Gaming Gun with Raspberry Pi and MPU Sensor

Mustafa Ali \*, Faizan Hussain\*, Murtaza Ahmed Khan\*

\*School of Electrical Engineering and Computer Science National University of Sciences and Technology

Islamabad, Pakistan

[mustafali.bee21seecs@seecs.edu.pk](mailto:mustafali.bee21seecs@seecs.edu.pk)  
[fhussain.bee21seecs@seecs.edu.pk](mailto:fhussain.bee21seecs@seecs.edu.pk)

[makhan.bee21seecs@student.nust.edu.pk](mailto:makhan.bee21seecs@student.nust.edu.pk)

***Abstract*— This project report outlines the development of an innovative Gaming Gun controller designed to enhance mobile gaming experiences. By utilizing a Raspberry Pi and an MPU sensor, the system detects the angle and movement of the gun and translates these physical movements into screen swipes using ADB commands. The Gaming Gun project aims to provide a more immersive and interactive gaming experience by allowing players to control in-game actions through the natural motion of the gun. The report covers the project's objectives, system design, hardware and software implementation, testing procedures, results, and challenges encountered. Additionally, it highlights potential future improvements and extensions to the system. The successful integration of sensor data with ADB commands demonstrates the feasibility of creating responsive and engaging physical controllers for mobile gaming applications.**

***Index Terms*—Gaming Gun controller, mobile gaming, Raspberry Pi, MPU sensor, screen swipes, ADB commands, immersive gaming experience, interactive gaming.**

I. Introduction

The Gaming Gun project aims to create an immersive gaming experience by integrating a physical gun controller with mobile devices. By utilizing a Raspberry Pi and an MPU sensor, the project enables users to control in-game actions through the physical movement of the gun, translating these movements into screen swipes using ADB commands.

II. METHODOLOGY

The methodology for the Gaming Gun project is divided into three main phases: hardware setup, software development, and testing. Each phase involves specific tasks and procedures to ensure the successful implementation of the gaming gun controller.

**Block Diagram**



## We captured noisy data from 9DOF IMU sensors (Accelerometer + Gyroscope + Magnetometer) (MPU\_9250)

## After that we sent the data to our raspberry pi which was operating on linux. Using CLI we configured our system to start our python scripts as soon as the RPI powered up.

## We then used various aspects of digital signal processing to reduce the noise of our input signal.

## After we obtained clean results, we converted those results to represent the number of pixels of the display.

## Those pixel values were then used to simulate virtual swipes on the mobile phone using ADB (android debug bridge)

## Buttons were working simultaneously with the same process above except for the fact that the do not need signal processing.

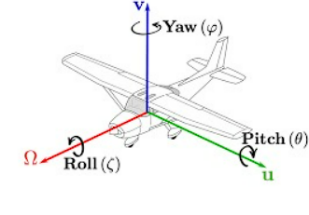
## 1. Hardware Setup

### 1.1 Raspberry Pi Configuration

* **Selection:** A Raspberry Pi Model 4 was chosen for its processing power and GPIO capabilities.
* **Setup:** Installed Raspbian OS on the Raspberry Pi.
* **Libraries:** Installed necessary Python libraries, including ‘os’ for executing system commands.

### 1.2 MPU Sensor Integration

* **Sensor Selection:** The MPU-6050 sensor was selected for its six-axis motion tracking capabilities (three-axis gyroscope and three-axis accelerometer).



**Yaw, Pitch, and Roll**

**Yaw:** Yaw is the rotation around the vertical axis of the gun. It measures the horizontal left or right movement of the gun. For example, if you were to turn your head from side to side, you would be yawing.

**Pitch:** Pitch is the rotation around the side-to-side axis of the gun. It measures the up or down movement of the gun. Imagine nodding your head up and down—that's pitch.

**Roll:** Roll is the rotation around the front-to-back axis of the gun. It measures the tilting left or right movement of the gun. Picture tilting your head sideways—that's roll.

