

## **Project Design: The Glove Mouse**

Jinhan Yang

Reese Dow

Professor Karen Craigs

TPJ 655

9 April 2025

## Table of Content

Table of Content .....	1
Introduction .....	4
Abstract .....	5
Background research and Development .....	6
Eye-Tracking Mouse .....	6
Vertical Ergonomic Mouses .....	7
Handheld Trackball Mouse .....	8
System Functional Features and Specifications .....	9
Key Functional Features: .....	9
The early design and planing for inputs and outputs .....	10
Input Devices: .....	12
Output Devices: .....	12
Initial Circuit Design: .....	13
Limitations of the Early Design .....	15
Initial Design Improvements(As Implemented in the Final Project) .....	16
Expanded Input and Output Interfaces (Implemented): .....	16
Clear Functional Separation Between Joystick and Gyroscope (Implemented): ..	16
Upgraded Power Management System (Implemented): .....	17
Added Overheat Protection Mechanism (Implemented): .....	17
Improved Switch Design to Prevent Misoperation (Implemented): .....	17
Logic Design .....	18
System block diagram: .....	18
Code Structure and Library Imports .....	19
Core Libraries : .....	19

Theory of Operation .....	20
1. Gyroscope-Based Cursor Control.....	20
2. Rotary Encoder & Speed Gearing.....	21
3. T9 Keypad Input Logic.....	22
4.Primary Button Mode Switching.....	23
Product Operating Instructions .....	24
Maintenance Requirements .....	26
Code Architecture Highlights: .....	27
Hardware Initialization: .....	27
Safety Features: .....	27
Bluetooth HID Operation: .....	28
Once connected, the glove emulates mouse movement and button clicks over BLE. ....	28
Main Loop Execution: The control_mouse() function continuously reads sensor data, processes input events, checks for overheat conditions, and sends appropriate HID reports for mouse control. ....	28
Integrated Debugging and Monitoring: .....	28
Putty running examples: .....	28
Circuit Design .....	29
Central Microcontroller (ESP32-WROOM-32E): .....	29
Input Devices: .....	29
Sensor Modules: .....	30
Output Devices: .....	30
Power Distribution: .....	30
Routing .....	32

3D Printing Design .....	36
Component Breakdown: .....	36
Ergonomic Considerations: .....	37
Material and Manufacturing: .....	37
Integration with Glove: .....	38
Final Bill of Materials (BOM) .....	39
PCB Assembly .....	40
Product Overlook .....	41
Technical Limitations and Challenges Highlights .....	42
Continuous Movement Tracking: .....	42
Logical Cursor Tracking at Different Hand Angles: .....	42
T9 Keypad Integration: .....	43
Current Technical Limitations and Challenges .....	43
Optimal Finger Support: .....	43
Speed and Precision: .....	43
Prototype Hardware Resource Constraints: .....	44
Long-Term Wearability: .....	44
Future Improvements and Ideal Model .....	45
Semi-Open Design: .....	45
Ultra-Lightweight Self-Supporting Structure: .....	45
AI-Based Gesture Learning: .....	46
Conclusion .....	46
References .....	48
Appendices .....	49

## Introduction

The widespread use of computers today brings a massive market for peripheral devices, and among these, the mouse—arguably the most essential—has several limitations. Iconic mice, relying on laser grid sensors to track hand movement and reflect it as cursor motion, require a flat surface for smooth operation. Additionally, due to wrist-based use and finger mobility limitations, the number of customizable input keys is restricted. Even though some mice offer extra programmable buttons, their market share remains relatively low (Pramod, 2024).

In response to these issues, this project introduces a wearable glove-shaped computer mouse that enables cursor control via gestures (or thumb motion) and text input through a mini keyboard on the palm. This “glove mouse” design removes dependence on flat surfaces, enhancing flexibility and user experience.

The design objectives are divided into the following:

**To improve cursor control:** Using a gyroscope or joystick for precise movement, the glove mouse allows usage in any posture, freeing users from the constraints of desk-based operation.

**To improve clicking flexibility:** Trigger-from-inside programmable keys enable faster and more diverse clicking actions, reducing reliance on just three fingers (index, thumb, and middle).

**To improve typing:** The palm-mounted mini keyboard allows input by tapping with the other hand, eliminating the need for a desktop keyboard, though compatibility remains if desired.

## **Abstract**

This project presents the design and development of a wearable glove mouse, an innovative alternative to traditional desktop pointing devices. Motivated by the limitations of conventional mice—such as reliance on flat surfaces, limited finger access, and ergonomic discomfort—the glove mouse offers enhanced flexibility, mobility, and multifunctional input capabilities through gesture tracking and compact hardware integration.

The system is built around an ESP32 microcontroller and features various input modules including a gyroscope, keypad, and rotary encoder, allowing for gesture-based cursor control and T9-style text input. Outputs such as an OLED display, buzzer, RGB LED, and vibration motor provide immediate user feedback. The device operates wirelessly via BLE HID protocol, enabling seamless interaction with a PC.

Initial prototypes revealed issues related to redundant controls, limited power, and unreliable switch mechanisms. These challenges were addressed through refined hardware layout, upgraded circuit design, the introduction of MOSFET-controlled power management, and dedicated buttons for improved control reliability. Overheat protection logic was also implemented to ensure safe operation during extended use.

Mechanical components, including fingertip-mounted tactile button supports, were 3D printed to enhance user comfort and responsiveness. While the current prototype demonstrates stable performance and intuitive control, future improvements include AI-based gesture learning, optimized sensor modules, and breathable glove materials for extended wearability.

The glove mouse project successfully integrates hardware, firmware, and mechanical design into a functional prototype that reimagines user interaction. It lays the groundwork for next-generation wearable input systems applicable in both consumer and assistive technology domains.

### **Background research and Development**

As a complement to current computer peripherals, it is essential to explore similar research and innovative mouse designs. This study discusses three different types of existing mouse improvements and compares each of them to further analyze the advantages, disadvantages, and potential user profiles of this wearable glove mouse design.

#### **Eye-Tracking Mouse**

Gill and Ramaneeek explain that traditional computer mice are not friendly to individuals with limited hand mobility. In response, one major advancement has been the inclusion of eye-tracking technology to assist disadvantaged users. The basic concept of this type of mouse is to replace the laser grid scanner and traditional buttons with eye-tracking technology. The system tracks eye movement to control the cursor and eye blinks to register clicks (Gill and Ramaneeek).

This design may seem similar to the wearable glove mouse since it also frees users from desktop or seated operation. However, the eye-tracking mouse is primarily designed for individuals with disabilities, such as elderly individuals with Parkinson's disease or those unable to use their hands. As a result, its main focus is on

accessibility rather than improving the overall mouse experience, making its target user group distinct from that of the wearable glove mouse.

Other limitations of eye-tracking mice include weight and dexterity. Both helmet-mounted and glasses-based eye-tracking modules are significantly heavier than a traditional mouse, which can cause discomfort during prolonged use. In terms of dexterity, the main issue lies in the clicking mechanism. Professional gamers can perform up to 700 actions per minute (APM)—equivalent to clicking more than 11 times per second—whereas eye blinking cannot even come close to this rate. This inherently limits eye-tracking mice to slower-paced, casual use.

### **Vertical Ergonomic Mouses**

Another approach to improving the traditional mouse is the Vertical Ergonomic Mouse. This design aims to reduce carpal tunnel pressure for users suffering from carpal tunnel syndrome, which has gained widespread popularity in recent years. However, some research suggests that this goal has largely remained unfulfilled (Schmid et al.).

As an alternative to the conventional mouse, the vertical ergonomic design still requires the user to operate it on a desk. While technologies like control yokes and laser grid scanners are well-established, the mechanical nature of a control yoke inherently limits cursor movement speed compared to a traditional mouse. Once again, this ergonomic vertical design sacrifices sensitivity, making it less responsive than a standard mouse.

Moreover, as highlighted in research studies, operating a control yoke often demands greater force than moving a traditional mouse. Consequently, in comparative studies,



the claimed reduction in carpal tunnel pressure for vertical ergonomic mice has not been observed as significantly different from traditional mice.

On the other hand, a wearable glove design, which utilizes a gyroscope to track hand movements, does not rely on wrist motion to control the cursor. Instead, arm or even full-body movements can alter gyroscope parameters to control the pointer. From a theoretical standpoint, this makes a glove-based design superior to the vertical ergonomic mouse.

### **Handheld Trackball Mouse**

Among commercially available designs, perhaps the closest to the wearable glove mouse concept is the Handheld Trackball Mouse. Like the wearable glove mouse, it adopts a one-handed operation, replacing the traditional laser grid scanner with a trackball. However, its weight makes prolonged use uncomfortable, and compared to gyroscopic gesture control or laser scanning, the trackball's response speed is relatively slower. This limitation primarily arises from the restricted dexterity of the user's thumb, which controls the trackball (Natapov and MacKenzie).

The trackball design originates from older mouse models. However, unlike traditional trackballs—where a rolling ball on the desk feeds X and Y movement data to rotary encoders—modern handheld trackball designs rely on the thumb to rotate the ball for cursor control. This allows users to operate the mouse away from the desk.

Yet again, while the trackball's response speed is sufficient, it is constrained by the user's thumb agility, preventing it from reaching the efficiency level of a conventional mouse. Furthermore, handheld trackball mice are primarily used in presentation scenarios, where intense interaction is unnecessary. This is a stark

contrast to a wearable glove mouse, which, despite also enabling desk-free operation, is designed for a wider range of applications.

### **System Functional Features and Specifications**

The glove mouse is a multifunctional wearable input device designed to replace traditional mice and partial keyboard usage. Its primary goal is to enable cursor control and macro input through intuitive gestures and fingertip interactions, operating seamlessly via Bluetooth Low Energy (BLE) as a Human Interface Device (HID). The system architecture integrates several microcontroller-driven input and output modules to deliver real-time performance, ergonomic control, and user feedback.

#### **Key Functional Features:**

- **Gesture-Based Cursor Control:**

Utilizes a gyroscope (MPU6050) to translate hand orientation and motion into real-time cursor movement on a PC screen.

- **Macro and Text Input:**

A 4×4 matrix membrane keypad mounted on the palm enables numeric entry and T9-style character input.

- **Programmable Click & Mode Buttons:**

Fingertip-mounted tactile switches allow customized left/right clicks and mode toggles, offering a hands-free alternative to traditional desktop operation.

- **Rotary Encoder Input:**

A KY-040 rotary encoder provides scroll wheel functionality with additional speed gear shifting.

### **Real-Time Feedback System:**

- OLED Display (0.96") shows BLE connection status, temperature warnings, and mode indicators.
- RGB LED delivers visual cues for system state and error notifications.
- Buzzer offers audio feedback for actions such as boot-up, clicks, and alarms.
- Vibration Motor provides haptic responses for clicks and system events.
- BLE HID Communication:

The glove mouse communicates wirelessly with a host PC using Bluetooth Low Energy HID protocol, emulating a standard mouse.

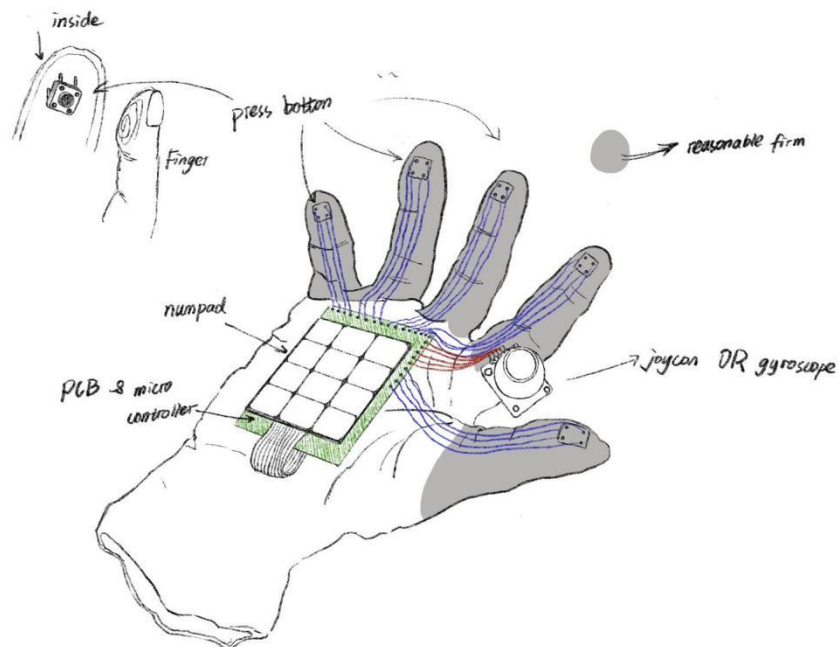
- **Overheat Monitoring & Safety Shutdown:**

Temperature sensors monitor system health and trigger shutdown logic to protect components from sustained high-load conditions.

