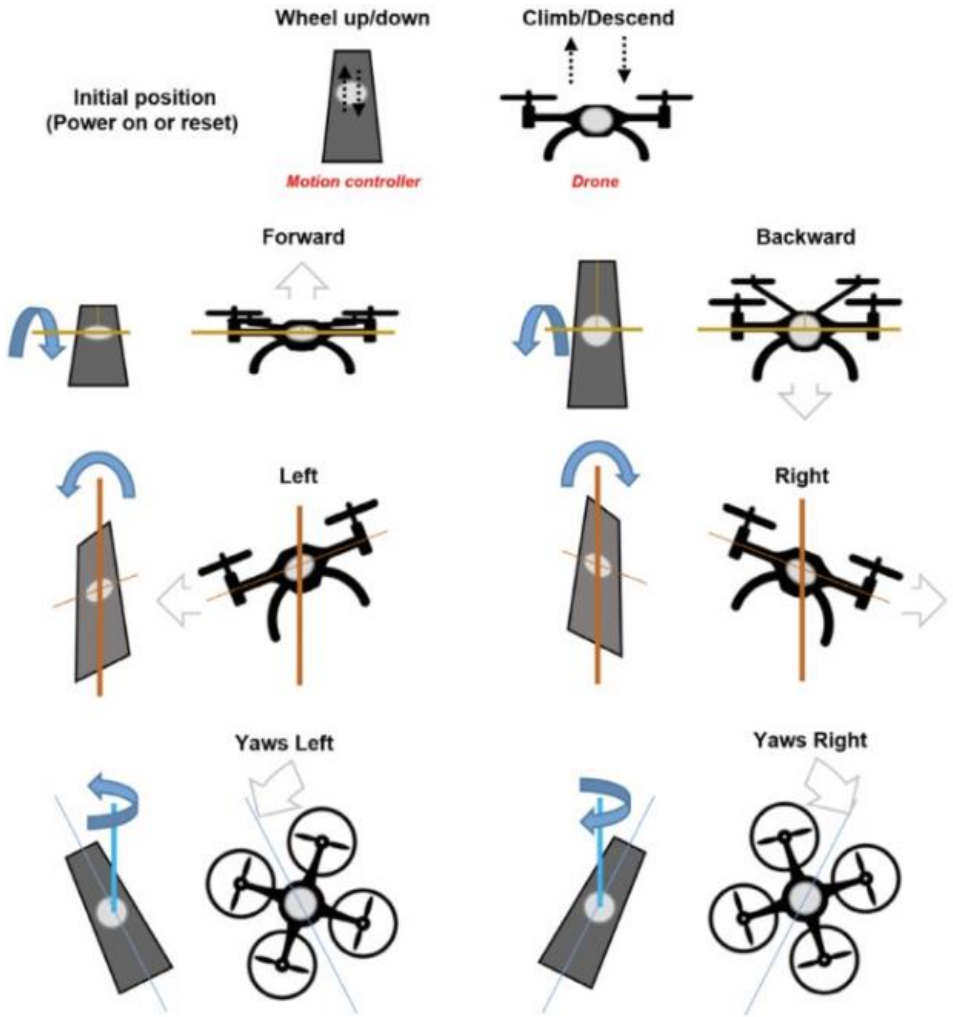
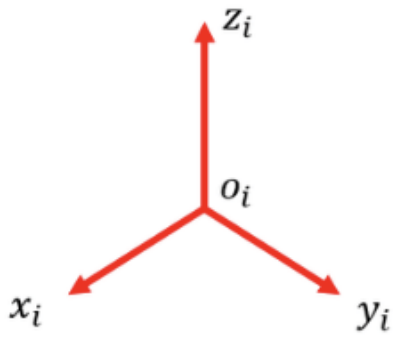
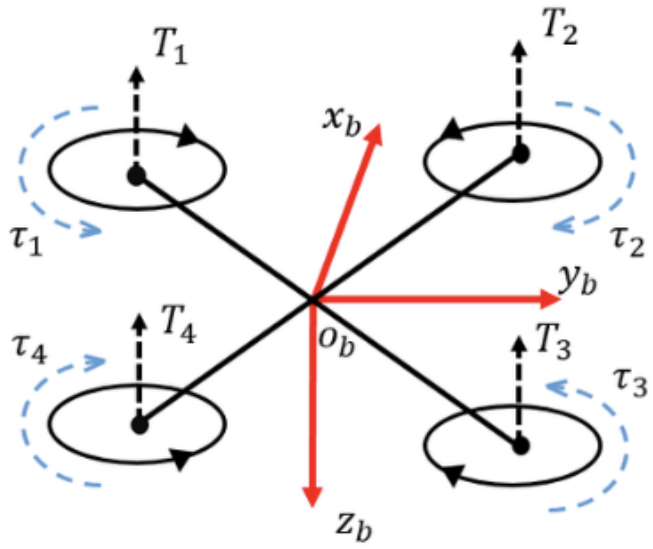
Mechatronics
Final Project Presentation

