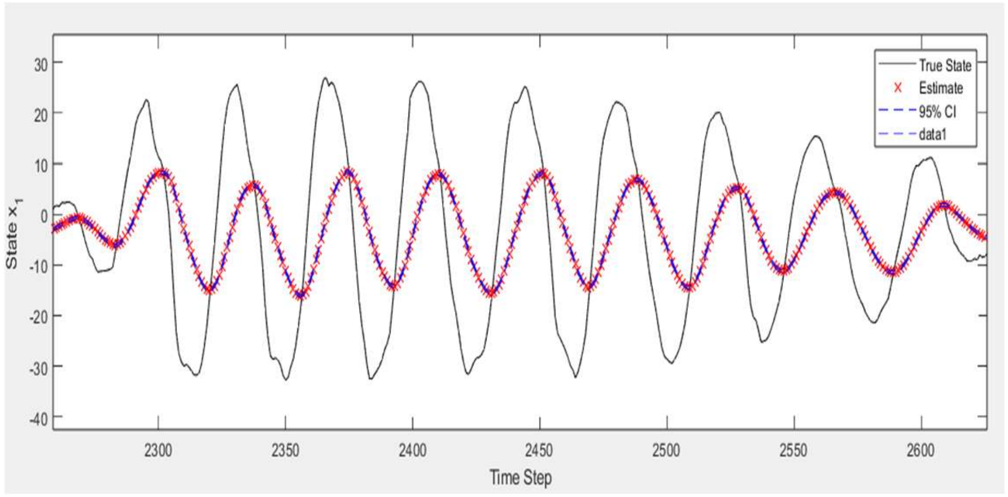


Figure 9: Pitch Angle Kalman Filter Shake in X-Direction with Bias

Overview

Live Arduino Kalman Filter Results

- The MATLAB-based Kalman filter was ported to C++ and run in real-time on the Arduino IDE
- Live pitch and roll angle estimates are displayed in the serial monitor and integrated with the Adafruit 3D model viewer
- The resulting motion is noticeably smoother, reflecting the effectiveness of the filter in real-time applications