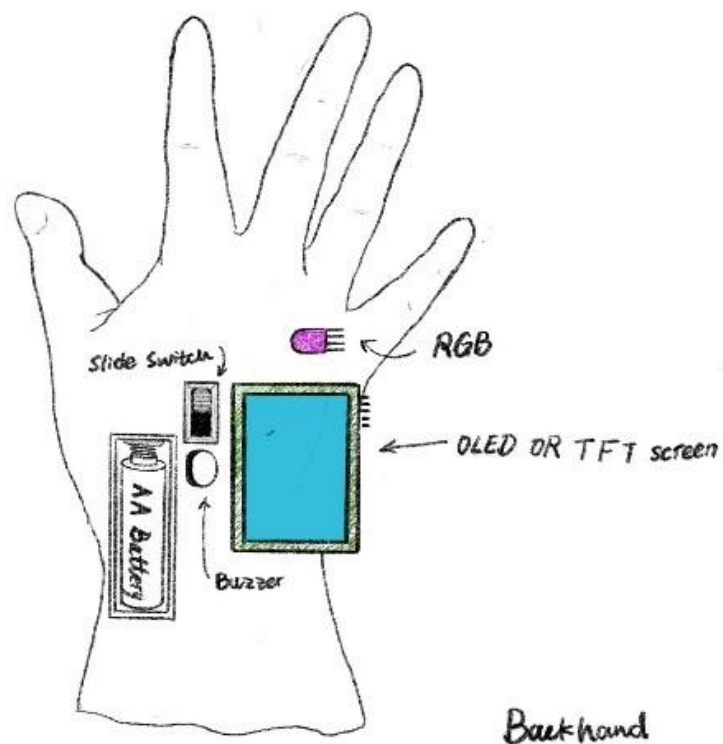
### **The early design and planing for inputs and outputs**

The early design featured a dual-layer glove structure, with the main circuit board and wiring embedded between the inner and outer layers. A matrix keypad was positioned on the palm to enable basic input functions, while a joystick was installed at the front of the thumb to serve both as a precise cursor control tool and as the mouse scroll wheel. The back of the glove housed the power supply, display screen,

main power switch, as well as a speaker and an RGB LED for visual and audio feedback. (See attached diagram for design illustration.)



GLOVE MOUSE DESIGN.



The corresponding input and output devices are as follows:

### **Input Devices:**

#### **Joycon (Joystick Module):**

Installed near the thumb, the joystick allows precise control of the cursor position. It also functions as the scroll wheel, enabling smooth navigation across pages and documents.

#### **Numpad (Matrix Keypad):**

Located on the palm, the matrix keypad provides basic input functions such as typing numbers and executing commands. It is designed for quick thumb or other finger operation, offering convenience without the need for a full-size keyboard.

#### **Gyroscope:**

Embedded within the glove, the gyroscope detects hand orientation and movement, enabling gesture-based control of the cursor. This allows the user to move the pointer simply by adjusting the angle and position of the hand in space.

### **Output Devices:**

#### **OLED Screen:**

Mounted on the back of the glove, the OLED screen displays system status, connectivity information (such as Bluetooth and Wi-Fi status), and real-time feedback

during operation. It helps the user monitor the glove's performance without needing an external display.

#### RGB LED:

Also located on the back of the glove, the RGB LED provides visual feedback for different operational modes. For example, it can indicate power status, connection status, or errors through various colors and flashing patterns.

#### Buzzer:

The buzzer delivers audio alerts for key events such as successful Bluetooth pairing, mode changes, or low battery warnings. It enhances the user experience by providing clear, audible notifications during operation.

### **Initial Circuit Design:**

The initial circuit design followed a straightforward approach to integrate all essential input and output devices. At the core of the system was the ESP32 microcontroller, responsible for managing all sensor data and controlling the output modules. The circuit included direct connections between the microcontroller and each peripheral component, with minimal use of expanders or external power management circuits.

The joystick and gyroscope were both connected to the analog and I2C pins of the ESP32 to handle motion control inputs. The matrix keypad was wired to the digital GPIO pins for basic key input functions.



## **Limitations of the Early Design**

### **1. Insufficient Input and Output Interfaces:**

The early design featured too few input and output components, which limited its functional capabilities. Without sufficient controls and feedback systems, the glove mouse could not fully meet the intended user interaction experience.

### **2. Redundancy Between Joystick and Gyroscope:**

Both the joystick and gyroscope were implemented for cursor control, leading to overlapping functionalities. This redundancy complicated the control logic without significantly improving precision or responsiveness.

### **3. Single Battery Power Supply Insufficiency:**

Relying on a single battery was not enough to meet the power demands of all components, including the OLED display, RGB LED, and multiple sensors. This resulted in unstable power delivery and reduced operating time.

### **Accidental Activation of Slide Switch:**

4. The early prototype's slide switch was prone to unintentional activation due to its placement and sensitivity. This caused unexpected mode changes and disrupted normal operation during use.



### **Initial Design Improvements(As Implemented in the Final Project)**

Based on the limitations of the initial prototype, the following specific improvements were applied in the actual project:

#### **Expanded Input and Output Interfaces (Implemented):**

A rotary encoder switch was introduced for mouse wheel scrolling, providing tactile feedback and a realistic scrolling experience, effectively replacing the initial joystick-wheel combination.

A vibration motor (coin style, 10mm x 2mm, 11000 RPM) was integrated to deliver haptic feedback, enabling the glove to respond physically to events such as clicks, mode changes, or game triggers.

The OLED screen and RGB LED were retained, while the buzzer control was optimized for improved power efficiency and clearer audio signaling.

#### **Clear Functional Separation Between Joystick and Gyroscope (Implemented):**

The joystick originally intended for cursor control was fully removed to eliminate redundancy.

The gyroscope was assigned complete responsibility for motion tracking, providing accurate and responsive control based on hand movement.

The rotary encoder and slide potentiometer were designated for scrolling and auxiliary control functions, maintaining clear separation between input roles.

**Upgraded Power Management System (Implemented):**

To address high power demands, a stable USB power supply was utilized for testing phases, with provisions in the circuit design for future dual battery integration.

MOSFET-based control circuits were implemented to safely drive power-hungry components like the vibration motor, preventing overload on the microcontroller's power lines.

**Added Overheat Protection Mechanism (Implemented):**

An overheat protection routine was introduced in the firmware to monitor the operating conditions of high-drain components, such as the vibration motor and RGB LED.

Control logic was configured to automatically limit or shut down intensive output functions during continuous use, effectively preventing overheating of both the motor and the control circuitry.

This feature ensures safer operation and extends the lifespan of sensitive electronic components under prolonged use.

**Improved Switch Design to Prevent Misoperation (Implemented):**

The original slide switch, which was prone to accidental activation, was replaced with dedicated GPIO-based push buttons for mode control.

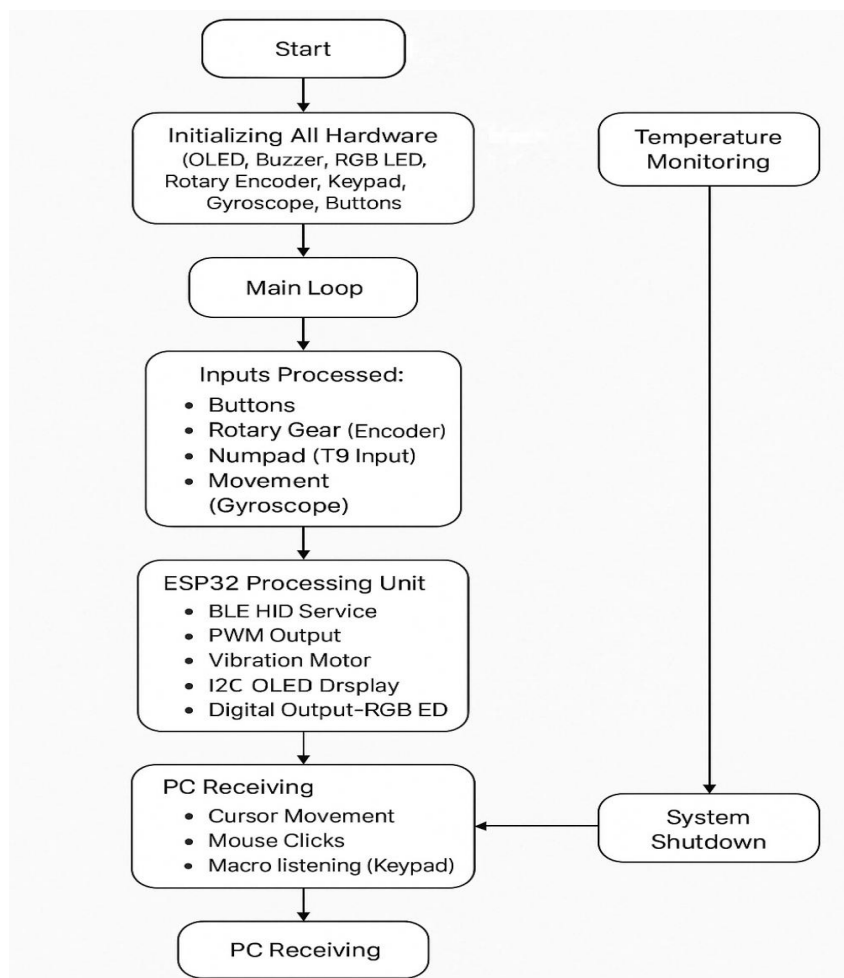
These buttons were strategically positioned to minimize the risk of unintentional activation during operation, enhancing overall control reliability.

These improvements successfully transformed the glove mouse from a conceptual design into a functional prototype, addressing key shortcomings and enhancing user experience through targeted hardware optimizations.

## Logic Design

The glove mouse system is driven by a single consolidated firmware, developed in MicroPython and deployed onto the ESP32 microcontroller. The code integrates multiple functional modules, handling motion tracking, input detection, output signaling, and communication with external devices.

### System block diagram:



## Code Structure and Library Imports

The glove mouse firmware is structured in a single comprehensive main.py file for ease of deployment and debugging. The codebase is fully written in **MicroPython**, leveraging the hardware features of the ESP32 microcontroller and integrating multiple input and output devices through clear modular functions.

### Core Libraries :

1. Machine(default micro python implemented): Provides access to hardware-level features, including GPIO control, PWM for the buzzer, and I2C communication for the OLED display and sensors.

2. time(default micro python implemented): Manages delays, timing events, and execution flow control.

I2C, Pin, PWM: Specific hardware classes used for peripheral connections.

3. ssd1306(default micro python implemented, self modified): Manages communication with the OLED display for real-time status updates.

4. hid\_services(open source, heavily modified): Implements BLE HID (Human Interface Device) profile(Groefsema, 2021), allowing the glove to act as a wireless mouse for the computer.

## Theory of Operation

The glove mouse operates as an integrated human-computer interaction device, utilizing sensor-driven input acquisition and BLE-based output delivery. The system's core functionality relies on real-time motion tracking, digital keypad input interpretation, and event-driven outputs. This section outlines the working principles behind each functional block and includes relevant calculations and signal analysis for key components.

