Somatosensory Enhancement FPS Controller
Final Report

By

Team 43

Peilin He

Haochen Zhang

Beining Chen

Final Report for ECE 445, Senior Design, Spring 2023

TA: Yixuan Wang

May 2nd 2023

Project No. 43

Abstract

This project aims to enhance the gaming experience of FPS (first-person shooter) players by incorporating somatosensory feedback and control buttons specifically for shooting games. The controller is equipped with a gyroscope, or an motion sensor that detects the player's movement and translates them into on-screen movements, providing a more immersive and intuitive gameplay experience. In addition to the gyroscope, the controller also includes WASD directional buttons for movement control, a left click or trigger button, and a control button that moves the player's viewport. As a result, this FPS controller is able to let user aim and shoot 10 targets in 50 seconds; have each button works correctly and does not interfere with each other when pushed simultaneously; recoil function vibrates to at least 20 Hz vibration frequency and 0.005 g vibration force reached when trigger is pulled.

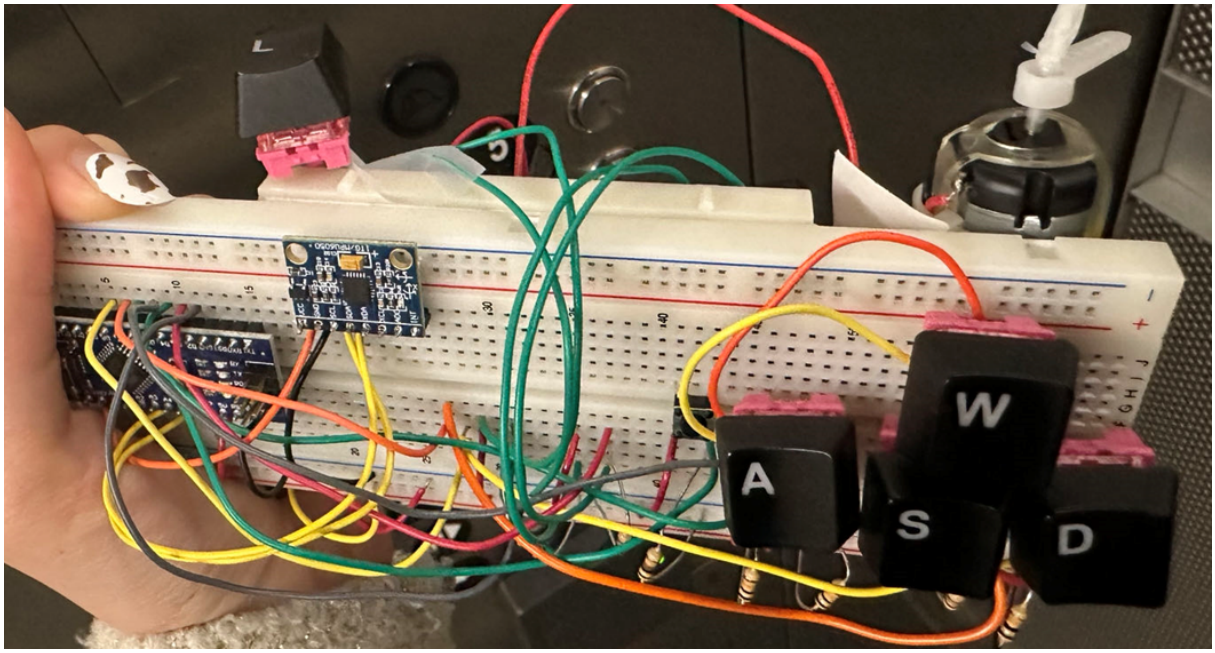


Figure 1 **Picture of final product of our Somatosensory Enhancement FPS Controller**

Contents

1. Introduction

1.1 Purpose and Functionality

The somatosensory enhancement FPS controller is designed to address some of the limitations of traditional controllers, particularly the use of a mouse for aiming movement. While a mouse is a precise tool for aiming, it requires a very flat and stable surface, and also requires the player to use one hand for aiming and the other for movement. This can result in less efficient and less natural game play. By incorporating somatosensory feedback, the FPS controller we implemented provides players with a more intuitive and immersive gameplay experience. The gyroscope allows a more natural and fluid movement control while the motor simulates the recoil and adds an extra layer of immersion.

In this document, we will first discuss the schematic and hardware subsystem design with block diagrams and verification for each block, including a MPU6050 gyroscope for motion sensing, control buttons, WASD directional buttons, and trigger, motor circuit for recoil simulation, and Atmega328P microcontroller for taking input and output to process; We will also go over the software algorithm that processes raw input into in-screen movement and aiming. Then, we will analyze the parts and labor cost of this project with a table of parts cost; Eventually, we will analyze our accomplishments, uncertainties we ran into when building the controller, ethical considerations, and recommendation for future work.

Overall, the somatosensory enhancement FPS controller successfully offers a solution to some of the limitations of traditional gaming controllers, or mouse and keyboard, providing players with a more engaging and enjoyable gameplay experience.

1.2 Subsystem Overview

In our design, we have 4 subsystems. Our power supply comes from the USB cord that connects the PC with our microcontroller on Arduino Nano, which also acts as a data bus that carries data to the PC. The vibration motor subsystem is implemented with LD293D Motor Driver IC. The microcontroller unit is soldered on Arduino nano which takes all button and gyroscope sensor data as inputs. The gyroscope subsystem contains an accelerometer that detects motion and outputs acceleration in X, Y, Z directions to the microcontroller.

