IoT Project Report

Project Title: **Air Mouse**

Students: Pang Panhakuntheakreaksmey, Lou Julie

Course: IoT

Instructor: Seng Theara

Date: 12/10/2025

Abstract

This project develops an IoT-based Air Mouse system using an ESP32 microcontroller and an MPU6050 gyroscope/accelerometer. The system functions as a Bluetooth mouse, translating hand motion into cursor movement. Sensor data is also transmitted to an InfluxDB time-series database and visualized using Grafana. Additionally, a Telegram bot sends a startup notification whenever the device is powered on. The final system demonstrates successful integration of wireless control, IoT data monitoring, and cloud-based visualization.

Introduction

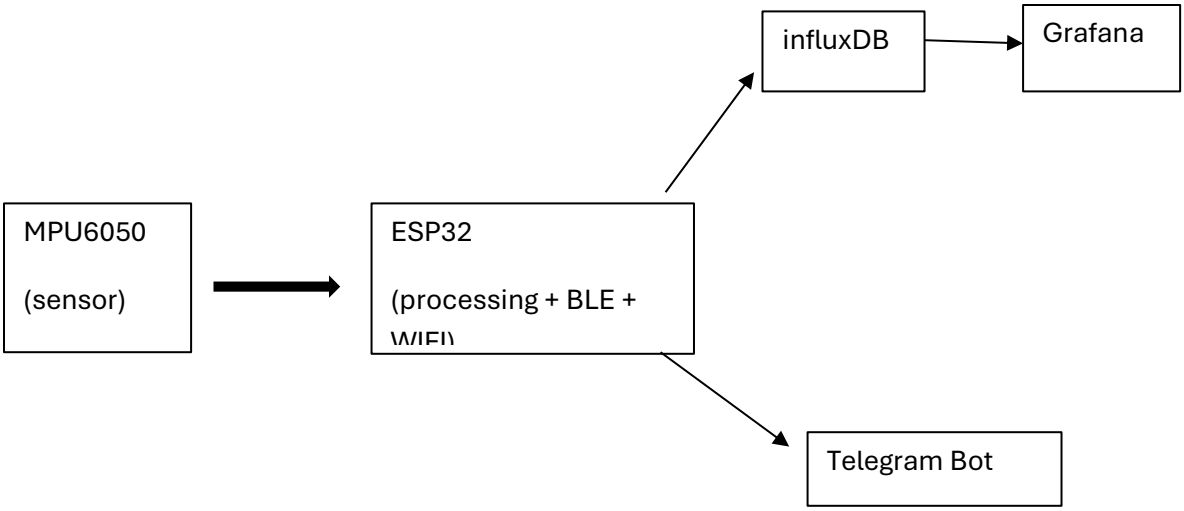
The Air Mouse project aims to eliminate limitations of traditional mice, such as cable constraints and surface dependency. By utilizing motion sensors and wireless connectivity, users can control a cursor through hand gestures. The project objectives include:

- Implementing a functional Bluetooth Air Mouse using ESP32.
- Capturing motion data using the MPU6050 sensor.
- Transmitting sensor data to an InfluxDB server.
- Visualizing real-time data with Grafana.
- Sending system startup notifications through a Telegram Bot.

Challenges include Wi-Fi reliability, BLE interference, sensor calibration, and timing limitations of the ESP32.

System Overview

The overall architecture of the IoT-based Air Mouse system is shown below:



System Workflow:

1. MPU6050 detects acceleration and gyroscope data.
2. ESP32 processes the data and converts it into cursor movements using BLE Mouse.
3. ESP32 sends sensor data to InfluxDB via HTTP POST.
4. Grafana reads InfluxDB data and displays real-time graphs.
5. Telegram Bot receives a startup notification from ESP32 through HTTPS.

Technologies Used:

- ESP32 Development Board
- MPU6050 Sensor
- BLE Mouse Library
- InfluxDB
- Grafana Dashboard
- Telegram Bot API

Hardware Components

1. ESP32 Microcontroller:

Handles Bluetooth mouse functionality, Wi-Fi communication, and sensor data processing.

2. MPU6050 Gyroscope + Accelerometer:

- Provides rotation (gyro) and acceleration readings used to control cursor movement.
- Wiring:
  - VCC → VCC
  - GND → GND
  - SDA → GPIO 21
  - SCL → GPIO 22

3. Buttons:

- Button1 (Left Click) → GPIO 18 → GND

- Button2 (Right Click) → GPIO 19 → GND

4. Breadboard & Jumper Wires:

- Used for assembling the circuit.