Farris Hamid and Ali Fakhar

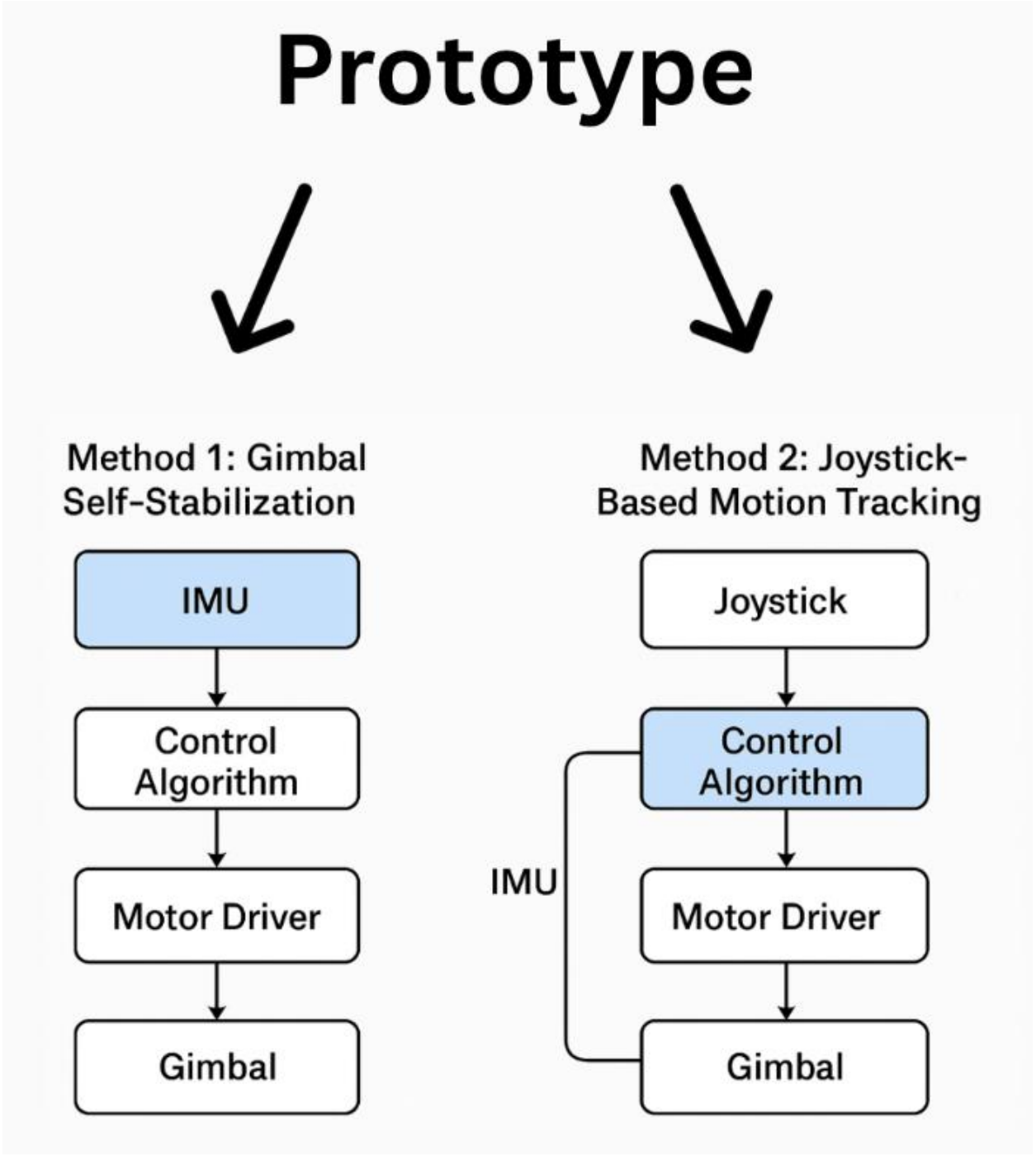
Motivation for the project -> Drone Control