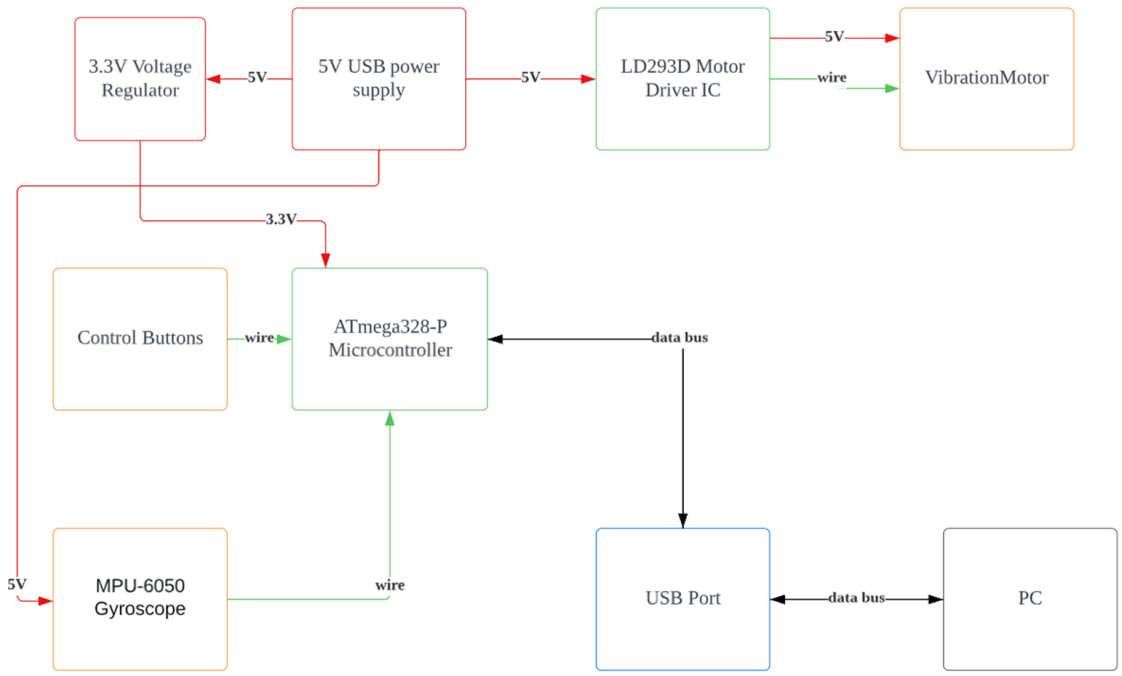


Figure 1 Block Diagram for FPS controller

1.3 High Level Requirements List

Our controller software works properly when

a. Aiming function works: screen position calculation algorithm reflects aiming point correctly on screen. Player is able to shoot 10 targets in 50 seconds
b. The aiming delay has to be less than 50ms
c. Each button works correctly with desired functions. Explicitly, when two buttons are pushed simultaneously, they can't interfere with each other

2 Design

To build the project we need a control unit to control the whole system and process any calculations in need. A gyroscope is used to detect the motion of users' hands and send that information to the control unit. The buttons work as input devices that give movement or trigger command to the microcontroller. The motor unit is controlled by the microcontroller to vibrate whenever a trigger is pressed. Beside these, we implemented a voltage regulator to provide 3.3V for components that are not driven by 5V. The USB port is used to make connections with the players' computers. The ISP unit is implemented for easier programming.