### 1. Gyroscope-Based Cursor Control

The MPU6050 gyroscope captures angular velocity ( $^{\circ}/s$ ) and acceleration ( $m/s^2$ ) data, which is interpreted by the ESP32 to simulate mouse movement.

#### Conceptual Flow:

User tilts hand  $\rightarrow$  sensor detects  $\Delta\theta$  (angle change)

ESP32 calculates directional speed

Resulting values are mapped to HID report: `mouse.set_axes(x, y)`

#### Simplified Motion Mapping Equation:

$$V_{\text{cursor}} = K \times \theta_{\text{rate}}$$

Where:

$$V_{\text{cursor}} = \text{cursor speed (pixels/s)}$$

$\theta\_rate$  = angular velocity in  $^{\circ}/s$

$K$  = scaling factor based on screen DPI and desired sensitivity

**Example:**

If the user rotates their hand by  $20^{\circ}/s$  in the X-axis, and the sensitivity factor

$$K = 2.5$$

$$V\_cursor = 2.5 \times 20 = 50 \text{ pixels/s}$$

## 2. Rotary Encoder & Speed Gearing

The rotary encoder provides gear shifting, which determines how cursor movement responds to gyroscope input.

Gear Level	Scaling Factor (K)
Gear 1	1.0
Gear 2	2.5
Gear 3	4.0

The encoder also provides scroll functionality by generating a pulse stream:

$$1 \text{ detent} = 1 \text{ pulse pair}$$

With a debounce interval of 5 ms, max detection rate = 200 detents/sec

In addition to gear control, the encoder's press-down action triggers a cursor logic switch between two motion modes:

Horizontal Mode: Cursor X/Y controlled by hand tilts with natural mapping.

Vertical Mode: Axis remapping reverse the X/Y logic applies for vertical palm orientation.

### 3. T9 Keypad Input Logic

The keypad uses a matrix scan system ( $4 \text{ rows} \times 4 \text{ columns} = 16 \text{ keys}$ ).

Each keypress is detected through GPIO row/column polling.

Due to BLE HID limitations, the glove itself cannot emulate a full keyboard.

Instead, the system uses a two-part logic:

**On-glove input:** Pressing a numpad key sends a unique, physically implausible cursor movement pattern (e.g., simultaneous upward and downward acceleration at the same time).

**On-PC macro:** A Python listener running on Windows detects these unique motion patterns and interprets them as specific key inputs using a T9 layout (see Appendix codes for mapping).

This system bypasses the HID input type limitation while preserving fast and intuitive typing input.

#### T9 Text Selection Delay:

To enable multi-press input on the same key (e.g., “2” → A, B, C), a delay window is implemented:

$t\_window = 0.3$  seconds

If a second press occurs within  $t\_window$ , the character cycles to the next option. Otherwise, it is accepted and sent to the PC.

#### **4.Primary Button Mode Switching**

The glove includes dedicated pushbuttons mounted at the fingertips or sides of the glove. These are used for:

Left click

Right click

#### **Mode switching**

The mode switching button allows toggling between two main operating modes:

Mouse Mode: All inputs control cursor movement and clicking directly via BLE HID reports.

Keyboard Mode: Inputs (especially from the keypad) send special motion patterns instead of direct cursor or click commands. These are interpreted by the macro system on the PC to generate complex key combinations or macros.

This mode switching is accompanied by OLED display status updates and vibration or audio signal confirmation (via buzzer or vibration motor)



The toggle logic ensures that only one mode is active at a time, and all input routing adjusts accordingly(see appendix).

## **Product Operating Instructions**

This section provides a clear, step-by-step guide for setting up, wearing, and using the glove mouse. Designed for ease of use, the instructions assume the user has a powered glove and a Windows-based computer with Bluetooth capability.

### **Setup Before Use**

Charge the glove or connect it via USB to ensure full power during operation. Put on the glove with the main control PCB on the back of the hand and keypad aligned with the palm.

- Ensure fingertip buttons are aligned and comfortable to press naturally.
- Turn on the glove using the side-mounted power switch.
- Initial Boot and Bluetooth Connection
- Once powered, the OLED screen displays the boot status.
- On your PC, open Bluetooth Settings → Add Device.
- Select "Glove\_Mouse" or the advertised name.
- When connected, the OLED screen shows “Connected”.

### **Cursor Control (Mouse Mode)**

Slightly tilt your hand to move the cursor:

Tilt left/right = move left/right

Tilt up/down = move up/down

**Adjust movement speed by rotating the gear selector (rotary encoder).**

To click:

Press left fingertip button for left click

Press right fingertip button for right click

**Mode Switching**

Press the mode button to toggle between:

- Mouse Mode
- keyboard Mode

**T9 Typing (Macro Mode)**

In Macro Mode, use the palm-mounted keypad like a T9 phone:

Press Key2 → cycle through A, B, C

Press Key4 → G, H, I, etc.

Wait ~0.3 seconds to confirm a character before switching to another key.

The Windows macro listener interprets movement signals and translates them into keypresses.

**Overheat Protection**

If the glove overheats:

OLED will show a warning message

The buzzer will sound a continuous alert

System will automatically shut down outputs to protect components.

## After Use

Power off the glove using the switch.

Remove the glove gently and store in a cool, dry place.

Recharge if needed.

## Maintenance Requirements

To ensure long-term reliability and performance, the glove mouse requires occasional maintenance. The following steps are recommended:

**Battery Care:** Recharge the glove regularly. Avoid over-discharging Li-ion batteries. If using USB power during testing, unplug after each session to prevent overheating.

**Sensor & Pin Check:** Inspect exposed wires, GPIO pin headers, and button connections weekly. Re-solder any loose joints and ensure the gyroscope and keypad remain firmly seated.

**Cleaning:** Wipe the outer glove surface with a dry cloth. Avoid using water or solvents near electronic components. Dust buildup on the OLED screen or rotary encoder should be removed with compressed air.

**Software Updates:** If firmware changes are made (e.g., code tweaks for sensitivity or macros), re-upload the updated `main.py` using a USB-to-serial tool and verify functionality through Putty.

**Component Storage:** When not in use, store the glove in a dry, cool environment. Avoid leaving it in direct sunlight or inside bags where pressure may bend the PCB or pull loose wires.

## **Code Architecture Highlights:**

### **Hardware Initialization:**

All hardware components are initialized at the beginning of the script, including the OLED screen, buzzer, RGB LED, rotary encoder, keypad, buttons, and BLE mouse.

### **Peripheral Functions:**

Specific functions handle output controls, such as:

1. `set_rgb()` to manage RGB LED status indicators.
2. `play_startup_sound()`, `play_connection_sound()`, and `play_overheat_alarm()` for audio feedback.
3. `update_speed_indicator()` to reflect the current speed mode using RGB colors.
4. Input Scanning and Handling:
5. Rotary encoder state is monitored via `check_rotary_button()` to adjust cursor movement speed.
6. The matrix keypad is scanned in `scan_keypad()` for additional commands.
7. Gyroscope readings are processed in `read_accel()` for gesture-based cursor control.

### **Safety Features:**

Temperature readings from the onboard sensor are obtained via `read_temperature()`.

`overheat_shutdown()` is triggered automatically if the temperature exceeds safe operating limits, displaying an alert on the OLED, sounding an alarm, and halting system operation.

### Bluetooth HID Operation:

The `ble_mouse_start()` function initializes and manages BLE HID services, including advertising, connection handling, and feedback display.

**Once connected, the glove emulates mouse movement and button clicks over BLE.**

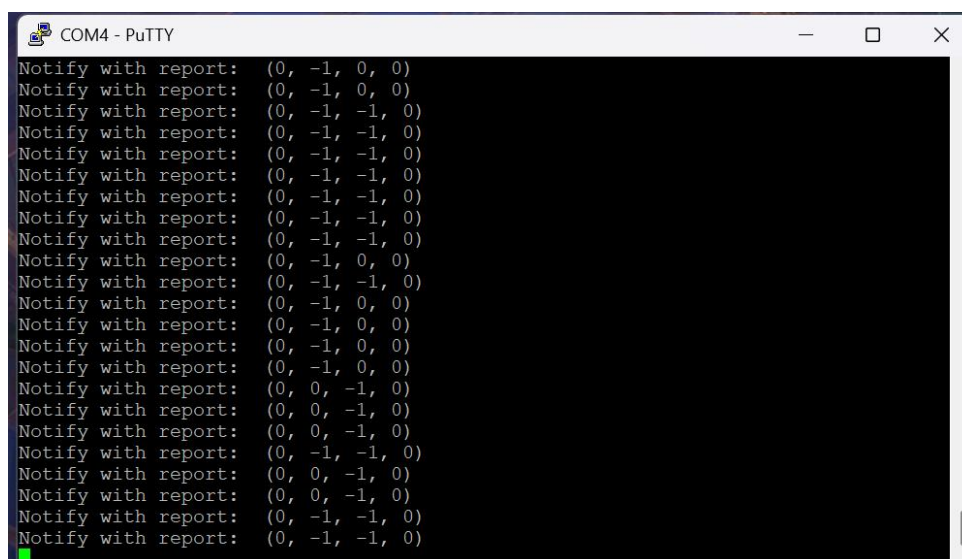
### Main Loop Execution:

The `control_mouse()` function continuously reads sensor data, processes input events, checks for overheat conditions, and sends appropriate HID reports for mouse control.

### Integrated Debugging and Monitoring:

Throughout the code, real-time debugging outputs are provided via serial print statements, which are monitored using PuTTY during development. This includes system initialization status, sensor readings, speed adjustments, BLE connection updates, and overheat warnings.

### Putty running examples:



```
COM4 - PuTTY
Notify with report: (0, -1, 0, 0)
Notify with report: (0, -1, 0, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, 0, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, 0, 0)
Notify with report: (0, -1, 0, 0)
Notify with report: (0, -1, 0, 0)
Notify with report: (0, 0, -1, 0)
Notify with report: (0, 0, -1, 0)
Notify with report: (0, 0, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, 0, -1, 0)
Notify with report: (0, 0, -1, 0)
Notify with report: (0, -1, -1, 0)
Notify with report: (0, -1, -1, 0)
```

## **Circuit Design**

The glove mouse system is built around a centralized Freenove ESP32-WROOM-32E microcontroller, directly connected to all input and output modules as shown in the schematic diagram (see attached image).

This latest circuit design reflects the final hardware layout of the working prototype.

### **Central Microcontroller (ESP32-WROOM-32E):**

Acts as the core controller, managing input acquisition, output control, communication protocols, and safety monitoring.

Power supplied via onboard 5V and 3.3V rails. GPIO pins are directly assigned to peripherals for clean, efficient wiring.

### **Input Devices:**

#### **1. Rotary Encoder Module KY-040 (U8):**

Connected to clk, dt, sw, vcc, gnd, handling scroll functionality and speed gear shifting.

#### **2. 4x4 Matrix Membrane Keypad (U9):**

Connected through int1–int8, enabling flexible command input and numeric functions.

#### **3. Push Buttons (U7, U10):**

Connected via int1pb, int2pb, int1, gndSw1, configured for primary actions including left click, right click, and mode switching.

**Sensor Modules:**

## 1. MPU6050 Gyroscope (U4):

Connected through VCC, GND, SCL, SDA, captures motion data for real-time cursor control.

## 2. MPL3115A2 Pressure Sensor (U3):

Connected using SDA, SCL, VCC, GND, reserved for future integration, such as environment-based controls.

**Output Devices:**

## 1. OLED Display 0.96" (U2):

Wired via SDA, SCL, VCC, GND, displays connection status, temperature warnings, and operational feedback.

## 2. RGB LED (LED1, LED3):

Each color channel connected separately, providing multi-color indicators for mode and status notifications.