Prototype



Does not use the PID controller (according to our midterm)

Joystick uses the IMU sensor for reading angles for the Gimbal.

Definitions

- Include Libraries
- Define Object MPU6050
- Initialize Gain Values for Roll, Pitch, Yaw
- Joystick Pins

```
#include <Wire.h>
#include <MPU6050.h>
#include <Servo.h>
#include "Kalman.h"

// --- IMU and Kalman Filter ---
MPU6050 mpu;
Kalman kalmanX;   // Roll filter
Kalman kalmanY;   // Pitch filter

// --- Servos ---
Servo servoRoll;
Servo servoPitch;
Servo servoYaw;

// --- Timing ---
unsigned long timerMicros;
double kalAngleX, kalAngleY;
double gyroXangle, gyroYangle, gyroZangle;
double compAngleX, compAngleY;

// --- Gains ---
#define ROLL_GAIN   1.5
#define PITCH_GAIN  1.5
#define YAW_GAIN    2.0
#define ALPHA       0.98   // complementary filter blend factor

// --- Joystick pins ---
// Joystick 1: pitch/roll
const int joy1RollPin  = A0;
const int joy1PitchPin = A1;
const int btnOverride1 = 4;   // active LOW

// Joystick 2: roll/pitch/yaw
const int joy2RollPin  = A2;
const int joy2PitchPin = A5;
const int joy2YawPin   = A3;  // NEW: yaw axis
const int btnOverride2 = 5;   // active LOW

// --- Helpers ---
static inline double rad2deg(double r) { return r * (180.0 / PI); }
```

Setup

- Initialize MPU6050
- Offsets
- Roll, Pitch, Yaw motions
- Accelerometer / Gyro calculations and pass through Kalman filter
- Complimentary
- Override buttons
- Start Timer for counting

```
43  void setup() {
44    Serial.begin(115200);
45    Wire.begin();
46
47
48    // Initialize MPU
49    mpu.initialize();
50    if (!mpu.testConnection()) {
51      Serial.println("MPU6050 connection failed");
52      while (1);
53    }
54    Serial.println("MPU6050 Connected");
55
56    // Calibration offsets
57    mpu.setXAccelOffset(-820);
58    mpu.setYAccelOffset(1633);
59    mpu.setZAccelOffset(1299);
60    mpu.setXGyroOffset(102);
61    mpu.setYGyroOffset(11);
62    mpu.setZGyroOffset(27);
63
64    // Initial sensor read
65    int16_t ax, ay, az, gx, gy, gz;
66    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
67
68    double accXangle = rad2deg(atan2((double)ay, (double)az));
69    double accYangle = rad2deg(atan2(-(double)ax, sqrt((double)ay * ay + (double)az * az)));
70
71    kalmanX.setAngle(accXangle);
72    kalmanY.setAngle(accYangle);
73
74    gyroXangle = accXangle;
75    gyroYangle = accYangle;
76    gyroZangle = 0.0;
77
78    compAngleX = accXangle;
79    compAngleY = accYangle;
80
81    // Attach servos
82    servoRoll.attach(9);
83    servoPitch.attach(8);
84    servoYaw.attach(7);
85
86    // Joystick buttons
87    pinMode(btnOverride1, INPUT_PULLUP);
88    pinMode(btnOverride2, INPUT_PULLUP);
89
90    timerMicros = micros();
91  }
```