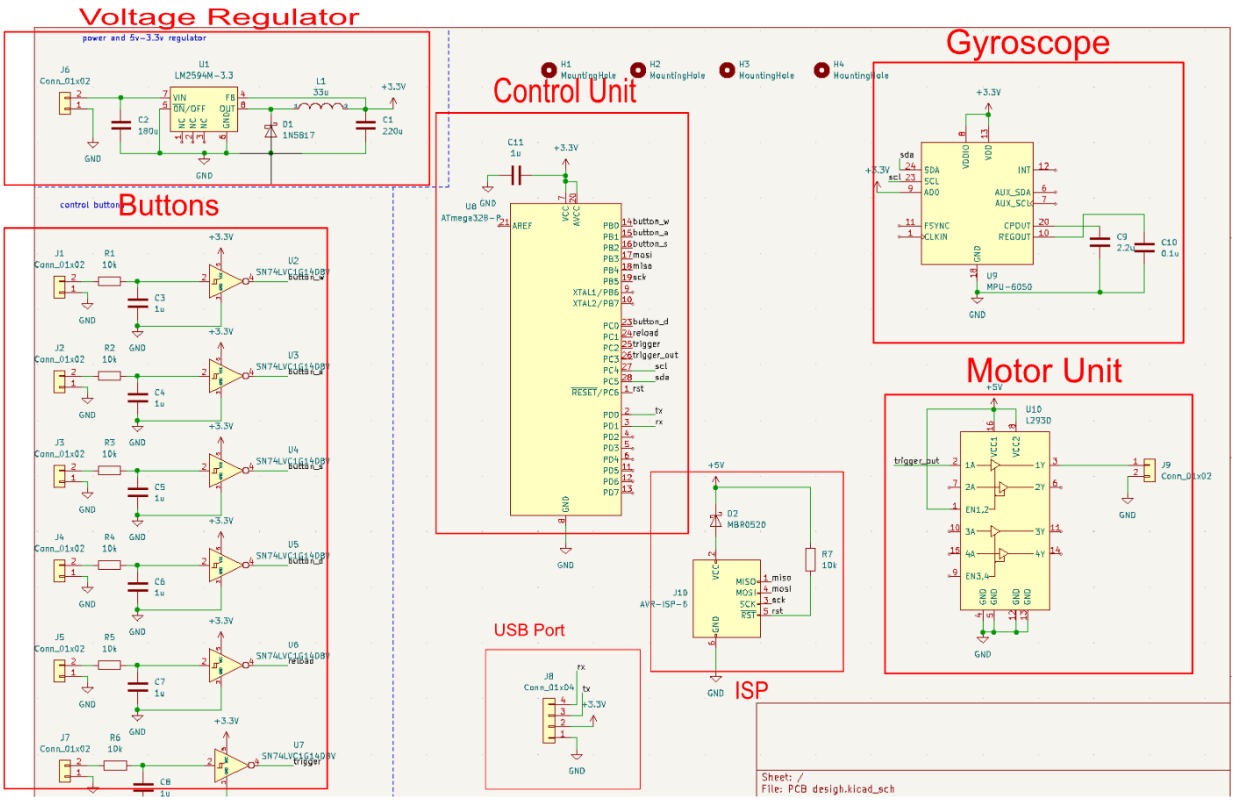


Figure 2 Circuit diagram for FPS controller

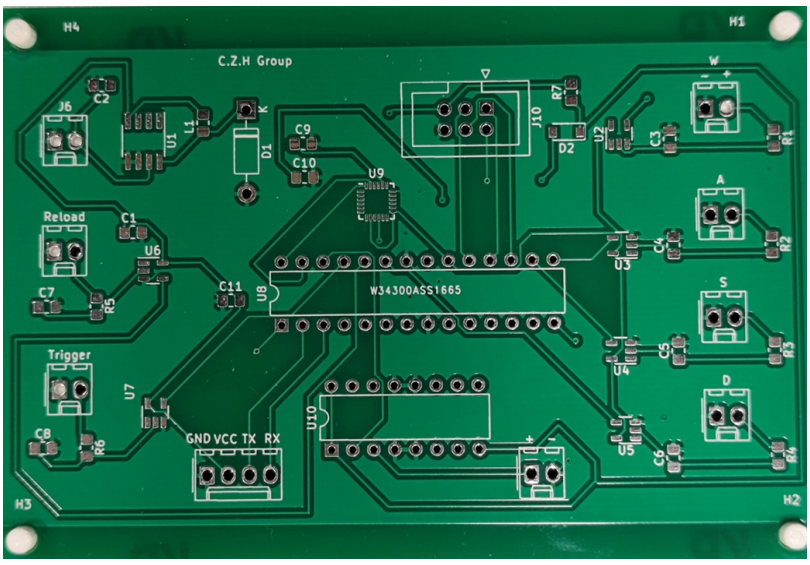


Figure 3 PCB for FPS controller

2.1 Control Unit Subsystem

This is the control unit subsystem consisting of a microcontroller which will incharge of controlling all the other subsystems. ATmega32-P microcontroller is shown in figure2. The

microcontroller is able to control the button & trigger which means if we press a key, the microcontroller needs to detect the keypress then it needs to send the signal to the computer. It also needs to process the signal received from the gyroscope and convert that into coordinates on screen. Figure4 shows the pins that are connected to the microcontroller. From the figure, the button and trigger pins are used to receive impulses from the buttons. The miso, mosi, and sck pins are used for ISP programming port. The sda and scl pin are used to control and receive data from the gyroscope. The tx and rx pins are used for USB data transport.

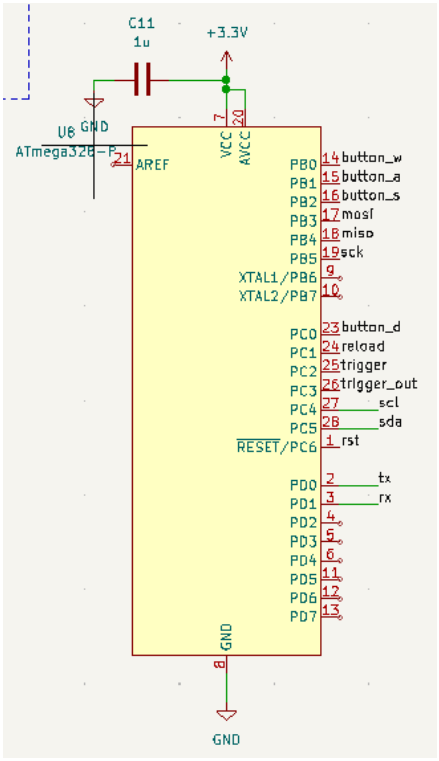


Figure 4 Circuit diagram and PCB of microcontroller ATmega328-P

2.2 Vibration Motor Subsystem

The main function of the vibration motor is that once you pull the trigger, the motor needs to vibrate and simulates a recoil force as a real gun in real life. This needs to be under the charge of the microcontroller; the microcontroller needs to read the signal when the trigger is pulled and send a signal or a current to the motor so that the motor can vibrate.