## 3. Buzzer (LS1):

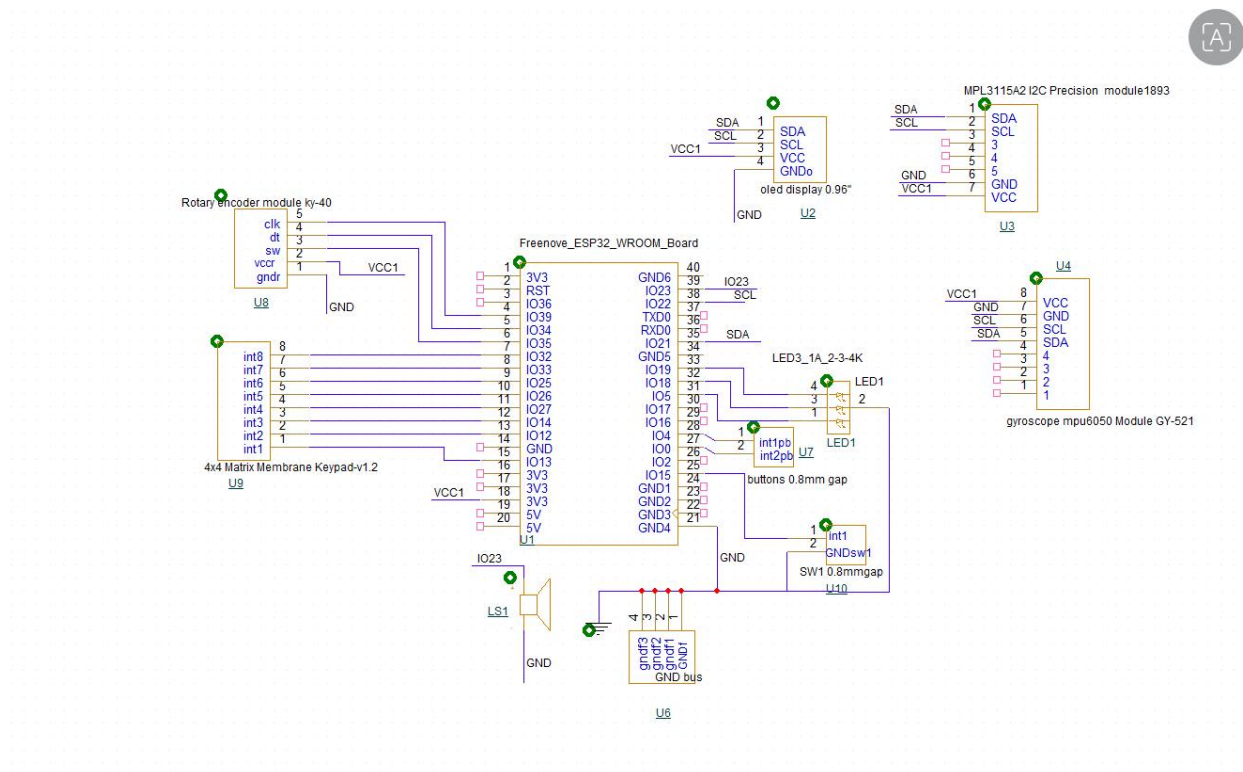
Connected to IO23 and GND, delivers audio alerts for startup, connection, and overheat alarms.

**Power Distribution:**

Power lines are distributed clearly with VCC1 and GND, ensuring stable voltage supply to all modules.

Common ground buses are used (U6) to prevent noise and voltage drops, supporting reliable operation across components.

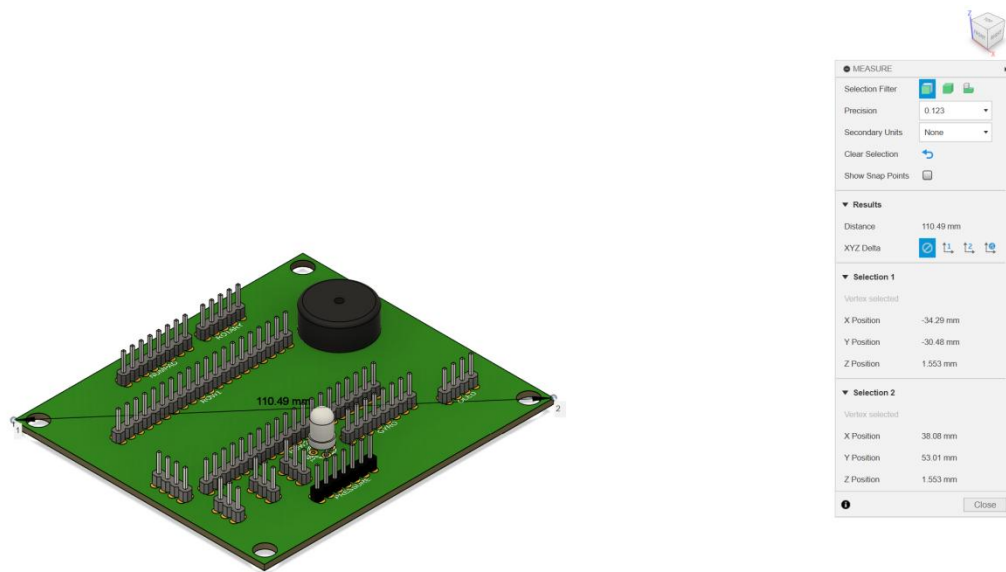
This circuit represents the final hardware architecture of the glove mouse project, balancing functionality, reliability, and future expandability for further development.



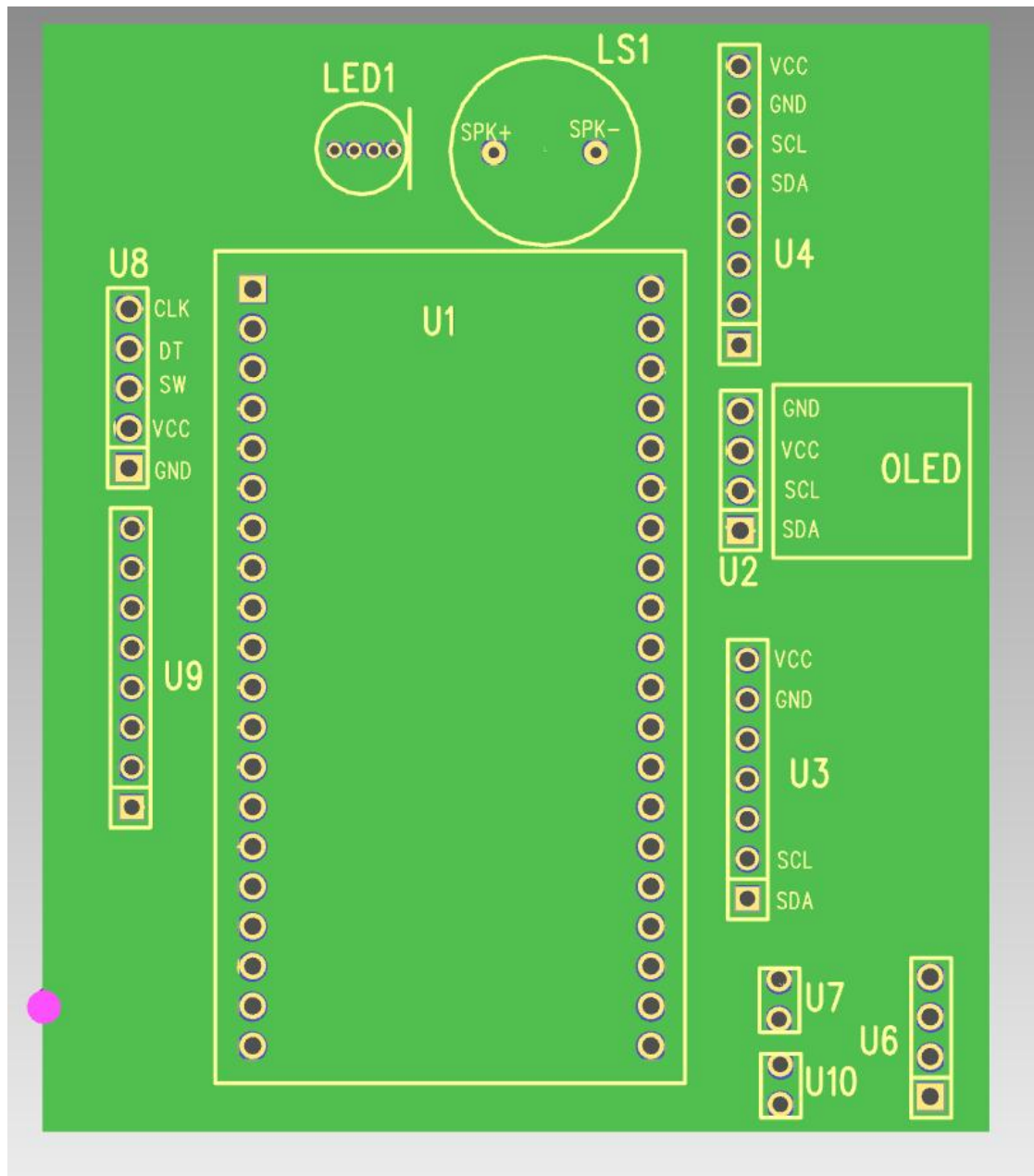




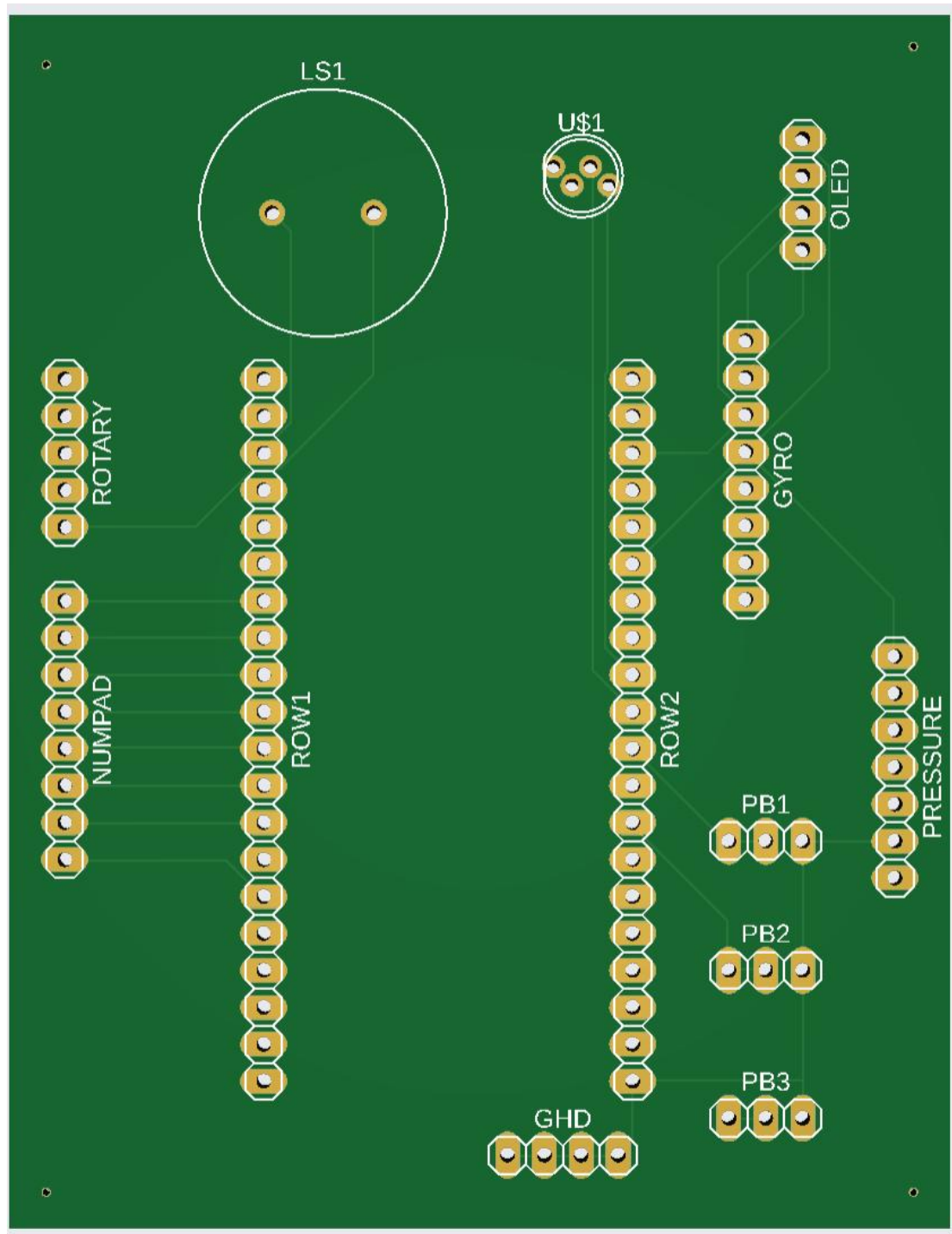
The printed layout diagram of this design is as follows:



However, due to the limitations of the single-sided design, the overall size of the PCB was too large for glove installation. To address this issue, the improved PCB version adopts a double-sided routing approach, which significantly reduces the required space. Please refer to the image below for version 2.0.