Loop

- Get Accelerometer and Gyro Readings and pass it through the Kalman Filter
- In this case, the override of joystick is set to LOW (default run only Gimbal Stabilization)

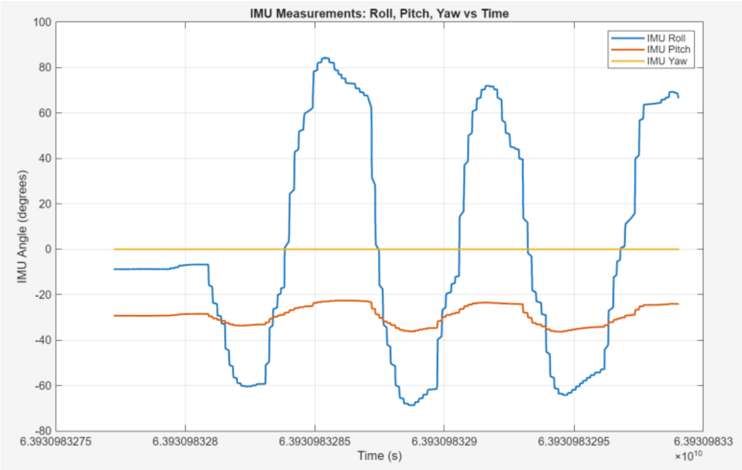
```
93  void loop() {
94    // --- Sensor read ---
95    int16_t ax, ay, az, gx, gy, gz;
96    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
97
98    unsigned long now = micros();
99    double dt = (double)(now - timerMicros) / 1e6;
100   timerMicros = now;
101   if (dt <= 0) dt = 1e-3;
102
103   double gyroXrate = (double)gx / 131.0;
104   double gyroYrate = (double)gy / 131.0;
105   double gyroZrate = (double)gz / 131.0;
106
107   double accXangle = rad2deg(atan2((double)ay, (double)az));
108   double accYangle = rad2deg(atan2(-(double)ax, sqrt((double)ay * ay + (double)az * az)));
109
110   // Integrate gyro
111   gyroXangle += gyroXrate * dt;
112   gyroYangle += gyroYrate * dt;
113   gyroZangle += gyroZrate * dt;
114
115   // Complementary filter
116   compAngleX = ALPHA * (compAngleX + gyroXrate * dt) + (1.0 - ALPHA) * accXangle;
117   compAngleY = ALPHA * (compAngleY + gyroYrate * dt) + (1.0 - ALPHA) * accYangle;
118
119   // Kalman fusion
120   kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, dt);
121   kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, dt);
122
123   // --- Override logic ---
124   bool override1 = (digitalRead(btnOverride1) == LOW);
125   bool override2 = (digitalRead(btnOverride2) == LOW);
126
127   int servoRollAngle;
128   int servoPitchAngle;
129   int servoYawAngle;
```

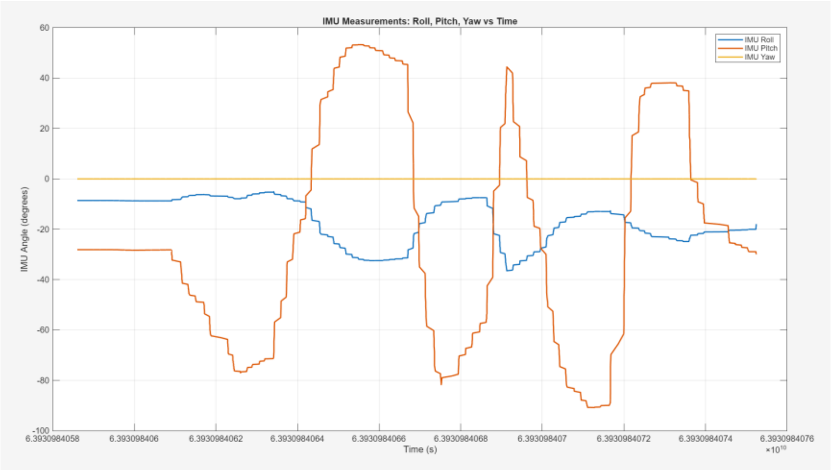
Loop (Contd.)