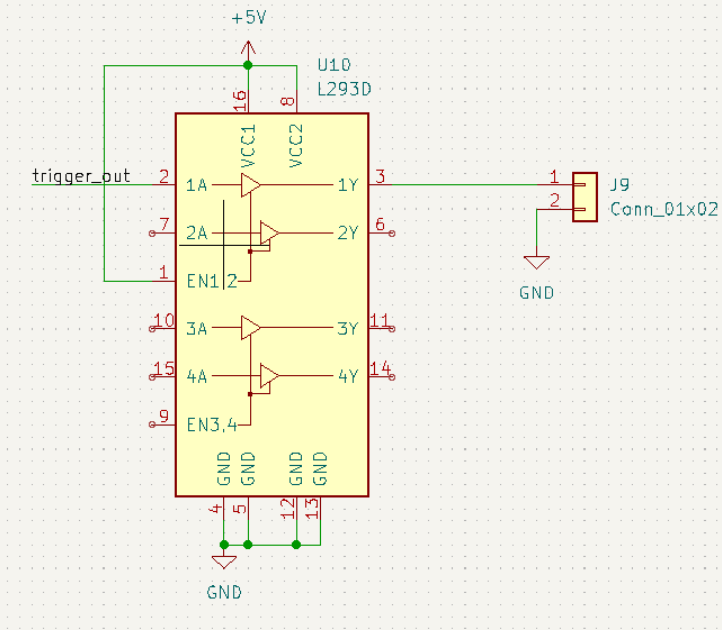


Figure 5 Circuit diagram and PCB of motor driver U10L293D

2.3 Button and Trigger Subsystem

In order to have a stable button subsystem, we used pull-down capacitors and smith trigger to eliminate glitches and double-click problems

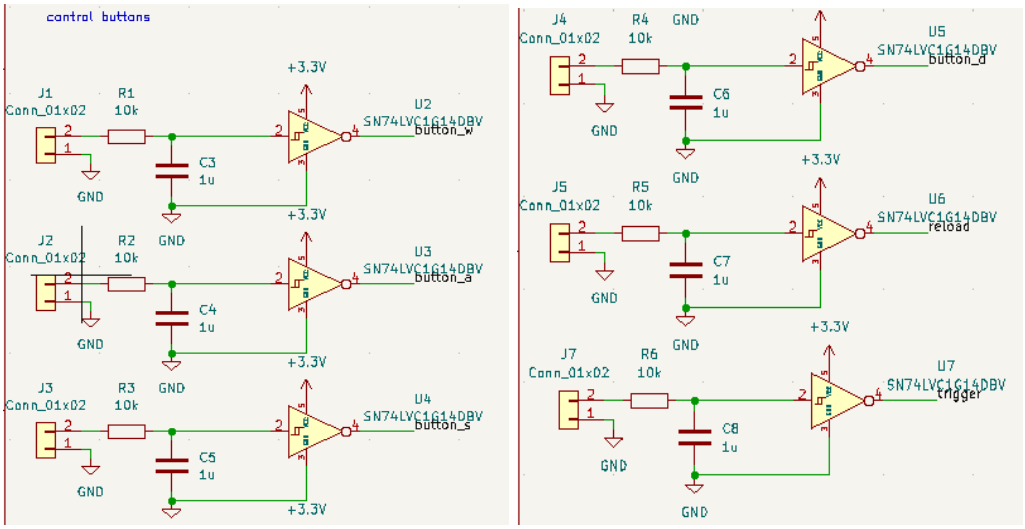


Figure 6 Circuit diagram of control buttons

2.4 Gyroscope Subsystem

MPU-6050 Gyroscope: The Gyroscope plays a role of motion detection. It means we need to use the Gyroscope to capture the data of how we move the model gun in real life and calculate with a customized algorithm to transport the data into the computer so that the aiming cursor in the

game will know the direction we are moving. The output parameters of Gyroscope are acceleration in x, y and z direction, and rotation in degree per second in x, y and z direction. Since the parameters are in terms of 3-dimensional space, and we want to project the movement of the controller on screen in terms of 2-dimensional space, we need to calculate acceleration only in x and y direction. The method we decided to use is calculating the exact distance the user rotated the gun muzzle using the degree given by gyroscope, and increment such a number to acceleration in x and y direction, and disregard acceleration of z direction. In order to achieve this goal, we decided to use Raspberry Pi to program this algorithm. With 2-dimensional x and y acceleration, we can then pass this vector data back to USB through microcontroller and eventually to PC.

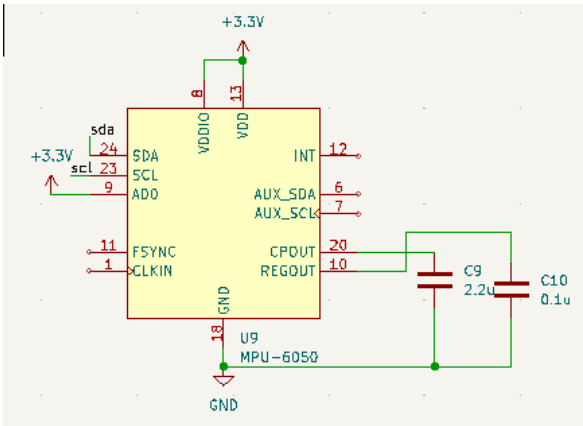


Figure 7 Circuit diagram of gyroscope

2.5 Power Supply Subsystem

Power Supply: A power supply is required for the model gun to function. We use a 5V USB power supply so no battery change is needed. Since some components such as the smith triggers need 3.3V power supply, a DC-to-DC voltage transformer is used in order to convert 5V to 3.3V.

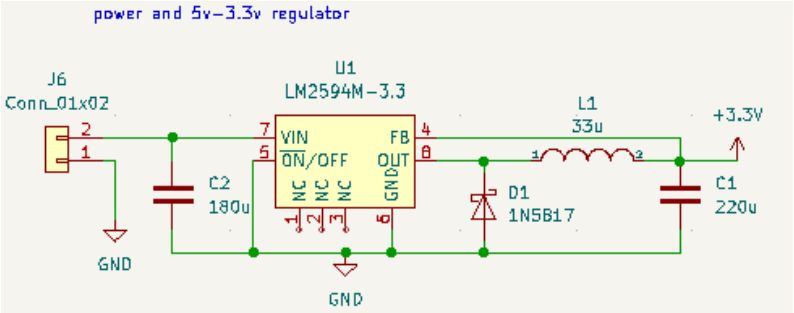


Figure 8 Circuit diagram of power regulator

2.6 Software Signal Processing

The software signal processing for the somatosensory enhancement FPS controller involves a series of steps that involve collecting, processing, and interpreting data from various sensors and buttons.