In the improved version, the V2.0 PCB measures only 60.3 mm × 70.5 mm, achieving significant space savings. However, to accomplish this compact size, the components had to be stacked and closely arranged for soldering, which introduced concerns regarding heat dissipation and increased the complexity of assembly. To address these challenges, the further improved V3.0 PCB design is shown below.

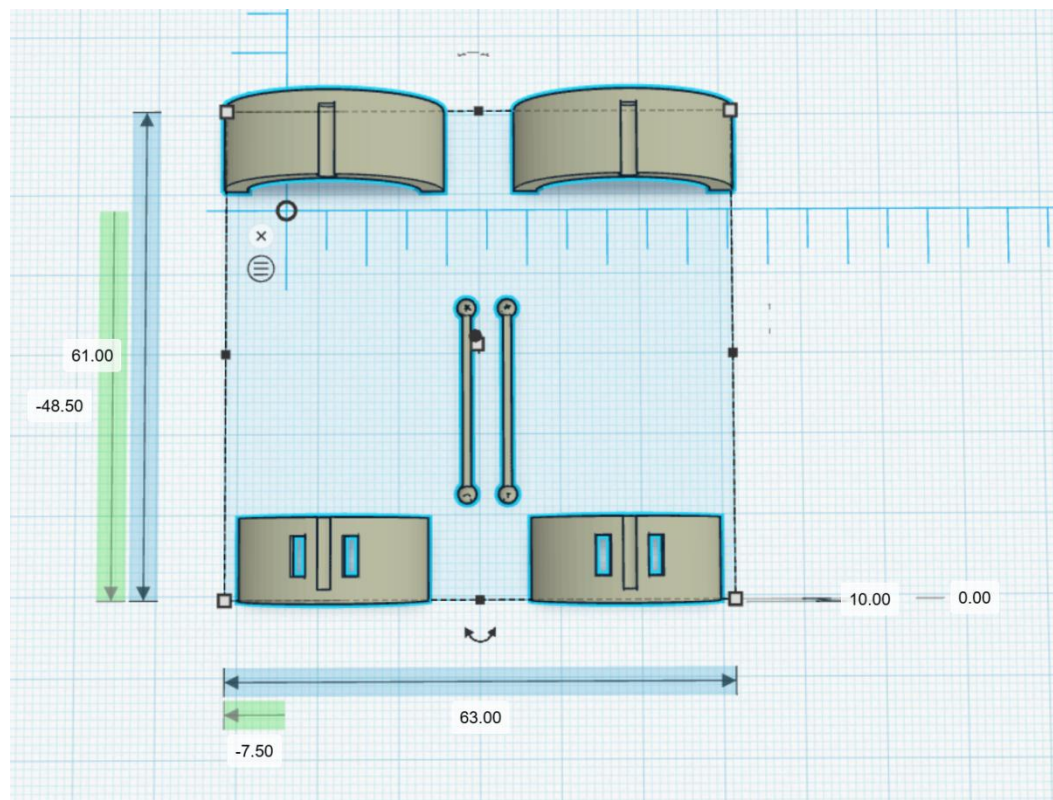


This design, while maintaining a compact size of 64.7 mm × 83.5 mm, provides sufficient space for all components. Ultimately, this version of the PCB was selected for the final design.

### 3D Printing Design

The 3D printing component of this project focuses on creating mechanical supports for the fingertip-mounted buttons, rather than a full enclosure. A total of four custom-designed pieces were printed to provide both mounting and pressing support for the trigger buttons.

The printed parts are designed exclusively to improve the button press experience at the fingertips, providing mechanical support for reliable activation without adding bulk to the glove.



#### Component Breakdown:

##### Two Top Pieces:

These are mounted directly on each fingertip, securely holding the tactile buttons in place. The design ensures precise alignment of the button relative to the

finger movement, maintaining consistent feedback and preventing button displacement during use.

#### Two Bottom Pieces:

These parts feature integrated lever-like structures, functioning as support platforms beneath each fingertip. When the user curls the finger to press the button, the bottom piece provides a solid mechanical resistance point, improving tactile response and reducing fatigue.

#### **Ergonomic Considerations:**

The shapes of both the top and bottom pieces follow the natural curvature of the fingers to maintain comfort during extended use.

The design ensures that button presses feel natural and responsive, without requiring excessive force.

#### **Material and Manufacturing:**

Lightweight PLA material was selected for its ease of printing and sufficient structural strength to withstand repetitive presses.

The parts were printed with fine resolution to achieve smooth surfaces, enhancing comfort and mechanical precision.

**Integration with Glove:**

The printed pieces are securely attached to the glove at the fingertip positions, with careful alignment to ensure that the levers and button mounts function effectively as a single unit.

Wiring from the buttons is routed through the glove fabric to connect with the main PCB, maintaining flexibility and unobstructed hand movement.

This targeted 3D printing solution effectively enhances the user interface of the glove mouse, providing mechanical reliability and tactile feedback for fingertip button presses while preserving comfort and wearability.

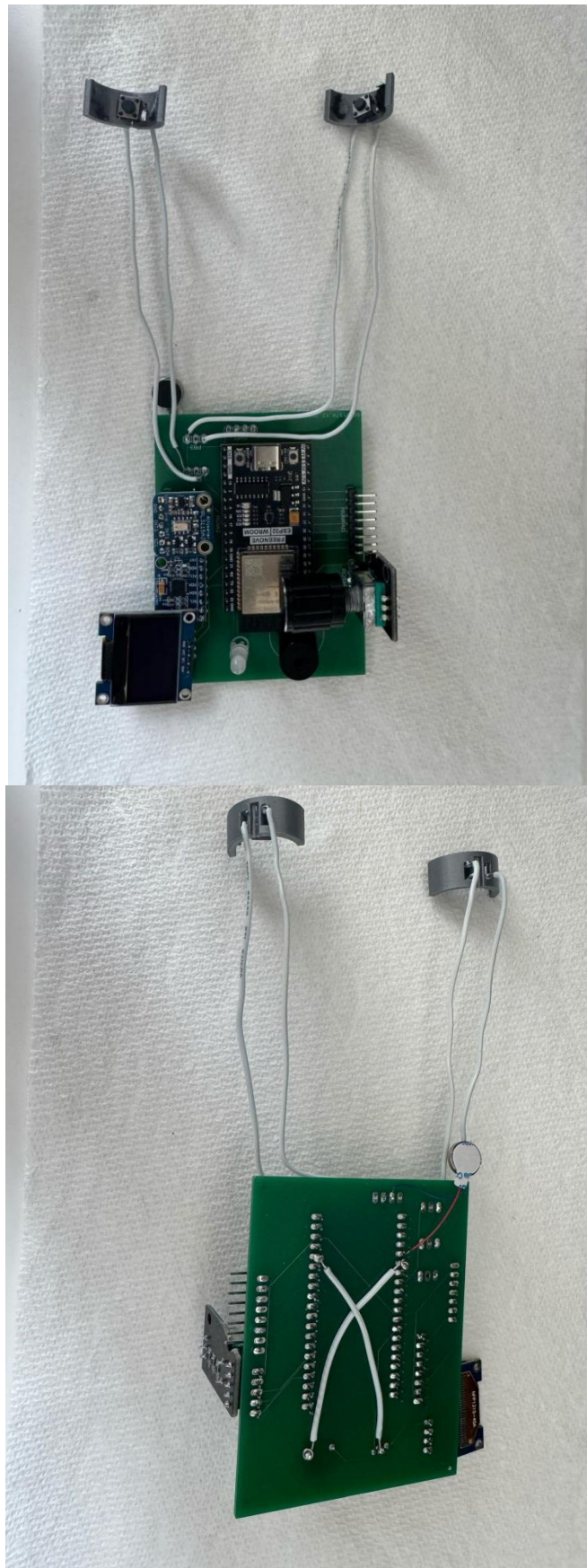


## Final Bill of Materials (BOM)

<b>Bill Of Materials (BOM) - Glove Mouse</b>							
# Component	Description / Notes	Qty	Unit Price (USD)	Total Price (USD)	Available At	Manufacturer	
1	ESP32-WROOM-32E Dev Board	1	\$12.00	\$12.00	<a href="https://www.amazon.ca/Freenove-ESP32-1">https://www.amazon.ca/Freenove-ESP32-1</a>	FreeNove	
2	MPU6050 IMU Sensor Module	1	\$12.00	\$12.00	<a href="https://www.amazon.ca/Pre-Soldered-Acc">https://www.amazon.ca/Pre-Soldered-Acc</a>	Generic	
3	Tactile Push Button	3	\$0.25	\$0.75	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	E-Switch	
4	Rotary Encoder (ZUDKSUY EK2045x4C)	1	\$3.00	\$3.00	<a href="https://www.amazon.ca/Taiss-KY-040-Encc">https://www.amazon.ca/Taiss-KY-040-Encc</a>	Taiss	
5	Mini Vibration Motor (Coin Type)	2	\$1.50	\$3.00	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	Seeed Technology Co., Ltd	
6	1kΩ Resistors	4	\$0.05	\$0.20	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	Stackpole Electronics Inc	
7	Perfboard / Custom PCB	1	\$10.00	\$10.00	N/A	Seneca PCB lab	
8	Jumper Wires / Female Headers	1	\$5.00	\$5.00	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	CNC Tech	
9	Power bank	1	\$27.00	\$27.00	<a href="https://www.amazon.ca/iWALK-Portable-C">https://www.amazon.ca/iWALK-Portable-C</a>	iWalk	
10	Glove (Fabric / DIY)	1	\$12.00	\$12.00	<a href="https://www.amazon.ca/SURAWIL-Leather">https://www.amazon.ca/SURAWIL-Leather</a>	SURAWIL	
11	OLED Display (SSD1306, 128x64)	1	\$6.00	\$6.00	<a href="https://www.amazon.ca/iPistBit-Display-Lu">https://www.amazon.ca/iPistBit-Display-Lu</a>	iPistBit	
12	Buzzer Module	1	\$0.80	\$0.80	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	Same Sky	
13	RGB LED	1	\$0.30	\$0.30	<a href="https://www.digikey.ca/en/products/detail">https://www.digikey.ca/en/products/detail</a>	King Bright	
				Total:	\$91.05		



## PCB Assembly



## Product Overlook



## **Technical Limitations and Challenges Highlights**

### **Continuous Movement Tracking:**

With a traditional desktop mouse, continuous movement in one direction is achieved by a combination of sliding the mouse, briefly lifting it, and then repositioning it for another slide in the same direction.

However, the glove mouse, using continuous gyroscope tracking, cannot replicate this natural pause and repositioning, since the gyroscope continuously registers motion without interruption.

To overcome this challenge, an acceleration threshold algorithm was implemented. When the user quickly flicks their hand in a certain direction, the system detects this rapid movement and temporarily ignores it. This allows the user to simulate the traditional lift-and-slide motion, enabling multiple consecutive cursor movements in the same direction without unintended drift

### **Logical Cursor Tracking at Different Hand Angles:**

The tracking behavior differs when the user holds their palm parallel to the ground versus when the palm is held vertically.