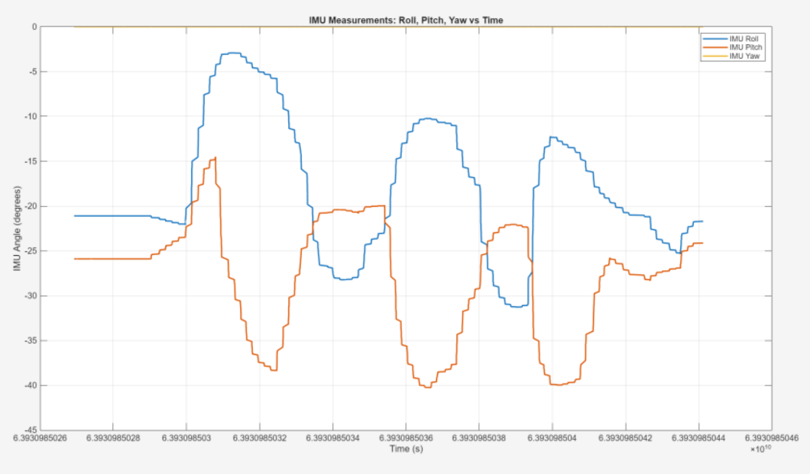
- In this case, if joystick is pressed, run the manual joystick to control roll and pitch and yaw.
- Else, map the gimbal angles to the servo motor to react opposite direction to keep platform fixed horizontal.

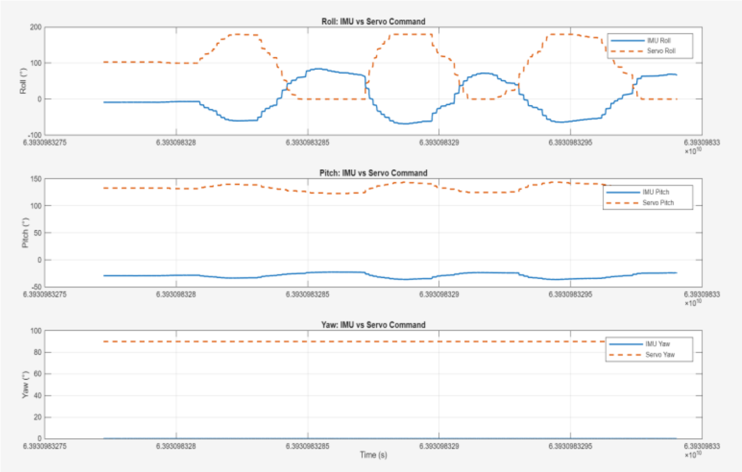
```
131   if (override1 || override2) {
132     int rollRaw  = override2 ? analogRead(joy2RollPin)  : analogRead(joy1RollPin);
133     int pitchRaw = override2 ? analogRead(joy2PitchPin) : analogRead(joy1PitchPin);
134
135     servoRollAngle  = map(rollRaw,  0, 1023, 0, 180);
136     servoPitchAngle = map(pitchRaw, 0, 1023, 0, 180);
137
138     if (override2) {
139       int yawRaw = analogRead(joy2YawPin);
140       servoYawAngle = map(yawRaw, 0, 1023, 0, 180);
141     } else {
142       servoYawAngle = map((long)(-gyroZangle * YAW_GAIN), -180, 180, 0, 180);
143     }
144
145     Serial.print("Manual Override ");
146     Serial.print(override2 ? "(Joy2)" : "(Joy1)");
147     Serial.print(" -> rollRaw: "); Serial.print(rollRaw);
148     Serial.print(" pitchRaw: "); Serial.print(pitchRaw);
149     if (override2) {
150       Serial.print(" yawRaw: "); Serial.print(analogRead(joy2YawPin));
151     }
152   } else {
153     // Automatic stabilization
154     servoRollAngle  = map((long)(-kalAngleX * ROLL_GAIN),  -90, 90, 0, 180);
155     servoPitchAngle = map((long)(-kalAngleY * PITCH_GAIN), -90, 90, 0, 180);
156     servoYawAngle   = map((long)(-gyroZangle * YAW_GAIN), -180, 180, 0, 180);
157   }
158
159   // --- Drive servos ---
160   servoRollAngle  = constrain(servoRollAngle, 0, 180);
161   servoPitchAngle = constrain(servoPitchAngle, 0, 180);
162   servoYawAngle   = constrain(servoYawAngle, 0, 180);
163
164   servoRoll.write(servoRollAngle);
165   servoPitch.write(servoPitchAngle);
166   servoYaw.write(servoYawAngle);
167
168   // --- Debug ---
169   Serial.print("\tKF Roll: ");  Serial.print(kalAngleX, 3);
170   Serial.print(" KF Pitch: ");  Serial.print(kalAngleY, 3);
171   Serial.print(" GyroYaw: ");   Serial.print(gyroZangle, 3);
172   Serial.print(" | ServoRoll: ");  Serial.print(servoRollAngle);
173   Serial.print(" ServoPitch: ");   Serial.print(servoPitchAngle);
174   Serial.print(" ServoYaw: ");     Serial.println(servoYawAngle);
175
176   delay(10); // ~100 Hz
177 }
```

