Ethan Liang and Selena Liu
EC 535 - Spring 2025
Professor Eshed Ohn-Bar
May 4, 2025

A Minecraft Pickaxe Controller - Technical Report

GitHub Repo: https://github.com/selenalliu/minecraft-pickaxe
YouTube video: Can be found on the GitHub readme

Abstract

Minecraft is a globally beloved sandbox game renowned for its creative freedom and iconic block-based mechanics, maintaining strong popularity for over a decade since its initial release. Due to its sandbox environment, first-person mechanics, and open-world design with gorgeous landscapes, Minecraft gameplay and user engagement can be greatly enhanced with gesture-based controls that mimic the physical actions of the in-game character. This project aims to develop a "pickaxe controller" that allows players to interact with the virtual Minecraft world through real-world motion. The controller utilizes an ESP32 microcontroller, MPU-6050 IMU sensor to detect gestures, and push buttons for control input, communicating with a computer as a Bluetooth HID device. The resulting system demonstrates both reliable gesture recognition and responsive, low-latency interaction, creating an intuitive and entertaining user experience, while also functioning effectively as a general purpose wireless mouse.

Introduction

Virtual reality (VR) and motion-based systems, such as Oculus Rift, have shown that physical interaction and body involvement can significantly increase player immersion, engagement, and enjoyment (Pallavicini & Pepe, 2020). However, many popular games, including Minecraft, still rely on traditional keyboard and mouse input or basic controller, non-motion-based input. This gap between the player's real-world motions and in-came actions presents an opportunity to create more immersive gaming experiences through gesture-based controls.

Minecraft is a sandbox game where mining blocks, building structures, fighting hostile creatures, and crafting items are key mechanics. These actions, which are reduced to simple mouse clicks and mouse holds, have great potential to become more meaningful when mapped to real-world gestures, especially since the game uses a first-person perspective camera angle where players can see the world through the perspective of the in-game character. Minecraft's realistic world generation and gorgeous landscapes only adds to the immersion. Inspired by the recently released "A Minecraft Movie" where the characters get transported from Earth into the Minecraft world, we aimed to build a physical controller in the shape of a pickaxe that lets players "mine" in real life. This system would not only map swings to mine/place blocks but would also control player perspective and vision panning.

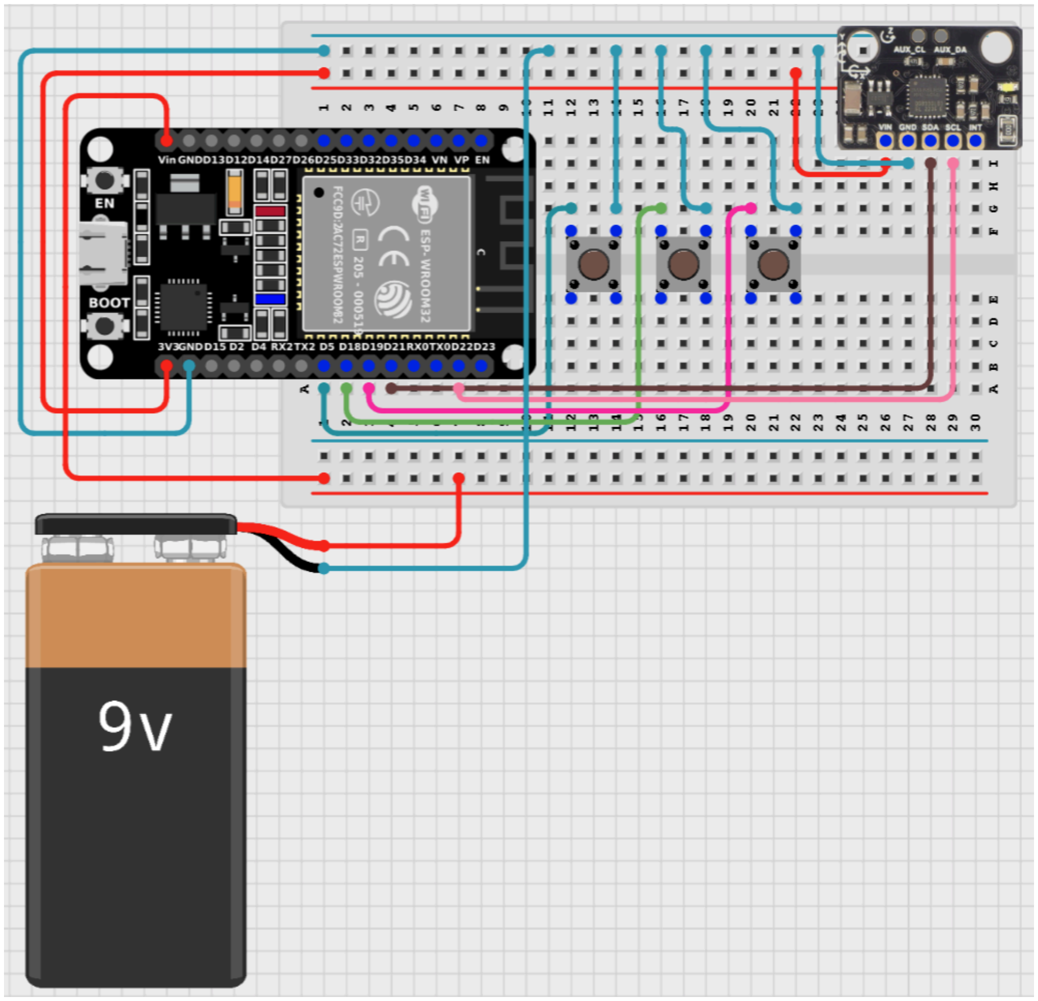
Minecraft controls require only keyboard and mouse inputs. More specifically, the only actions needed from the mouse are left clicks, right clicks, left holds, right holds, and movement/panning. The physical controller would take the place of the mouse and should be able to accomplish these mouse tasks.