To ensure consistent and intuitive control, the code was designed to detect these two specific hand orientations and apply distinct tracking strategies for each mode.

By switching between tracking modes dynamically based on hand angle, the system maintains smooth cursor control, allowing natural hand positioning without sacrificing accuracy.

**T9 Keypad Integration:**

The compact keypad naturally limits the range of possible inputs. To overcome this constraint, a T9 (Text on 9) input method was implemented, allowing the user to achieve nearly the same input capabilities as a full-sized keyboard, despite using a mini keypad.

**Current Technical Limitations and Challenges****Optimal Finger Support:**

The current support system relies on mechanical structures combining 3D-printed parts and linkages. While functional, this approach is far from ideal.

A better solution would involve using specialized flexible materials to construct the finger supports. These materials would provide firm resistance when bending the finger to press a button, while remaining soft and comfortable during idle use.

Such an improvement would greatly enhance both the pressing accuracy and the overall wearing comfort of the glove, making it more suitable for prolonged use.

**Speed and Precision:**

Compared to the laser grid scanner used in traditional optical mice, the gyroscope currently employed in the glove mouse has significant limitations in both speed and precision.

Even advanced models within the same gyroscope category struggle to match the responsiveness of a medium-quality laser sensor.

Addressing this issue may require the integration of specialized gyroscope modules, specifically designed for ultra-fast response and high sensitivity in small-range movements, to bridge the performance gap.

### **Prototype Hardware Resource Constraints:**

The current prototype operates within the limitations of the ESP32 microcontroller and basic sensor modules.

Limited processing power and memory restrict the possibility of running more advanced algorithms, such as predictive motion tracking or AI-assisted gesture recognition.

Future improvements could involve upgrading to more powerful hardware platforms or integrating dedicated co-processors to handle specific tasks.

### **Long-Term Wearability:**

The current glove design uses a wrap-around structure, which, while securing the components, adds noticeable weight and causes heat buildup.

Prolonged wear leads to discomfort, as the enclosed design hinders natural skin ventilation and increases sweating.

Enhancing long-term comfort would require a more breathable, lightweight structure, possibly adopting semi-open designs and advanced textile materials to balance structural support with skin breathability.

## **Future Improvements and Ideal Model**

### **Semi-Open Design:**

To improve long-term wearability, future versions of the glove mouse will adopt a semi-open structure.

By exposing non-critical areas and integrating breathable materials, the design will reduce heat accumulation and allow natural skin ventilation.

This approach will also reduce the weight of the glove, making it more comfortable for extended use without sacrificing structural integrity or support for the electronics.

### **Ultra-Lightweight Self-Supporting Structure:**

Instead of relying on bulky mechanical supports, the ideal model will incorporate self-supporting, ultra-lightweight materials that provide structural rigidity where needed while remaining flexible and lightweight.

Advanced composites or woven smart textiles could be used to deliver both strength and flexibility, significantly improving overall ergonomics.

This improvement aims to make the glove feel like a natural extension of the hand, enabling fluid movement and reducing fatigue during prolonged operation.

### **AI-Based Gesture Learning:**

Future designs will explore AI-powered gesture recognition algorithms to enhance control accuracy and flexibility.

Machine learning models can be trained to recognize user-specific gestures, allowing the system to adapt to individual usage patterns and improve over time.

This would enable more complex interactions beyond simple directional control, such as custom gestures for shortcuts, commands, or game controls, further expanding the glove mouse's capabilities.

These represent just some of the future improvement ideas for the glove mouse. With ongoing development, many more enhancements can be explored to optimize performance, comfort, and functionality, ultimately transforming the glove mouse into a high-performance, user-friendly wearable solution.

### **Conclusion**

The glove mouse project successfully demonstrates the integration of wearable technology with traditional computer input functionalities. By combining motion tracking, tactile feedback, wireless communication, and ergonomic design, the prototype achieves core usability goals while introducing innovative interaction methods.

Throughout the development process, multiple challenges were addressed, including continuous motion tracking, accurate cursor control at varying hand angles, and reliable integration of a compact keypad using the T9 input method. Hardware optimizations such as overheat protection, modular PCB design, and fingertip mechanical supports further improved the system's reliability and user experience.

While the current prototype delivers stable performance, the project also identifies key areas for future development, including improvements in materials for better comfort and support, advanced sensors for higher precision, and the potential integration of AI-driven gesture learning to expand functionality.

This project not only provides a working solution for replacing traditional desktop mice in specific scenarios but also lays a strong foundation for further research and development in wearable input devices. The glove mouse represents a step forward in creating flexible, intuitive, and portable human-computer interaction tools.



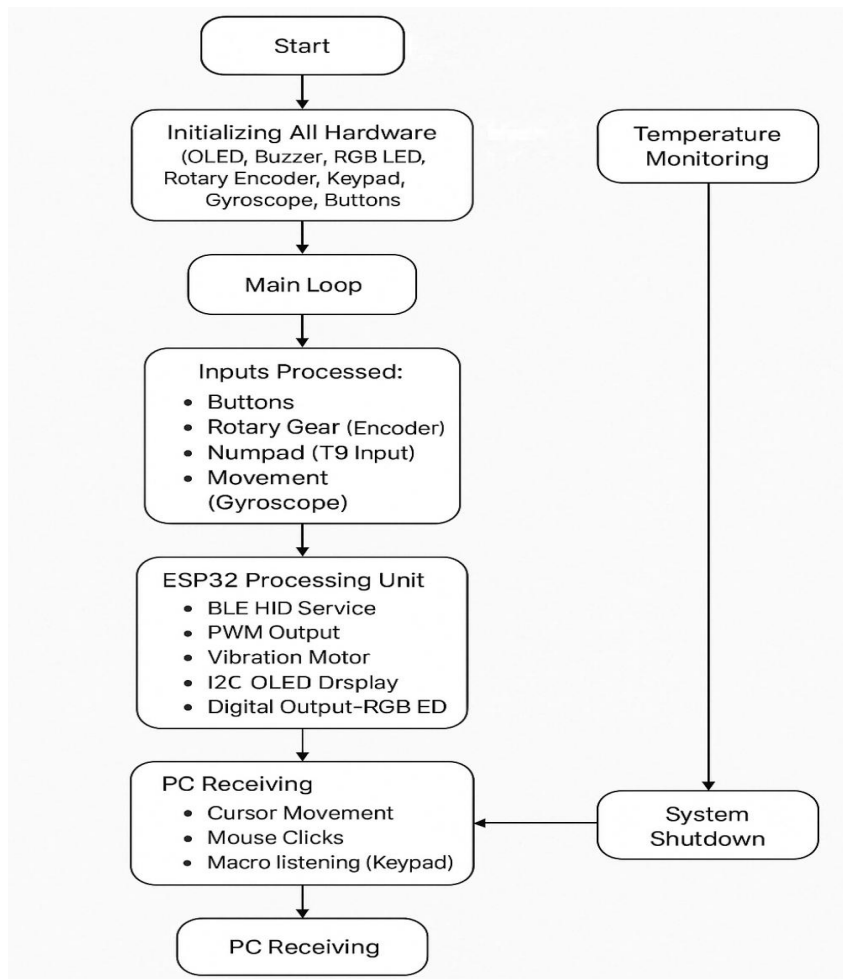
## References

- Gill, Ramaneeek. *The Innovative Computer Mouse*. 2012.
- Natapov, Daniel, and I. Scott MacKenzie. "The Trackball Controller: Improving the Analog Stick." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.
- Schmid, Anina B., et al. "A Vertical Mouse and Ergonomic Mouse Pads Alter Wrist Position but Do Not Reduce Carpal Tunnel Pressure in Patients with Carpal Tunnel Syndrome." *Journal of Applied Ergonomics*, vol. 46, 2015, pp. 164–169.
- Pramod. *Limitations of Traditional Mice and Future Trends*. 2024.
- "ESP32 Series Datasheet and Technical Reference Manual." *Espressif Systems*, 2023, [https://www.espressif.com/sites/default/files/documentation/esp32\\_technical\\_reference\\_manual\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf). Accessed 10 Apr. 2025.
- "ESP32-WROOM-32E User Guide." *Freenove*, <https://freenove.com/esp32-wroom-32e>. Accessed 10 Apr. 2025.
- "InvenSense MPU-6000 and MPU-6050 Product Specification." *TDK InvenSense*, <https://invensense.tdk.com/download-pdf/mpu-6000-datasheet/>. Accessed 10 Apr. 2025.
- "MPL3115A2 Pressure and Temperature Sensor Datasheet." *NXP Semiconductors*, <https://www.nxp.com/docs/en/data-sheet/MPL3115A2.pdf>. Accessed 10 Apr. 2025.
- Heerkog. *hid\_services.py*. GitHub, 2024, [https://github.com/Heerkog/MicroPythonBLEHID/blob/master/hid\\_services.py](https://github.com/Heerkog/MicroPythonBLEHID/blob/master/hid_services.py).