2.6.1 From Gyroscope to Computer

First, the MPU6050 gyroscope is used to sense acceleration in X, Y and Z directions. This raw data is then input into Arduino Nano which lets Atmega328P microcontroller feed data to the computer through a database from the USB cable. Arduino code then processes the data and prints out ASCII text to the serial port indicating which button was pressed or the gyroscope's motion. The following figure shows how we configure Arduino nano to connect with the PC and in Arduino code how we assign each of its input ports to corresponding variables.

sketch_apr17a.ino

```
#include <MPU6050_tockn.h>                    // Download and install this library https://github.com/tockn/MPU6050_tockn
#include <Wire.h>

#define LeftB     2                           // Left Button Pin
#define RightB    4                           // Right Button Pin
#define MouseB    3                           // Mouse Enable Button Pin
#define WKeyB     5                           // W Key Button Pin
#define AKeyB     6                           // A Key Button Pin
#define SKeyB     7                           // S Key Button Pin
#define DKeyB     8                           // D Key Button Pin

MPU6050 mpu6050(Wire);
int X,Y,Z;                                    // Data Variables for mouse co-ordinates
int OX,OY,OZ;                                 // Angle Variables for calucating gyroscope zero error

void setup() {

  Serial.begin(115200);                       // Initialise Serial communication
  Wire.begin();                               // Initialise I2C communication
  mpu6050.begin();                            // Initialise Gyro communication
  Serial.println("STRTM");                    // Identifier "STARTM" for start mouse
  mpu6050.calcGyroOffsets(true);              // Setting Gyro offsets
  mpu6050.update();                           // Command to calculate the sensor data before using the get command
  OX = mpu6050.getAngleX();                   // Getting angle X Y Z
  OY = mpu6050.getAngleY();
  OZ = mpu6050.getAngleZ();

  pinMode(LeftB,INPUT);                       // Setting Pinmode for all three buttons as INPUT
  pinMode(RightB,INPUT);
  pinMode(MouseB,INPUT);
  pinMode(WKeyB,INPUT);
  pinMode(AKeyB,INPUT);
  pinMode(SKeyB,INPUT);
  pinMode(DKeyB,INPUT);
```

Figure 9 Arduino Code 1

The following figure shows how we process raw X, Y, Z directions of acceleration from the gyroscope by inverting the sign and correcting the zero error.

sketch_apr17a.ino

```
  pinMode(LeftB,INPUT);                                  // Setting Pinmode for all three buttons as INPUT
  pinMode(RightB,INPUT);
  pinMode(MouseB,INPUT);
  pinMode(WKeyB,INPUT);
  pinMode(AKeyB,INPUT);
  pinMode(SKeyB,INPUT);
  pinMode(DKeyB,INPUT);

  if(OX < 0){                                            // Inverting the sign of all the three offset values for zero error correction
    OX = OX *(-1);}
  else{
    OX = (OX-OX)-OX;}

  if(OY < 0){
    OY = (OY *(-1));}
  else{
    OY = ((OY-OY)-OY)+10;}

  if(OZ < 0){
    OZ = OZ *(-1);}
  else{
    OZ = (OZ-OZ)-OZ;}
}

void loop() {
  mpu6050.update();
  X = OX + mpu6050.getAngleX();                          // Getting current angle for X Y Z and correcting the zero error
  Y = OY + mpu6050.getAngleY();
  Z = OZ + mpu6050.getAngleZ();

  if(digitalRead(MouseB) == HIGH){                       // Checks if Mouse Enabled Button is pushed
    delay(10);                                            // Mouse movement resolution delay
    Serial.println("DATAL,"+String(X/4)+','+String(Y/4)+','+String(Z/4)); // Sends corrected gyro data to the Serial Port with the identifier "DATAL"
  }
  if(digitalRead(LeftB) == HIGH){                        // Checks if Left Mouse Button is pushed
    delay(100);                                          // Debounce delay
    Serial.println("DATAB,L");                           // Sends "L" stating the left button is pressed with the identifier "DATAB"
  }
  if(digitalRead(RightB) == HIGH){                       // Checks if Right Enabled Button is pushed
    delay(100);                                          // Debounce delay
    Serial.println("DATAB,R");                           // Sends "R" stating the left button is pressed with the identifier "DATAB"
  }
  if(digitalRead(WKeyB) == HIGH){                       // Checks if W Enabled Button is pushed
```