Method

The pickaxe controller utilizes an ESP32-WROOM-32 as a microcontroller, an MPU-6050 which contains a 3-axis gyroscope and 3-axis accelerometer, and 3 push buttons for control input. A prototype circuit was constructed on a breadboard with jumper wires. Additionally, a 9V battery can be used instead of micro-USB for power, which allows the pickaxe to be swung freely without being tethered to a computer. Each of the 3 push buttons are connected to a GPIO pin configured with an internal pull-up resistor to keep the line high until a button press pulls it low. The ESP32's two I2C pins are connected to the MPU-6050.

Figure 1

Circuit Diagram for Pickaxe Controller System



EN
BOOT
Vin GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
ESP-WROOM32
AUX_CL AUX_DA
VIN GND SDA SCL INT
9v

The MPU-6050 employs a double buffer technique for sensor data readings over the I2C bus. New accelerometer and gyroscope readings are written into internal registers at the device's sample rate. User-facing read registers are exposed to the I2C bus and contain a latched value of the readings. Values from the internal registers are copied into the user-facing registers only whenever the I2C bus is idle, which avoids race conditions if sensor values are read mid-update.

Mouse actions are sent to the computer in USB packets, whose format is detailed in the USB HID mouse descriptor, which can be found online in the USB specification. The packet contains bits for 3 mouse buttons (left, middle, right), and 1 byte each for ΔX, ΔY, and ΔWheel values. These packets are placed in a FIFO queue where they are buffered to be sent. Program functions to click, press, release, and move the mouse simply set/clear the appropriate button bits and motion Δ bytes in the packets.

To meet the project's schedule, Arduino library abstractions were used instead of implementing I2C, USB, and Bluetooth drivers on the bare metal level. The code was compiled and flashed in Arduino IDE, after installing the following required libraries: MPU-6050 and ESP32 BLE Mouse. To compile and flash the sketch onto the board, ESP32 Dev Module was selected from the list of target boards, and the correct COM port to upload to was selected.

The pickaxe is simple enough to not require an operating system or task scheduling. Because there are not many events that require interrupts, real time features such as task priority or interrupt service routines are not needed. A simple cyclic executive superloop saves resources and is enough to accomplish the task.

The superloop does three things:

1) Check if the bluetooth is still connected. This is done simply by reading a variable from the bluetooth mouse class. If it's not connected, skip the current loop iteration to avoid unnecessary computes, memory operations, etc.
2) As long as no buttons are pressed, gyroscope values gy and gz are read (y and z axis rotation) to determine ΔX and ΔY for the mouse. ΔX and ΔY are calculated with the following equations:

   `SPEED = 200`

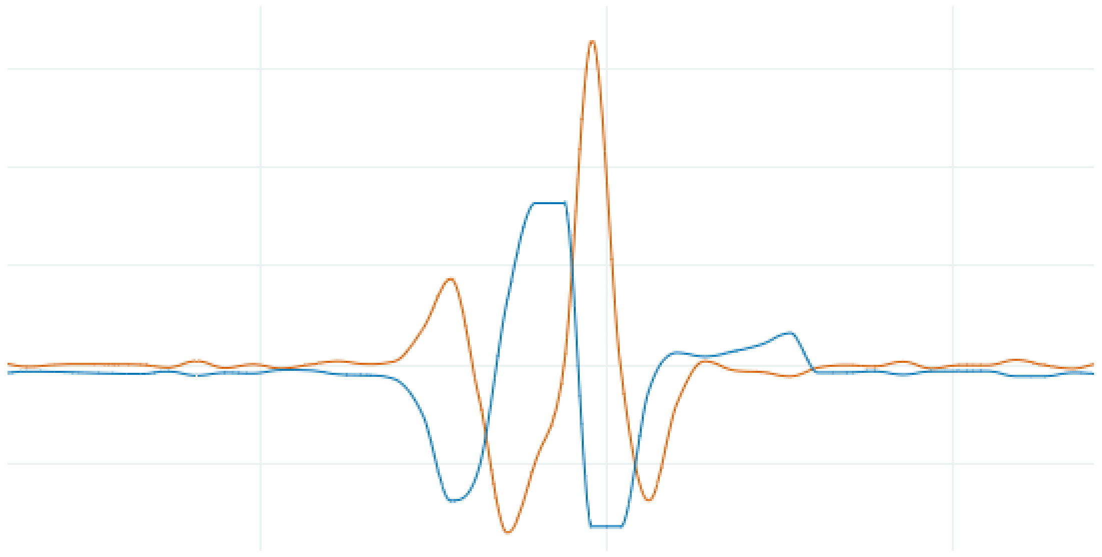
   $\Delta X =$ `gy * 1.3 / SPEED`

   $\Delta Y =$ `gz * 0.9 / SPEED`

   This adjustment is made to adjust panning sensitivity for a more comfortable user experience. It's important that the mouse does not pan while a mouse button is held because the swing would be registered as a mouse movement and the cursor would fly to the side of the screen.
3) Reads accelerometer value ax (acceleration along the x-axis) and calculates the difference between the current reading and previous one. The difference is then compared to a defined swing threshold to determine if the pickaxe was swung. Using the differential value instead of the raw acceleration measurement produces a sharper spike that more precisely reflects the onset of the swing. When a swing is detected, the state of the 3

button GPIO inputs are scanned to see if any of the buttons were pressed at the time of the swing.

Figure 2
The differential value (orange) results in a sharper spike that has a smaller duration and higher magnitude than the raw acceleration measurement (blue).



- Left/right clicks: If either the left or right button lines are asserted, the current timestamp is retrieved with `millis()`, which gets the number of milliseconds elapsed since the program began (overflows after about 50 days). A global variable is used to store the time of the previous click and if the difference is greater than a constant defined click cooldown interval, a left/right click will be registered. This prevents spurious double clicks and acts as a debouncing mechanism.
- Left/right holds: If the middle button was held concurrently with the left or right button, a mouse press is registered, and a software timer is initialized with a period defined in a constant. While the timer is still alive, the function checks for additional swings which refreshes the timer. A small blocking delay is used here to allow for the period between swings. When the timer expires, a mouse release is registered.

Results

Since our device was designed heavily for the purpose of creating an immersive user experience when playing Minecraft, our evaluations of the system focused on areas related to user experience, which included ease of use and input latency.

To produce quantitative metrics in the case of measuring the ease of use, we utilized a mouse accuracy test, which would help demonstrate the level of general usability of the device.

To perform our test, we used https://mouseaccuracy.com/, which is a tool that assesses mouse accuracy based on the number of circular targets you successfully click on. The targets spawn in random locations, and gradually decrease in size the longer they stay unclicked. The tool also has configuration settings that allow users to control the difficulty level of the test, along with the target size and duration of the test. For our tests, we set our difficulty level to "Easy", target size to "Large", and test duration to 30 seconds.

After completing the test, the tool returns results in regards to measuring mouse accuracy, including click accuracy, which measures the number of successful target hits over the total number of clicks performed, and statistics related to clicks, which includes the average number of clicks per second (CPS) performed over the test duration period.

Table 1

Mouse Accuracy Results - Measured Click Accuracy (Hits/Clicks) & Clicks Per Second (CPS)

Test #	Hits/Clicks	CPS
1	21/22	0.7
2	23/23	0.8
3	22/24	0.8
4	22/22	0.7
5	20/23	0.8
6	21/23	0.8
7	22/22	0.7
8	21/23	0.8
9	23/23	0.8
10	22/22	0.7
Average	217/227 = 95.6%	0.76

Across ten test rounds of the 30-second mouse accuracy test using our pickaxe controller, we achieved an average click accuracy of 95.6% (217 successful clicks out of 227 total attempts) and an average click rate of 0.76 clicks per second (CPS). These results demonstrate that the controller is capable of consistent and accurate mouse input. While the CPS is naturally lower than what can be achieved with a traditional mouse, the high accuracy rate reflects the device's reliability and responsiveness for casual to moderate use cases. It is also important to note that missed hits throughout these tests are not caused by miscalibration or other software or hardware-based mishaps, but rather can be attributed to user error. As such, these metrics can be used to effectively evaluate the usability of the device.