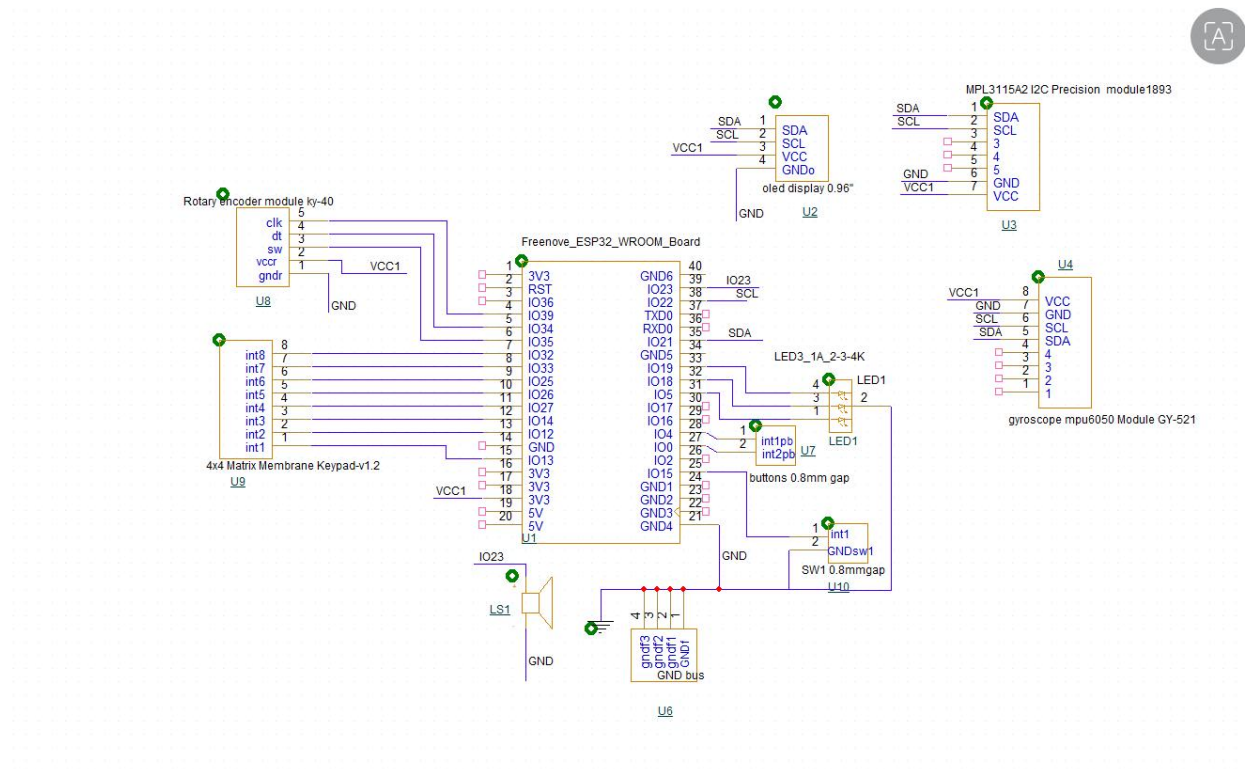
## Appendices

### Appendix 15A – System Block Diagram

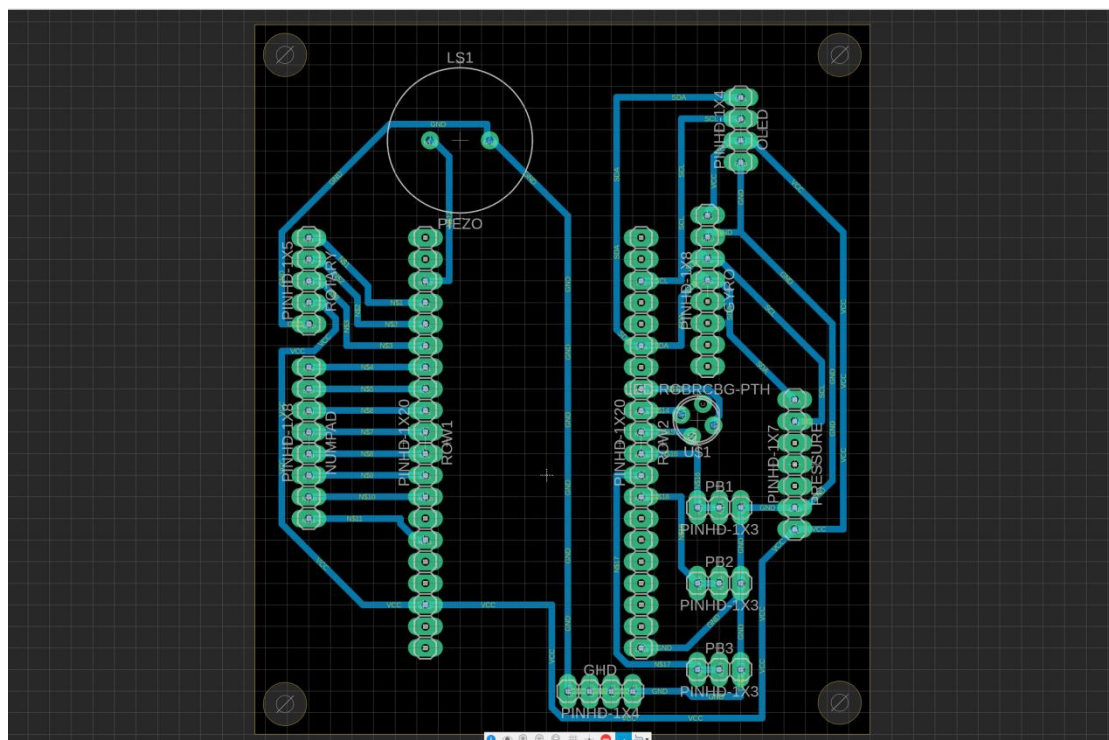
A diagram showing the interaction between input modules, ESP32, and output devices in the glove system.

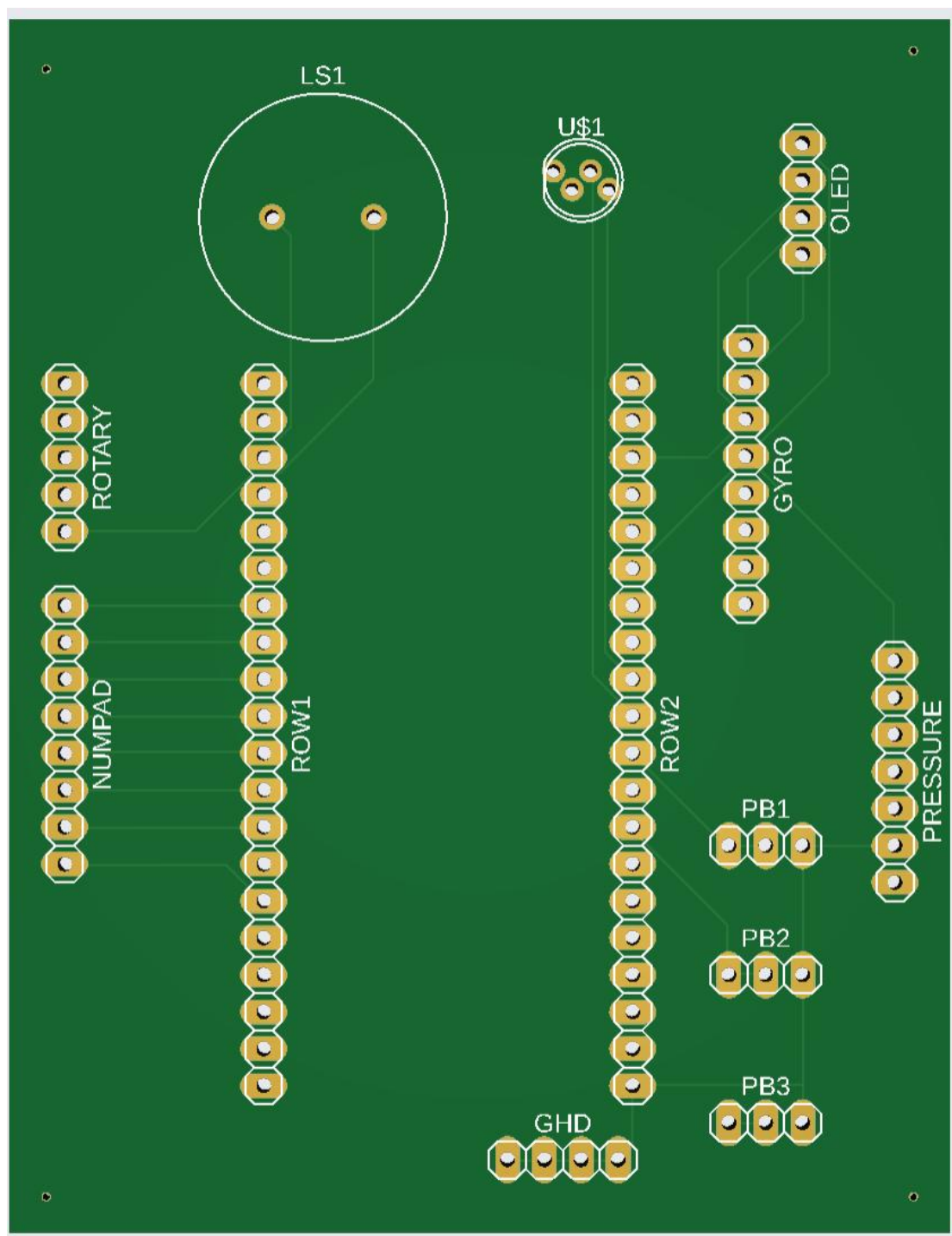


## Appendix 15B – Electrical Schematics

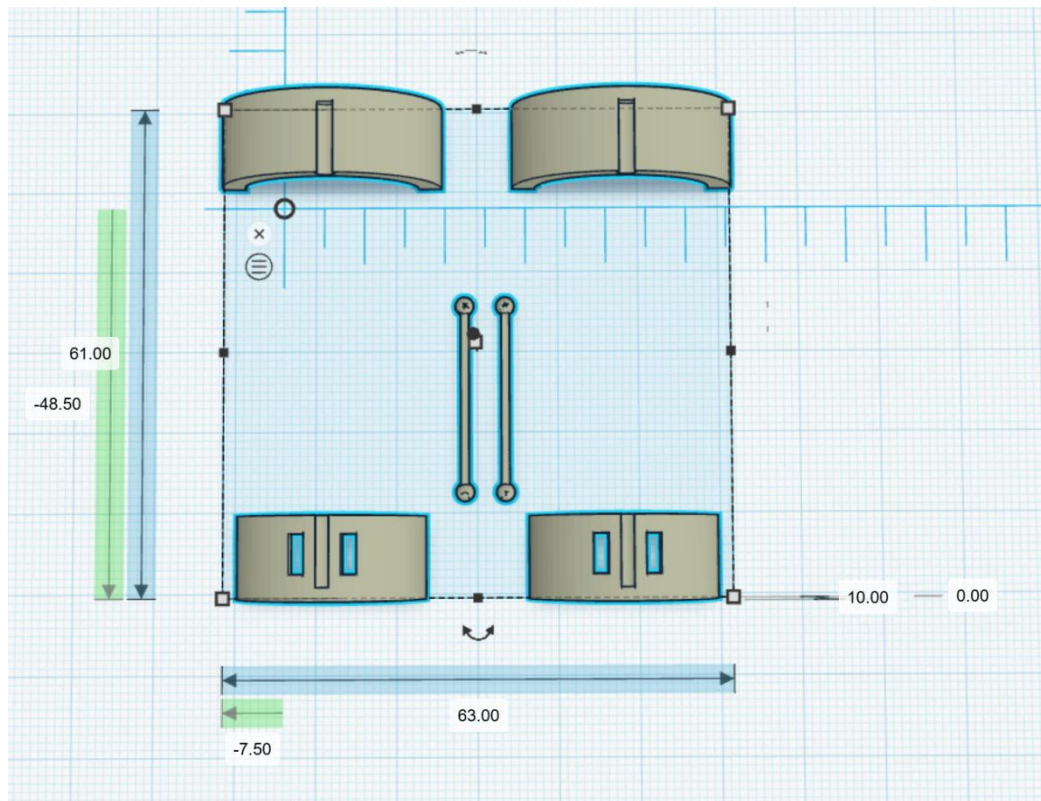


## Appendix 15C – PCB Layout and Routing Images





## Appendix 15D – Mechanical Diagram: 3D Printed Finger Supports



## Appendix 15E – IoT Dashboard

Not applicable for this prototype. Future versions may integrate MQTT/Firebase for real-time telemetry and cloud logging.

## Appendix 15F – Code Flowchart

### ESP32 side:

```
import time
import machine
from machine import I2C, Pin, PWM
from ssd1306 import SSD1306_I2C
from hid_services import Mouse

# === INITIAL SETUP ===
orientation_mode = 0
az_offset = 0.0
ay_offset = 0.0
keypad_mode = False
# Pins
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
oled = SSD1306_I2C(128, 32, i2c)
RED = Pin(19, Pin.OUT)
```

```

GREEN = Pin(18, Pin.OUT)
BLUE = Pin(5, Pin.OUT)
CLK = Pin(39, Pin.IN, Pin.PULL_UP)
DT = Pin(34, Pin.IN, Pin.PULL_UP)
ROT_SW = Pin(35, Pin.IN, Pin.PULL_UP)
rows = [Pin(pin, Pin.OUT) for pin in (32, 33, 25, 26)]
cols = [Pin(pin, Pin.IN, Pin.PULL_UP) for pin in (27, 14, 12, 13)]
main_button = Pin(17, Pin.IN, Pin.PULL_UP)
left_click_button = Pin(4, Pin.IN, Pin.PULL_UP)
right_click_button = Pin(16, Pin.IN, Pin.PULL_UP)
buzzer_pin = Pin(2, Pin.OUT)
buzzer_pin.value(0)
buzzer = PWM(buzzer_pin)
buzzer.freq(2000)
buzzer.duty(0)
vibrator = PWM(Pin(15))
vibrator.freq(2000)
vibrator.duty(0)
mouse = Mouse("ESP32 Glove Mouse")
speeds = [3.0, 4.0, 5.5]
current_speed_index = 0
last_dt_state = DT.value()
last_rotary_press_time = time.ticks_ms()
rotary_click_last = ROT_SW.value()
rotary_click_last_time = time.ticks_ms()
last_main_button_state = main_button.value()
last_key_time = time.ticks_ms()
keypad_map = [['D', 'C', 'B', 'A'],
               ['#', '9', '6', '3'],
               ['0', '8', '5', '2'],
               ['*', '7', '4', '1']]
macro_coords = {
    '1': (10, -10),
    '2': (-20, 20),
    '3': (15, -30),
    '4': (-25, -25),
    '5': (30, -15),
    '6': (-15, 30),
    '7': (40, -40),
    '8': (-35, 10),
    '9': (25, -5),
    '0': (-50, 50),
    'A': (60, -60),
    'B': (-70, 70),
    'C': (80, -10),
    'D': (-90, 90),
    '*': (100, -100),
    '#': (-110, 110),
}
}
def scan_keypad_macro_mode():
    global last_key_time
    now = time.ticks_ms()
    if time.ticks_diff(now, last_key_time) < 300:
        return
    for row_num, row_pin in enumerate(rows):
        row_pin.value(0)
        for col_num, col_pin in enumerate(cols):