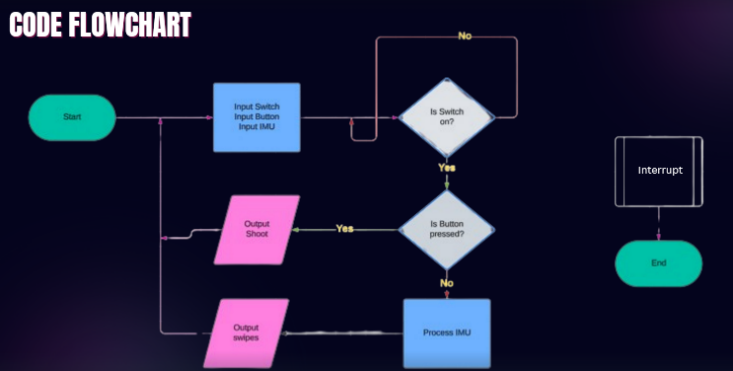
These three terms describe the orientation of the gun in three-dimensional space, allowing us to understand how it's positioned and moving relative to its starting point.

* **Connection:** Wired the MPU-6050 sensor to the Raspberry Pi’s GPIO pins using I2C communication protocol.
* VCC to 3.3V
* GND to GND
* SDA to GPIO2 (SDA)
* SCL to GPIO3 (SCL)
* **Initialization:** Developed a Python script to initialize and configure the MPU-6050 sensor.

### 1.3 Gaming Gun Assembly

* **Housing:** Integrated the MPU sensor and Raspberry Pi into the gaming gun, ensuring secure connections and proper alignment.
* **Power Supply:** Connected a portable power supply to the Raspberry Pi to allow for mobility during gameplay.

## 2. Software Development



### 2.1 Sensor Data Processing

* **Data Acquisition**: Developed a Python script to read raw data from the MPU sensor.
* **Calibration:** Implemented calibration routines to ensure accurate angle measurements, including zeroing out any constant bias in sensor readings.
* **Angle Calculation:** Applied sensor fusion algorithms to calculate the pitch and roll angles from the accelerometer and gyroscope data.
* **Average Filter:** Implemented an average filter using the 10 previous values to reduce minor fluctuations in the sensor data.
* **Relative Change:** Used relative changes in pitch and yaw rather than absolute angles to perform digital signal processing (DSP).
* **Band Stop Filter:** Added a band stop filter to eliminate certain values that were identified as noise through experimentation.
* **Sampling Rate Correlation:** Correlated the sampling rate with frequency values obtained. High sampling rates resulted in smaller values, while low sampling rates resulted in greater values. Found the optimal match between the two for increased precision.

### 2.2 ADB Command Execution

* **ADB Setup:** Installed ADB tools on the Raspberry Pi and established a connection with the target Android device.
* **Swipe Command Function:** Created a Python function to send swipe commands to the Android device using the ‘os’ system method.
* **Coordinate Mapping:** Developed a mapping function to convert the calculated angles into screen coordinates for the swipe commands.

### 2.3 Integration

* **Main Loop:** Combined sensor data processing and ADB command execution into a main loop that continuously reads sensor data, calculates angles, and sends corresponding swipe commands.
* **Optimization:** Optimized the loop for minimal latency, ensuring real-time responsiveness.

III. Results and Discussion

### 3.1 Calibration

* **Procedure:** Calibrated the MPU sensor by holding the gun in a known orientation and adjusting the raw sensor data to match expected values.
* **Validation:** Verified calibration accuracy by comparing calculated angles to known reference angles.

### 3.2 Functionality Test

* **Movement Translation:** Tested the system by moving the gaming gun in different directions and observing the corresponding screen swipes on the mobile device.
* **Consistency:** Ensured consistent and accurate translation of physical movements to screen actions.

### 3.3 Performance Test

* **Latency:** Measured the time delay between physical movement and corresponding screen swipe to ensure real-time responsiveness.
* **Accuracy:** Assessed the precision of angle detection and swipe execution through repeated tests in various scenarios.

IV. FUTure enhancement

**Multi-Threading:** Used multi-threading to handle different tasks concurrently. Buttons for additional controls will be implemented on a separate thread to prevent movement delays and ensure seamless operation.

**AI Models:** Plan to incorporate AI models to increase accuracy and provide aim assist for a better gaming experience.

V. Conclusion

The Gaming Gun project successfully developed a novel and immersive gaming controller that enhances mobile gaming experiences through natural motion interaction. By leveraging a Raspberry Pi and an MPU sensor, the project achieved real-time translation of physical gun movements into screen swipes on a mobile device using ADB commands. This integration allowed for a more engaging and interactive gaming experience, showcasing the potential of combining hardware and software to create innovative gaming peripherals.