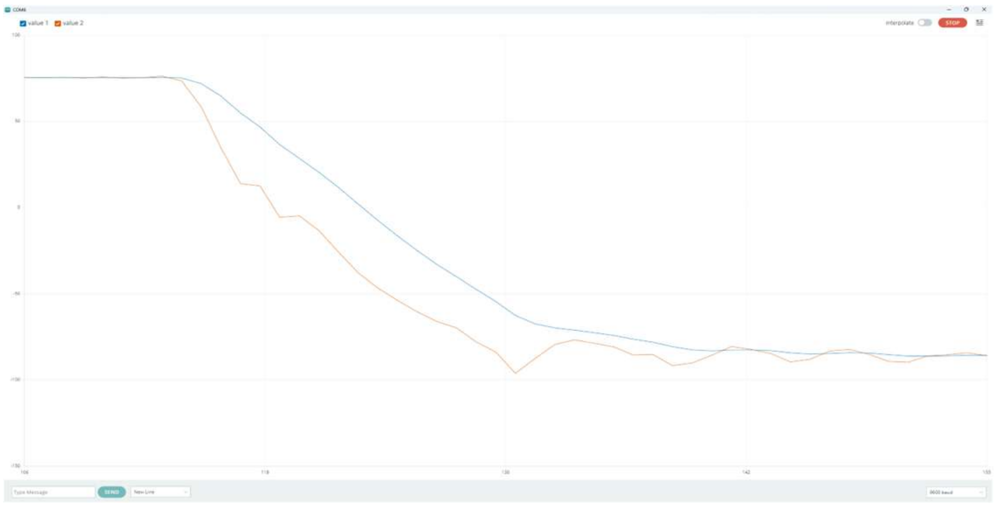


Figure 10: Live Pitch Angle Kalman Filtering with Bias

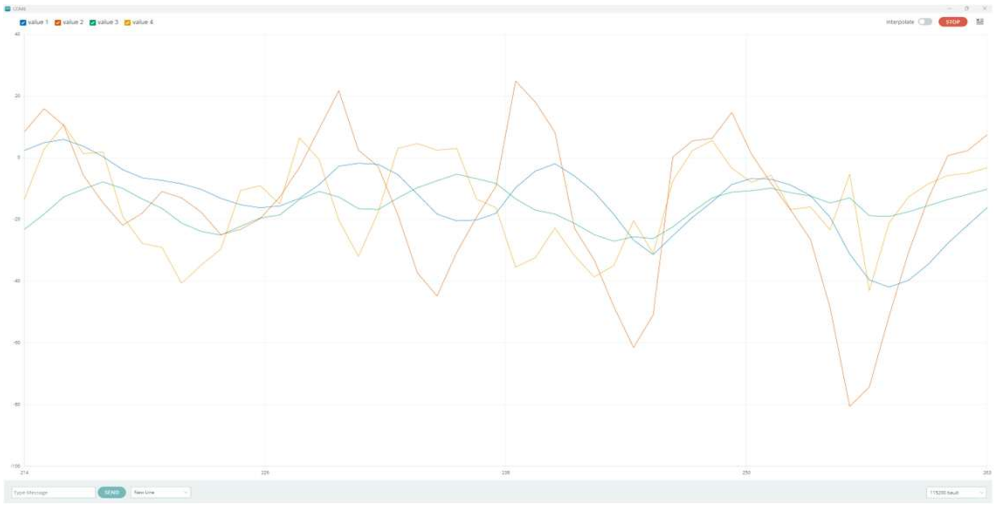
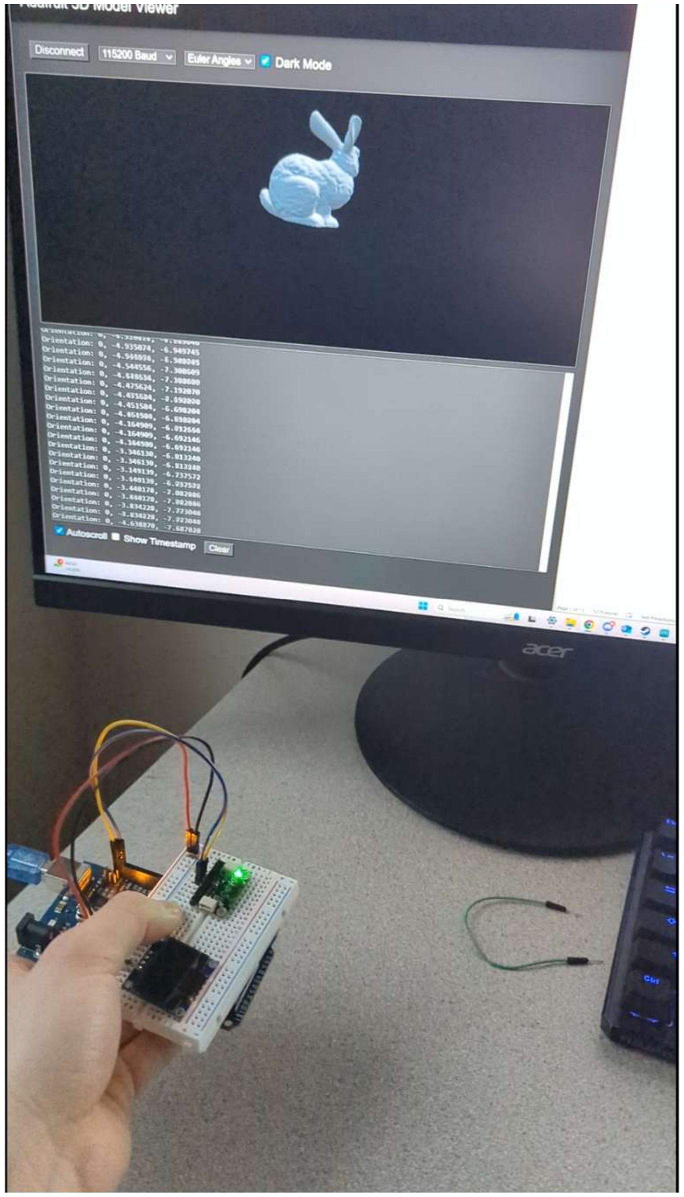