Figure 10 Arduino Code 2

The following figure shows how python code that processes serial output results from Arduino Code shown previously. We used the pyserial library to connect to the serial port and interpret data by checking different identifiers. Each key or gyroscope movement output contains its own identifier that will be recognized by python and let the computer do corresponding key presses or mouse movement. pynput.mouse and pynput.keyboard are virtual mouse and keyboard classes that move the mouse and presses keys for you.

```
import serial
import pydirectinput
import time
from serial import Serial
from pynput.mouse import Button, Controller as mouseController
from pynput.keyboard import Key, Controller as keyController

mouse = mouseController()
keyboard= keyController()
ser = serial.Serial('COM3',baudrate = 115200)
# Setting Serial port number and baudrate
while 1:                                          # While loop to continuesly scan and read data from serial port and execu
    dump = ser.readline()                         # Reading Serial port
    dump = str(dump)                              # Converting byte data into string
    dump = dump[2:-5]                             # Cleaning up the raw data recieved from serial port
    data = dump.split(',')                        # Spliting up the data to individual items in a list. the first item bein
    print(data)
    if data[0] == 'DATAL':                        # Checking if the identifier is "DATAL" which the Arduino sends the data
        mouse.move(int(data[1]), int(data[2]))      # Moving the mouse by using the X and Y values after converting them in
    if data[0] == 'DATAB':                        # Checking if the identifier is "DATAB" which the Arduino sends the value
        if data[1] == 'L':                      # wIf the Left button is pressed
            mouse.press(Button.left)              # The corresponding button is pressed and released
            mouse.release(Button.left)
        if data[1] == 'R':                      # If the Right button is pressed
            mouse.press(Button.right)         # The corresponding button is pressed and released
            mouse.release(Button.right)
        if data[1] == 'W':                      # If the W button is pressed
            keyboard.press('w')         # The corresponding button is pressed and released
            keyboard.release('w')
        if data[1] == 'A':                      # If the A button is pressed
            keyboard.press('a')         # The corresponding button is pressed and released
            keyboard.release('a')
        if data[1] == 'S':                      # If the W button is pressed
            keyboard.press('s')         # The corresponding button is pressed and released
            keyboard.release('s')
        if data[1] == 'D':                      # If the W button is pressed
            keyboard.press('d')         # The corresponding button is pressed and released
            keyboard.release('d')
```

```
['========================================']
['Calculating gyro offsets']
['DO NOT MOVE MPU6050...']
['Done!']
['X : 9.13']
['Y : 5.09']
['Z : -1.30']
['Program will start after 3 seconds']
['========================================DATAL', '0', '0', '0']
['DATAL', '0', '0', '0']
['DATAL', '0', '0', '0']
['DATAL', '0', '0', '0']
```

Figure 11 Python Code

3. Design Verification

3.1 Control Unit Requirements & Verification

For the control unit, we have the WASD movement key which controls the character in game to move with the corresponding direction; we have the aiming function key which is used to move the screen position; we have the trigger button which is used to control the shooting and the vibration of the motor. This control unit along with the microcontroller is able to control all the buttons, which means if we press a button, the microcontroller needs to detect the key press and send the signal to the computer. This process, from the key press to the computer, usually takes time to finish; therefore, a potential issue related to delay needs to be considered and solved. Therefore, we have set the requirement as when buttons are pressed or pulled, the signal should be sent to the PC with a delay time no more than 50ms. In order to verify whether or not the requirement is met, we decided to use a 60Hz video to record the buttons and the screen in the same scope so that we could verify the number of frames from the time that button is pushed and the time a movement is witnessed on the screen. Besides the delay test, we have to verify that when two buttons are pressed simultaneously, each button should work independently without interference.

3.1.1 Buttons & Trigger Delay Test

The first delay test is the aiming function delay test. As mentioned above, we are going to use a 60Hz video to record the video of pushing the button and the screen in the same scope. Below is the figure shown:

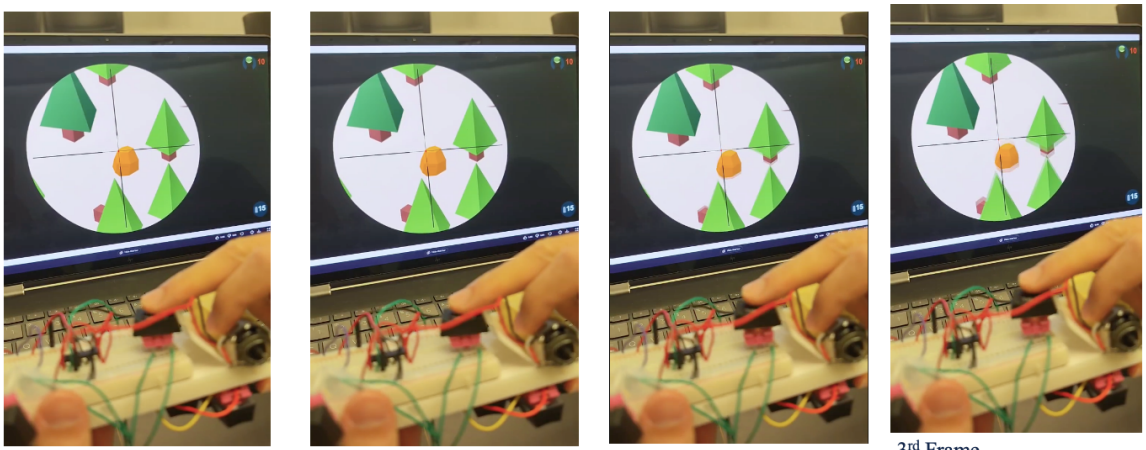
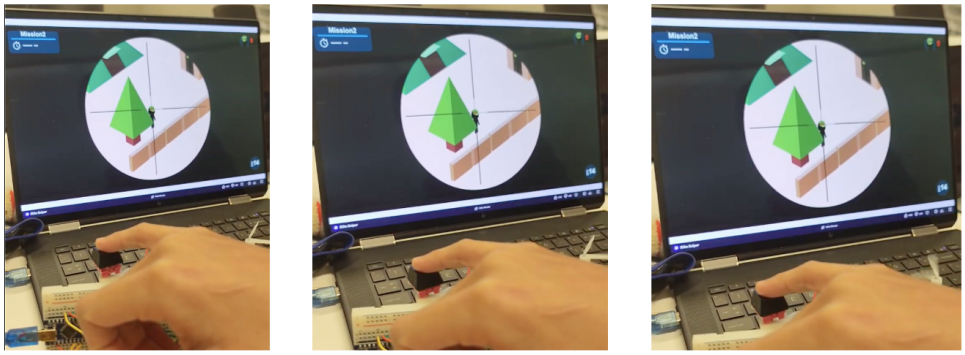


Figure 12 Frame screenshot 1

From the figure shown, after three frames of the time when the button is pushed, the movement of the screen position is captured; therefore, we could calculate how much time it takes. A 60Hz

video indicates that there are 60 frames in one second, so we could calculate that one frame takes 1/60 secs which is approximately 16.66ms; therefore, three frames take 49.98ms which is smaller than 50ms which means the requirement is met.