```

```

    if col_pin.value() == 0:
        key = keypad_map[row_num][col_num]
        if key in macro_coords:
            x, y = macro_coords[key]
            print(f"[MACRO] {key} → Move to ({x}, {y})")
            mouse.set_axes(x=x, y=y)
            mouse.notify_hid_report()
            vibrate(0.1)
            last_key_time = now
            while col_pin.value() == 0:
                time.sleep(0.01)
        row_pin.value(1)
def set_rgb(r, g, b):
    RED.value(not r)
    GREEN.value(not g)
    BLUE.value(not b)
def vibrate(duration=0.2, intensity=512):
    vibrator.duty(intensity)
    time.sleep(duration)
    vibrator.duty(0)

def play_tone(duration_ms=100, frequency=2000):
    buzzer.freq(frequency)
    buzzer.duty(512)
    time.sleep_ms(duration_ms)
    buzzer.duty(0)
def play_connected_chime():
    play_tone(120, 880) # A5
    play_tone(120, 988) # B5
    play_tone(120, 1047) # C6
    play_tone(150, 1319) # E6
def play_mode_switch_chime():
    play_tone(100, 988) # B5
    play_tone(120, 784) # G5
def update_speed_indicator():
    if current_speed_index == 0:
        set_rgb(0, 1, 0) # Green
    elif current_speed_index == 1:
        set_rgb(0, 0, 1) # Blue
    elif current_speed_index == 2:
        set_rgb(1, 0, 0) # Red
def check_rotary_button():
    global current_speed_index, last_dt_state, last_rotary_press_time
    dt_state = DT.value()
    now = time.ticks_ms()
    if dt_state != last_dt_state:
        if time.ticks_diff(now, last_rotary_press_time) > 300:
            current_speed_index = (current_speed_index + 1) % len(speeds)
            update_speed_indicator()
            vibrate(0.1)
            last_rotary_press_time = now
    last_dt_state = dt_state
def check_rotary_click():
    global orientation_mode, rotary_click_last_time, rotary_click_last, az_offset, ay_offset
    current_click = ROT_SW.value()
    now = time.ticks_ms()
    if current_click == 0 and rotary_click_last == 1:

```

```

    if time.ticks_diff(now, rotary_click_last_time) > 300:
        orientation_mode ^= 1
        rotary_click_last_time = now
        _, ay, az = read_accel()
        az_offset = az if orientation_mode == 1 else 0.0
        ay_offset = ay if orientation_mode == 1 else 0.0
        oled.fill(0)
        oled.text("Orientation:", 0, 0)
        oled.text("Vertical" if orientation_mode else "Horizontal", 0, 12)
        oled.show()
        vibrate(0.2)
    rotary_click_last = current_click
def init_mpu6050():
    i2c.writeto_mem(0x68, 0x6B, b'\x00')
    time.sleep(0.1)
def read_accel():
    def convert(data):
        val = int.from_bytes(data, 'big')
        return val - 65536 if val > 32767 else val
    ax = convert(i2c.readfrom_mem(0x68, 0x3B, 2)) / 16384.0
    ay = convert(i2c.readfrom_mem(0x68, 0x3D, 2)) / 16384.0
    az = convert(i2c.readfrom_mem(0x68, 0x3F, 2)) / 16384.0
    return ax, ay, az
def ble_mouse_start():
    mouse.start()
    mouse.start_advertising()
    oled.fill(0)
    oled.text("ESP32 Mouse", 10, 0)
    oled.text("Pairing...", 20, 15)
    oled.show()
    while not mouse.is_connected():
        time.sleep(0.1)
    if mouse.is_advertising():
        mouse.stop_advertising()
    play_connected_chime()
    time.sleep(2)
    oled.fill(0)
    oled.text("Mode:", 0, 0)
    oled.text("Keypad" if keypad_mode else "Mouse", 0, 12)
    oled.show()
# === STARTUP SEQUENCE ===
set_rgb(1, 1, 1)
init_mpu6050()
update_speed_indicator()
oled.fill(0)
oled.text("Press Main BTN", 0, 0)
oled.text("to start BLE", 0, 10)
oled.show()
# === MAIN LOOP ===
while True:
    if main_button.value() == 0:
        while main_button.value() == 0:
            time.sleep(0.01)
        ble_mouse_start()
        while mouse.is_connected():
            if main_button.value() == 0 and last_main_button_state == 1:
                keypad_mode = not keypad_mode

```



```

oled.fill(0)
oled.text("Mode Switched:", 0, 0)
oled.text("Keypad" if keypad_mode else "Mouse", 0, 12)
oled.show()
vibrate(0.2)
play_mode_switch_chime()
last_main_button_state = main_button.value()
if keypad_mode:
    scan_keypad_macro_mode()
else:
    check_rotary_button()
    check_rotary_click()
    ax, ay, az = read_accel()
    if abs(ax) < 0.4: ax = 0
    if abs(ay) < 0.4: ay = 0
    if abs(az) < 0.4: az = 0
    speed = speeds[current_speed_index]
    if orientation_mode == 0:
        move_x = int(ax * 10 * speed)
        move_y = int(ay * 10 * speed)
    else:
        adjusted_az = az - az_offset
        adjusted_ay = ay - ay_offset
        if abs(adjusted_az) < 0.3: adjusted_az = 0
        if abs(adjusted_ay) < 0.3: adjusted_ay = 0
        move_x = int(adjusted_az * 10 * speed)
        move_y = int(adjusted_ay * 10 * speed)
    #move_x = max(-10, min(10, move_x))
    #move_y = max(-10, min(10, move_y))
    if move_x != 0 or move_y != 0:
        mouse.set_axes(x=move_x, y=move_y)
        mouse.notify_hid_report()
    if left_click_button.value() == 0:
        mouse.set_buttons(1, 0, 0)
        mouse.notify_hid_report()
        time.sleep(0.1)
        mouse.set_buttons(0, 0, 0)
        mouse.notify_hid_report()
    if right_click_button.value() == 0:
        mouse.set_buttons(0, 1, 0)
        mouse.notify_hid_report()
        time.sleep(0.1)
        mouse.set_buttons(0, 0, 0)
        mouse.notify_hid_report()
    time.sleep(0.05)
time.sleep(0.05)

```

### PC macro:

```

from pynput.mouse import Controller as MouseController
from pynput.keyboard import Controller as KeyboardController, Key
import time

```

```

mouse = MouseController()
keyboard = KeyboardController()

```

```

# === T9 and Number Maps ===

```

```

T9_MAP = {
    'Key2': 'abc',
    'Key3': 'def',
    'Key4': 'ghi',
    'Key5': 'jkl',
    'Key6': 'mno',
    'Key7': 'pqrs',
    'Key8': 'tuv',
    'Key9': 'wxyz',
    'Key1': '.,!?'
}

NUMBER_MAP = {
    'Key1': '1',
    'Key2': '2',
    'Key3': '3',
    'Key4': '4',
    'Key5': '5',
    'Key6': '6',
    'Key7': '7',
    'Key8': '8',
    'Key9': '9',
    'Key0': '0'
}

# === T9 State ===
class T9State:
    def __init__(self):
        self.last_key = None
        self.press_count = 0
        self.last_press_time = 0
        self.upper = False
        self.number_mode = False
        self.delay = 0.5 # 0.5s flush timeout

    def toggle_case(self):
        self.upper = not self.upper
        print(f"Case Toggled: {'UPPER' if self.upper else 'lower'}")

    def toggle_number_mode(self):
        self.number_mode = not self.number_mode
        print(f"Number Mode: {'ON' if self.number_mode else 'OFF'}")

    def update_key(self, key):
        now = time.time()
        if key == self.last_key and (now - self.last_press_time) < self.delay:
            self.press_count += 1
        else:
            self.flush()
            self.last_key = key
            self.press_count = 0
            self.last_press_time = now

    def flush(self):
        if self.last_key and self.last_key in T9_MAP:
            chars = T9_MAP[self.last_key]
            char = chars[self.press_count % len(chars)]

```

```

        if self.upper:
            char = char.upper()
            print(f" Typed: {char}")
            keyboard.type(char)
        self.last_key = None
        self.press_count = 0

t9 = T9State()

# === Special Functional Keys ===
def send_special(keyname):
    if keyname == 'Key0':
        keyboard.type(' ')
    elif keyname == 'KeyD':
        keyboard.press(Key.enter)
        keyboard.release(Key.enter)
    elif keyname == 'KeyA':
        keyboard.press(Key.backspace)
        keyboard.release(Key.backspace)
    elif keyname == 'Key*':
        t9.toggle_case()
    elif keyname == 'KeyB':
        t9.toggle_number_mode()

# === Macro Movement Mapping ===
macro_map = {
    (10, -10): 'Key1',
    (-20, 20): 'Key2',
    (15, -30): 'Key3',
    (-25, -25): 'Key4',
    (30, -15): 'Key5',
    (-15, 30): 'Key6',
    (40, -40): 'Key7',
    (-35, 10): 'Key8',
    (25, -5): 'Key9',
    (-50, 50): 'Key0',
    (60, -60): 'KeyA',
    (-70, 70): 'KeyB',
    (80, -10): 'KeyC',
    (-90, 90): 'KeyD',
    (100, -100): 'Key*',
    (-110, 110): 'Key#'
}

# === Listener Setup ===
last_x, last_y = mouse.position
last_trigger_time = time.time()
cooldown = 0.2 # faster detection

print(" T9 Macro Listener Started — Press Ctrl+C to exit.")

try:
    while True:
        x, y = mouse.position
        dx = x - last_x
        dy = y - last_y

```

```

if abs(dx) > 5 or abs(dy) > 5:
    now = time.time()
    if now - last_trigger_time > cooldown:
        coords = (dx, dy)
        if coords in macro_map:
            keyname = macro_map[coords]
            print(f" Key Triggered: {keyname}")

        if t9.number_mode and keyname in NUMBER_MAP:
            t9.flush()
            print(f"Typed: {NUMBER_MAP[keyname]}")
            keyboard.type(NUMBER_MAP[keyname])
        elif keyname in T9_MAP:
            t9.update_key(keyname)
        else:
            t9.flush()
            send_special(keyname)

    last_trigger_time = now

# Flush if idle timeout reached
if t9.last_key and time.time() - t9.last_press_time > t9.delay:
    t9.flush()

last_x, last_y = x, y
time.sleep(0.01)

except KeyboardInterrupt:
    print("\nListener stopped.")

```

### Selfdefined Librarts:

ssd1306(see attachments)  
 hid\_services(see attachments)

### Appendix 15G – Bill of Materials (BOM)

#	Component	Qty	Price (USD)
1	ESP32-WROOM-32E Dev Board	1	\$12.00
2	MPU6050 Gyroscope	1	\$12.00
3	Tactile Push Buttons	2	\$1.50
4	Rotary Encoder (ZUDKSUY EK2045)	1	\$3.00
5	Coin Vibration Motor	2	\$3.00
6	Resistors, 1kΩ	4	\$0.20
7	Custom PCB	1	\$10.00
8	Jumper Wires & Headers	1	\$5.00
9	Power Bank	1	\$12.00
10	Fabric Glove (SURAWL)	1	\$3.00
11	OLED Display 0.96"	1	\$6.00

	(SSD1306)		
12	Piezo Buzzer	1	\$0.80
13	RGB LED	1	\$0.30

Total Cost: \$91.05

### **Appendix 15H – Team Contact Info**

Name	Contact Email	Phone Number
Jinhan Yang	jinhan.yang@email.com	226-503-6214
Reese Dow	reese.dow@email.com	289-338-6127