Figure 11: Live Pitch and Roll Angle Kalman Filtering with Bias

Live Arduino Kalman Filter Results

- No filter

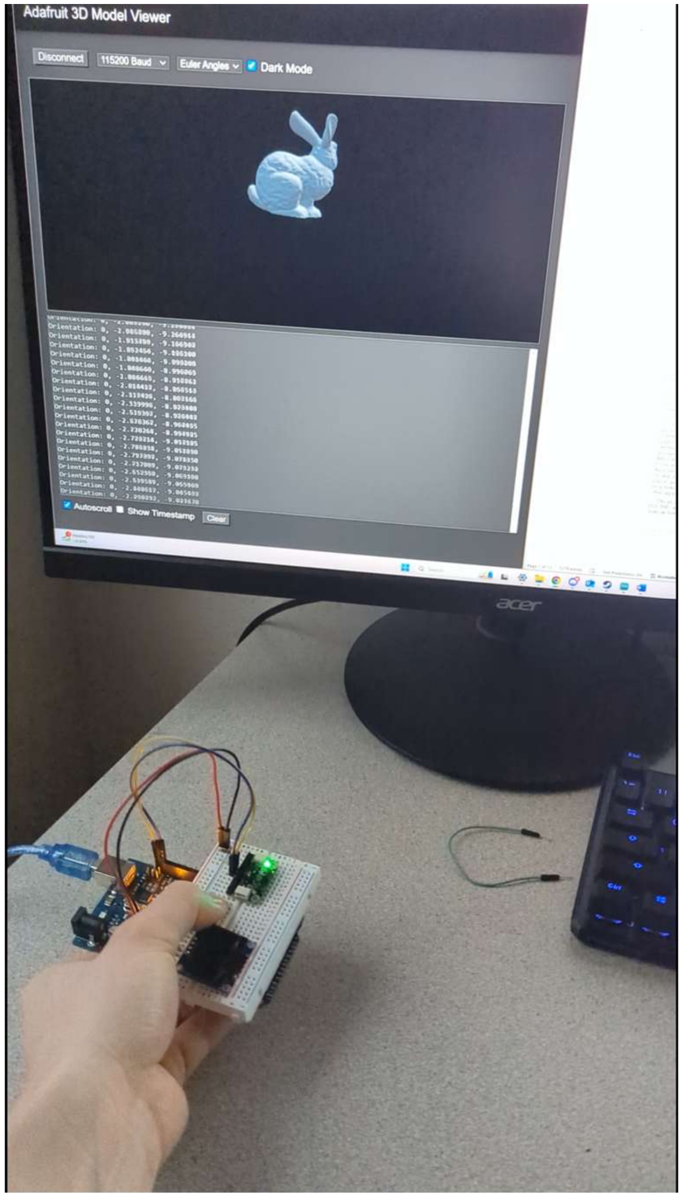


```
Orientation: 0, -4.935074, -6.949745
Orientation: 0, -4.940090, -6.980085
Orientation: 0, -4.544556, -7.300609
Orientation: 0, -4.488536, -7.300600
Orientation: 0, -4.475624, -7.192070
Orientation: 0, -4.475604, -6.692000
Orientation: 0, -4.451504, -6.690204
Orientation: 0, -4.054900, -6.690204
Orientation: 0, -4.164909, -6.092666
Orientation: 0, -4.164909, -6.692146
Orientation: 0, -4.164900, -6.692246
Orientation: 0, -3.346130, -6.813240
Orientation: 0, -3.346130, -6.813240
Orientation: 0, -3.149139, -6.737572
Orientation: 0, -3.440130, -6.237532
Orientation: 0, -3.440170, -7.082086
Orientation: 0, -3.440170, -7.082086
Orientation: 0, -3.834220, -7.773046
Orientation: 0, -3.834220, -7.773048
Orientation: 0, -4.638870, -7.687020
```