The second delay test is the motor delay test. We are going to use the same method to use a 60Hz video to record the video of pushing the button and the screen in the same scope. Below is the figure shown:



Button Pushed | 1st Frame | 2nd Frame: The Motor Spins

Figure 13 Frame screenshot 2

From the figure shown we can see that after two frames of the time when the button is pressed, the motor starts to spin. What's more, two frames take 33.32ms which is less than 50ms; therefore, the requirement is met.

3.1.2 No Interference Test

When two or even multiple buttons are pressed simultaneously, each button has to work independently without interference. One way to verify this is to actually play the game, for example if you press the forward movement button and the right movement button at the same time, you should expect the character in the game to move in a diagonal direction. We have uploaded a video reference to the website page, and from the video shown we have verified that when two or even multiple buttons are pressed simultaneously, each button works independently without interference.

3.2 Vibration Motor Requirements & Verification

The Motor Driver IC and the vibrator unit has the function to simulate a recoil force back to the user. This needs to be under the charge of the microcontroller because the microcontroller needs to read the signal when the trigger is pulled and send a signal or a current to the motor so that the motor knows when to vibrate. In order to simulate a real-life recoil force, we have set the requirement of 20Hz vibration frequency and 0.005g vibration force.

3.2.1 Continuous Vibration test

The test mainly ensures that the vibration is continuous when the trigger is held. Refer to the video uploaded on the website, the video verifies that when we held the trigger the motor was vibrating continuously.

3.2.2 Recoil Force test

In order to simulate the recoil force as real as the recoil force from the real gun, we have set the requirement to be 20Hz vibration frequency and 0.005g vibration force. We used a measurement app to measure the vibration frequency to be 43Hz and the vibration force to be 0.0035g. Below is the photo of the measurement:

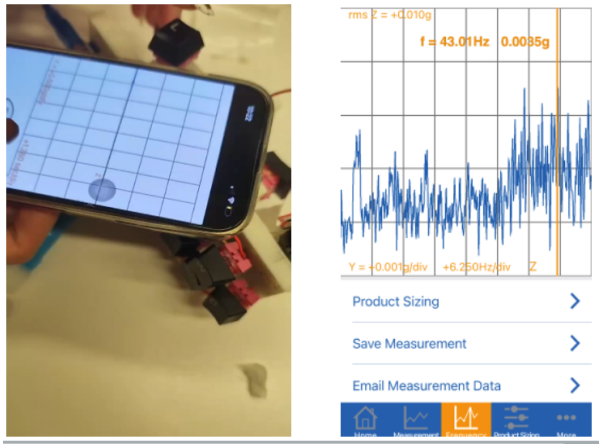


Figure 14 Vibration Test

3.3 Gyroscope Requirements & Verification

The component of the gyroscope is used to detect the movement direction of the controller. For example, if the controller is leaning rightward, then the gyroscope needs to detect the change of the movement and send the detected data to the computer so that the scope of the screen needs to be adjusted. Therefore, one requirement we have set is that the rotation direction of the character in game must be the same as the rotation direction of the gun controller. Referring to the video submitted on the course website, we have shown that the screen scope has changed with the same direction as how we move the controller.

4. Costs

4.1 Parts

Table 1 Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Push buttons	ECE machine shop	0	0	0
MPU-6050 Gyroscope	HiLetgo	1.40	4.20	4.20
mechanical switch	YUNZII	13.99	13.99	13.99
Lithium-Ion battery	YDL battery	11	11	11
VL120628H Linear Vibration Motor	Digi-key	6.42	6.42	6.42
Resistors (100-10k Ohm)	ECE machine shop	0	0	0
Capacitors (10n - 100u F)	ECE machine shop	0	0	0
LM2594M switching regulator	Digi-key	3	3	6
1N2594D Diode	Digi-key	0.43	0.43	1.72
1465 Adapter Mini AVR ISP	Digi-key	0.95	0.95	1.9
LM3294M-3.3 Switching Regulator	Digi-key	3	3	6
ATmega328P	Digi-key	5.9	5.9	11.8
L392D Bipolar motor Driver	Digi-key	8.52	8.52	17.04
MBR0520-TP diode	Digi-key	0.2	0.2	0.4
SN74LVC1G14DBVR smith trigger	Digi-key	0.35	0.35	3.5
410-MPU6050	Digi-key	9.11	9.11	18.22
Adruino Nano	LAFVIN	10	7	21
Total				**123.19**

4.2 Labor

Labor: ($44.63/hour) x 2.5 months x 50 hrs = $5,578.75 per group member

Total Labor: $5,578.75 x 3 = $16,736.3

We can estimate each group member will devote 50 hours per month into this project. Average starting salary for UIUC Electrical Engineering major is around $80,296, and average starting salary for UIUC Computer Engineering major is around $105,352. We will take an average of both salaries of $92,824, and around 14 days of holidays. We can calculate that the average hourly wage would be $44.63.