Software Architecture

System Flow Overview:

The air mouse system uses two programming languages:

- C/C++ (Arduino): Runs on ESP32 microcontroller
- Python: Runs on computer for data logging

System Data Flow:

1. ESP32 reads gyroscope motion
2. Convert motion to mouse movements via bluetooth
3. Logs activity data through USB serial
4. Python script captures the logs
5. Stores data in database and send alerts

Flowchart:

graph TD

A[ESP32 Starts] --> B{Bluetooth Connected?}

B -->|No| B

B -->|Yes| C[Wake Up Sensor]

C --> D[Read Gyroscope]

D --> E[Move Mouse via BLE]

E --> F{Button Pressed?}

F -->|Yes| G[Send Click]

F -->|No| H[Print to Serial]

G --> H

H --> I[Python Reads Serial]

I --> J[Save to InfluxDB]

J --> D

Component Breakdown:

Part 1: ESP32 Firmware (C/Arduino): Controls the hardware and mouse

// Initialize sensor mpu.begin();

// Read motion data sensors_event_t a, g, temp; mpu.getEvent(&a, &g, &temp);

// Move mouse based on gyroscope bleMouse.move(g.gyro.z * -SPEED, g.gyro.x * -SPEED);

// Handle button clicks if (!digitalRead(LEFTBUTTON)) { bleMouse.click(MOUSE_LEFT); }

Explanation:

- Reads X, Y, Z rotation from gyroscope
- Multiplies by speed constant (SPEED = 10)
- Sends movement to computer via Bluetooth
- Checks if buttons are pressed and sends clicks

Part2: Python Data Logger: Capture serial output and stores data

Read data from ESP32

line = ser.readline().decode("utf-8")

Parse motion data

if "Motion -> dx:" in line: dx = float(...) # Extract numbers dy = float(...) # Format for database data = f"motion,device=esp32 dx={dx},dy={dy}"

Parse clicks

if "Left click" in line: data = "click,button=left value=1i"

Send to InfluxDB

requests.post(INFLUX_URL, data=data)

Explanation:

- Opens serial port (USB connection to ESP32)
- Reads text lines like "Motion -> dx: 0.195, dy: 0.226"
- Extracts the numbers using regex
- Formats data in InfluxDB format

- Sends via HTTP POST request

Part 3: Telegram Bridge: Sends notifications

```
@app.route("/telegram", methods=["POST"])
def telegram_bridge():
    msg = request.args.get("msg")

    url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendMessage"
    payload = {"chat_id": CHAT_ID, "text": msg}

    requests.post(url, data=payload)
```

Explanation:

- Creates a web endpoint
- Receives messages via HTTP
- Forwards to Telegram API

Simple Pseudocode:

ESP32 Logic

```
START
  Connect Bluetooth
  Initialize sensor

  REPEAT forever:
    Read gyroscope (x, y, z values)
    Calculate: dx = z * -10, dy = x * -10
    Send mouse movement (dx, dy)

    IF left button pressed:
      Send left click

    IF right button pressed:
      Send right click
  END REPEAT
END
```

Python Logic

```
START
  Open serial port
```

```
  REPEAT forever:
    Read one line from serial

    IF line contains "Motion":
      Extract dx and dy numbers
      Format as database entry
      Send to InfluxDB

    IF line contains "click":
      Format click data
      Send to InfluxDB
  END REPEAT
END
```

Data Formats:

Serial Output (from ESP32)

```
Motion -> dx: 0.195 , dy: 0.226
Left click
Right click
```

Database Format (InfluxDB)

```
motion,device=esp32 dx=0.195,dy=0.226
click,button=left value=1i
click,button=right value=1i
```

Error Handling:

Problem	Solution
Sensor not found	Stop program, show error
Bluetooth disconnected	Wait until reconnected
Serial read fails	Skip that line, continue
Database timeout	Log error, keep running

Strategy: Critical hardware errors stop the program. Data transmission errors are logged but don't crash the system.

Communication Summary:

- **BLE (Bluetooth Low Energy)**: Mouse control

- **Serial UART**: 115200 baud for logging
- **HTTP POST**: Database and Telegram
- **InfluxDB Protocol**: Time-series data format

Cloud / Dashboard Setup

InfluxDB:

- Stores time-series sensor measurements (accelX, gyroY, etc.).

Grafana Dashboard:

- Connected to InfluxDB as a data source.
- Displays live graphs of accelerometer and gyroscope data.

Telegram Bot:

- Configured using BotFather.
- ESP32 sends HTTPS messages to notify system startup.

Results and Testing

Test Summary:

- Bluetooth mouse movement was smooth and responsive.
- Sensor data uploaded successfully to InfluxDB.
- Grafana displayed real-time motion graphs.
- Telegram Bot delivered device startup messages correctly.

Limitations:

- Wi-Fi instability caused occasional data loss.
- MPU6050 noise required filtering.
- BLE movement lagged slightly during heavy Wi-Fi activity.

Discussion

The Air Mouse system met all intended objectives. Sensor accuracy, wireless communication, and cloud data visualization worked cohesively. The team learned the importance of handling ESP32 multitasking, calibrating motion sensors, and deploying cloud-based IoT dashboards.
Possible improvements include smoothing algorithms (Kalman Filter), better housing design, enhanced BLE performance, and gesture recognition.

Conclusion

The IoT-based Air Mouse successfully integrates motion sensing, BLE control, cloud monitoring, and user notifications. The system demonstrates practical IoT applications in human-computer interaction and serves as a foundation for future improvements.

Appendix

Full code, wiring diagrams, dashboard exports, and additional figures can be placed here.