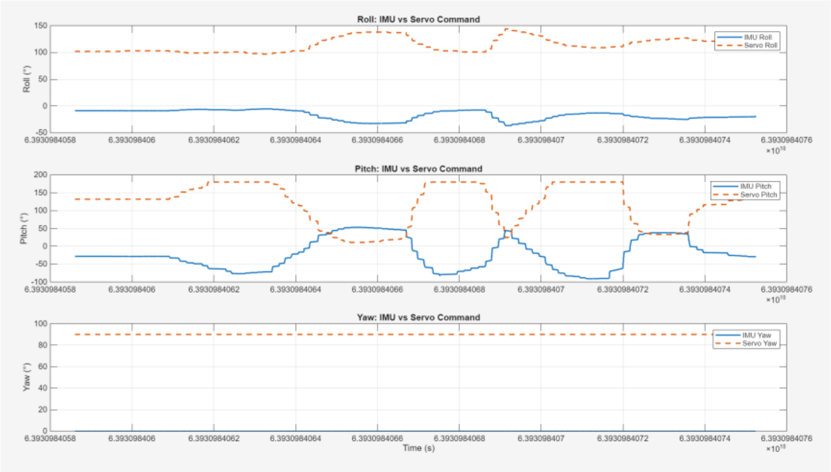
Gimbal IMU vs Servo Graph Readings

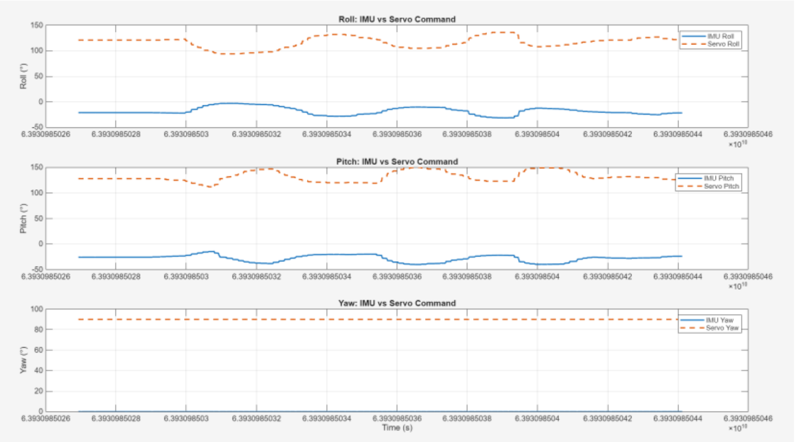




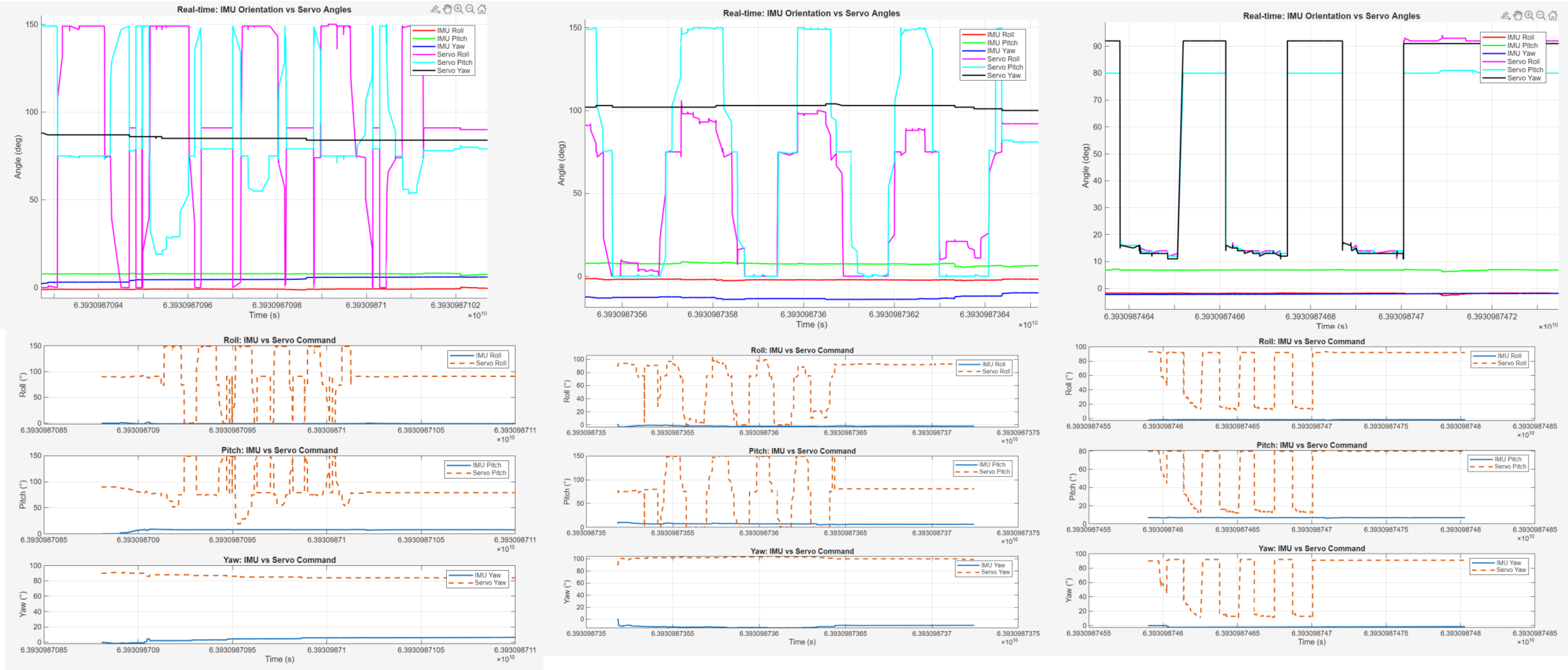








Gimbal IMU vs Joystick Graph Readings



Future Improvements

- MPU needs to have a magnetometer in order to detect motion in the yaw axis for the hand motion tracking.
- Servo not able to completely rotate in all three axis (roll, pitch, yaw).
  - Issue could arise from Gearbox conversion DAC
- The motor commands can be red using the cereal monitor for both IMU and joystick control.