Figure 12: 3D Model Viewer w/ no filter

Live Arduino Kalman Filter Results

- With filter
- R = 0.03


Adafruit 3D Model Viewer

Disconnect | 115200 Baud | Euler Angles | ✓ Dark Mode

```
Orientation: 0, [illegible]
Orientation: 0, -2.085890, -9.260914
Orientation: 0, -1.955890, -9.166902
Orientation: 0, -1.892456, -9.086300
Orientation: 0, -1.803466, -9.090200
Orientation: 0, -1.808640, -8.996065
Orientation: 0, -1.806645, -8.958063
Orientation: 0, -2.014412, -8.856540
Orientation: 0, -2.117420, -8.807166
Orientation: 0, -2.339990, -8.823400
Orientation: 0, -2.519392, -8.926003
Orientation: 0, -2.678362, -8.968095
Orientation: 0, -2.738268, -8.994985
Orientation: 0, -2.720214, -9.017105
Orientation: 0, -2.786838, -9.051050
Orientation: 0, -2.797898, -9.078350
Orientation: 0, -2.717089, -9.079230
Orientation: 0, -2.652958, -9.069100
Orientation: 0, -2.539589, -9.055969
Orientation: 0, -2.408617, -9.045433
Orientation: 0, -2.298332, -9.023670
```