To test the input latency of our device, we used an online click speed test tool (https://clickspeedtest.com) and conducted five test runs, each lasting five seconds. During these tests, we looked for any noticeable delay between a physical swing on the controller and the resulting click registered on the screen. Across all trials, we observed no noticeable latency, and the device consistently responded in real time. This suggests that the controller is fully capable of supporting smooth Minecraft gameplay, without perceptible input lag that would detract from the user experience.

Given that the pickaxe is a controller designed for Minecraft gameplay, we tested its functionalities in-game as well. Breaking blocks (left hold), placing blocks (right click), eating food (right hold), and engaging in combat with mobs (left click) utilize every functionality of the controller and can be successfully performed with the controller, as seen in video and live demonstrations. In particular, the “Water Bucket MLG” or “Water Bucket Clutch” is a challenging maneuver where a player falling at terminal velocity avoids taking fall damage and dying by precisely placing a water bucket on the ground a split second before they would land, since falling on water in Minecraft results in no fall damage. While this technique requires extremely accurate timing abilities on the user’s end, performing it successfully would not be possible without a low latency device. The pickaxe controller consistently passed this demanding test, which proves its suitability for gameplay even in the most intense in-game moments.

Limitations and Future Work

The current implementation of mouse panning is based on acceleration, which is suboptimal for accurate cursor control. Ideally mouse panning would be based off of position instead, which has an intuitive and direct one-to-one relationship with cursor movement, making it more consistent and easier for users to get used to. However, integrating twice to calculate the position based on the measured acceleration is not feasible for the current setup because the MPU-6050 introduces too much noise to accurately integrate the acceleration into a smooth, usable position measurement. Using a better accelerometer and then controlling mouse panning based on position would provide an instant improvement in mouse panning accuracy and ease of use.

In terms of the device aesthetics and ergonomics, improvements can be made in terms of the physical build of the controller, where a 3D-printed pickaxe-shaped enclosure could be used to house electronics and achieve a more refined product, rather than the cardboard pickaxe cutout that we opted for in the sake of time. A plastic enclosure may also be more comfortable to hold in the hand, and allow for better and more ergonomic button placement, as opposed to buttons being placed in a line on a 2D plane (on a breadboard). The use of soldered connections would improve the durability and compactness of the circuitry and would also allow more flexibility in component placement. The buttons themselves could also be replaced by colored, arcade-style buttons with a larger surface area and more satisfying bottom-out, making for easier and more enjoyable usage.

Future enhancements to the project could add a scroll wheel or capacitive touch bar, which would add scrolling functionality into the controller. Currently, the controller is incapable of generating HID scroll events as mouse button holding doesn't allow movement of the mouse while any mouse button is held, which is a necessary condition as described in the method section. Scrolling is not required to play Minecraft but is a useful quality of life option, and would also make the device more practical as a general purpose mouse. Finally, the pickaxe is currently held with the dominant hand while the other hand uses a keyboard. Replacing the keyboard with a similar handheld module that mirrors common WASD and hot-key inputs would allow for untethered, unrestricted movement approaching a lightweight virtual reality setup.

Conclusion

Overall, a fully working product was created that is intuitive and enhances Minecraft gameplay by adding gesture controls which increases user engagement and immersion. Utilizing an ESP32 microcontroller, MPU-6050 IMU sensor, and Bluetooth HID communication, the controller effectively translates physical swings, arm movements, and button presses into low-latency mouse actions. Our evaluation using the mouse accuracy test, click test, and in-game demonstrations including demanding Minecraft maneuvers such as the "Water Bucket MLG" confirms the controller's efficacy as a Minecraft controller and general-purpose mouse.

While the controller meets its intended objectives, future enhancements can be made such as using a higher-quality accelerometer for position tracking, improving build quality with 3D-printed enclosures, and adding functionality with additional input mechanisms like a scroll wheel or capacitive touch bar. Ultimately, this prototype serves as a first step towards creating an intuitive and immersive Minecraft experience, providing players with an accessible and engaging alternative to traditional mouse and keyboard controls.

References

Pallavicini, Federica, and Alessandro Pepe. “Virtual Reality Games to Enhance Positive Emotions and Decrease Anxiety: A Pilot Study on the Role of Body Involvement (Preprint).” *JMIR Serious Games*, vol. 8, no. 2, 25 July 2019, https://doi.org/10.2196/15635.