✓ Autoscroll ☐ Show Timestamp | Clear

Figure 13: 3D Model Viewer w/ filter R = 0.03

Live Arduino Kalman Filter Results

- With filter
- R = 0.3

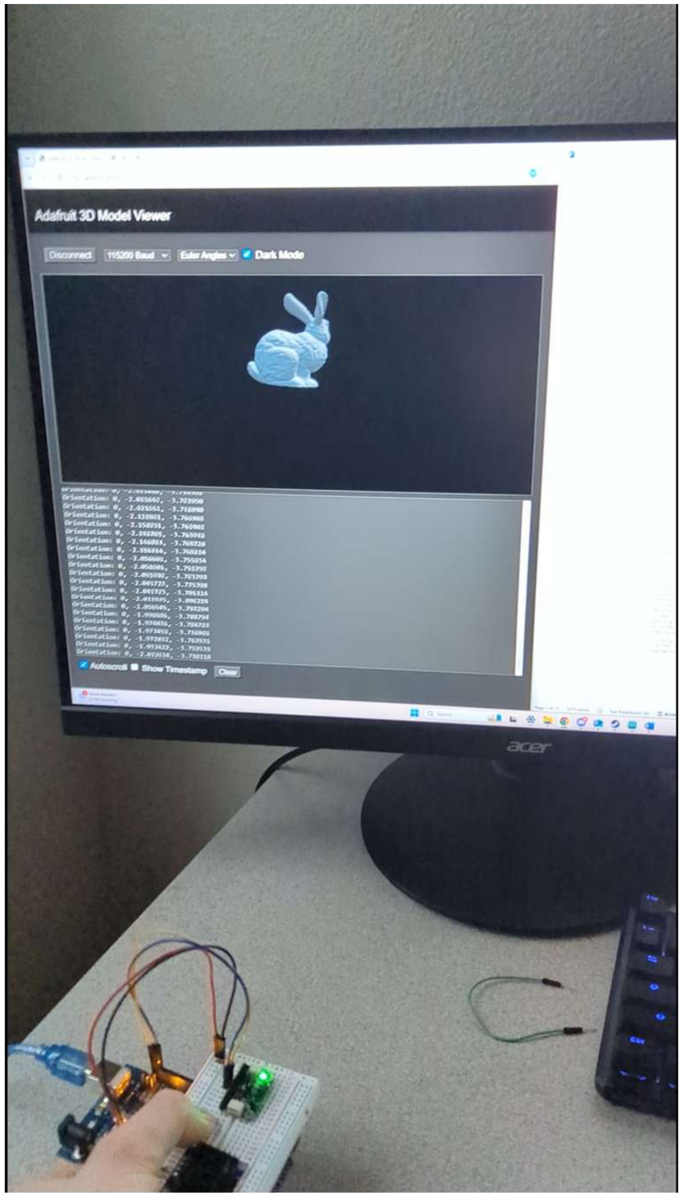
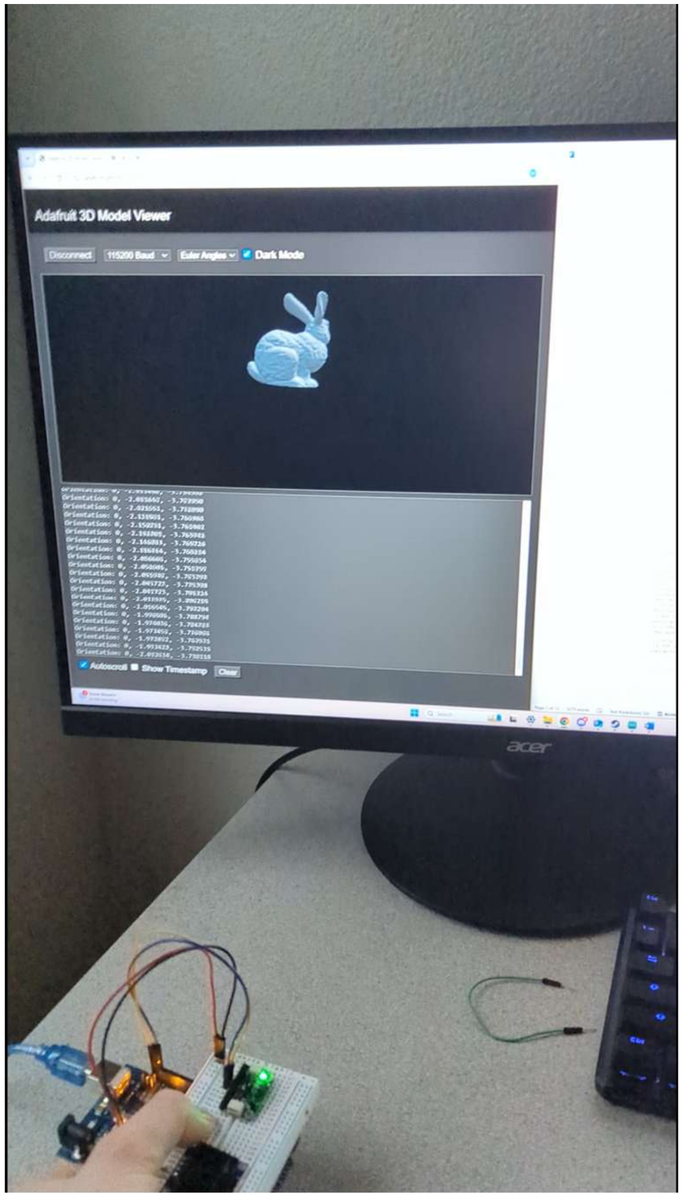


Figure 14: 3D Model Viewer w/ filter R = 0.3

Overview

Conclusion

- The Kalman filter implementation with a 6-DoF IMU effectively reduced noise and drift for more reliable attitude estimation
- By fusing accelerometer, gyroscope data, and bias correction, pitch and roll measurements became more accurate over time compared to the basic filter
- These results demonstrate the potential of Kalman filtering in improving low-cost IMU-based systems, with further optimization and hardware improvements likely to enhance real-time performance



```
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -1.[illegible], -3.[illegible]
Orientation: 0, -2.[illegible], -3.[illegible]
```

Figure 14: 3D Model Viewer w/ filter R = 0.3