5. Conclusion

In the end, we were able to successfully complete the implementation of somatosensory enhancement FPS controllers. Our project meets all 3 high-level requirements previously outlined, and has a very nice user experience. During the demonstration, we successfully connected and ran our device with a laptop and used it to play a few different FPS games and showcased all of our requirements and verifications. We have all the parts ready and functional as we wished, and motion of our controller reflects on-screen movement in real time.

During the process of designing and building our controller, we discovered multiple issues that caused us to take alternative plans. For example, hardware parts arrived later than expected, parts' sizes are not compatible with the pcb, and the arduino cannot connect with python code. However, we resolved the issue eventually with all the resources and material we could find. If we did this project again, we would try to get an early start with pcb design, and possibly get different drafts of pcb and eventually have a functional final pcb to implement. When we are playing with our product, we feel the happiness it brings to us, and we are genuinely proud of this product. We believe our product can improve most users' experience on FPS games.

5.1 Accomplishments

We believe this project was an overall success. All high-level requirements and subsystem requirements were verified by the demonstration. We successfully accomplished less than 50ms of motor delay when button pushed, and less than 50ms of trigger delay from trigger pulled to on-screen view point moved. We achieved at least 20 Hz of motor vibration frequency and 0.005 g vibration force to simulate recoil when the trigger is pulled. Each button works correctly with desired functions, even when two are pressed together, they cannot interfere with each other's functionality. We also successfully set up connection from our Atmega328P hardware to Arduino code to process raw data, and also processed by python code to eventually move our mouse and presses keyboard.

5.2 Uncertainties

Like any project, the somatosensory FPS controller has many uncertainties that arise during its development. The first uncertainty is the design of the printed circuit board (PCB). Designing a PCB can be a complex process, requiring careful consideration of various factors

such as component placement, power supply, and signal routing, especially when none of the team members had previous experience with it. Design errors or oversights can result in malfunctioning or unreliable hardware, requiring additional design revisions. In our case, we took too much time to consult TAs for help in schematic design, and our first PCB design, which ended up with not enough time for design revisions and hardware component changes.

Another uncertainty is the appearance of the final product. Our original design of the gun-shaped controller raises several ethical concerns which we will discuss in the following section. Since our controller design might create an unsafe environment in university, we had to come up with a new design even when our schematic is already set up, Pcb already designed and placement of buttons already set. It was difficult to envision a new design before a physical prototype was created. There are design challenges that arise during the fabrication process, such as difficulties in fitting components within the desired dimensions or challenges in finding materials that meet both functional and aesthetic requirements.

5.3 Ethical considerations

Our original design of the gun-shaped controller raises several ethical concerns and was brought to attention during our first design review session. After our discussion, we believe that the presence of a gun-shaped device in a school could be perceived as promoting violence and can potentially cause fear or panic among students, faculty, and parents. Furthermore, creating a gun-shaped device may also possibly trigger anxiety or trauma for students creating unsafe environments. From the points we discussed, gun-shaped controller are generally ethically problematic, which violates 7.8 IEEE code of ethics that " to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment", we decided to change our device's appearance to a more typical controller outlook.

In the spirit of 7.8 IEEE code of Ethics I5 " to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors", we will thoroughly test our design with other team-members, TA, our family and friends of teammates. We will accept and value any feedback they provide and make improvements on our design.

Our controller is designed to be hand-held, and we must ensure the material of the device is completely safe to hold and grip for a long time with warm body temperature and some degree of sweat. However, our controller is not designed to be water-proof and should avoid any spills or wet environment.

5.4 Future work

There are several potential avenues for future development of our device. One area of focus could be the incorporation of haptic feedback through the use of haptic cables. With the immediate tactile feedback to the players, our device will reach a new level of precision, speed

and realism. Another area that we could work on in the future could be implementing a fully functional and optimized PCB. This could involve buying hardware parts of correct sizes, refining the current design to improve its reliability, or adding additional features such as bluetooth.

Creating a user-friendly software interface could also be a valuable addition to the controller. This could include developing a desktop and mobile application to adjust controller settings or turn the controller on and off, allowing for a more personalized and intuitive gameplay experience.

Finally, incorporating bluetooth connectivity could allow for wireless operation of the controller, freeing the player from the constraints of using USB physical wire. This could potentially open up new gameplay opportunities, such as using the controller to play virtual reality games or other immersive environment games.

References

[1] "7.8 IEEE Code of Ethics." *IEEE Section 7 - Professional Activities*, https://www.ieee.org/about/corporate/governance/p7-8.html#:~:text=To%20treat%20all%20persons%20fairly,and%20to%20avoid%20injuring%20others.

[2] Grainger Engineering Office of Marketing and Communications. "Salary Averages." *Electrical & Computer Engineering | UIUC*, https://ece.illinois.edu/admissions/why-ece/salary-averages.

[3] Last Minute Engineers. "In-Depth: Interface MPU6050 Accelerometer & Gyroscope Sensor with Arduino." *Last Minute Engineers*, Last Minute Engineers, 29 Oct. 2022, https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/.

[4] Microchip Technology Inc., "Atmel-7810-Automotive-Microcontrollers-ATmega328P Datasheet," Microchip.com. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Accessed: 03-May-2023].

[5] Q. Viper, "Air Mouse: Controlling mouse with gestures in Air," *Quassarian Viper*, 12-Jul-2021. [Online]. Available: https://q-viper.github.io/2021/07/12/air-mouse-control-mouse-with-gestures/. [Accessed: 03-May-2023].

[6] R. Candido, "Arduino with python: How to get started," *Real Python*, 25-Jan-2023. [Online]. Available: https://realpython.com/arduino-python/. [Accessed: 03-May-2023].

[7] Texas Instruments, "LM2594 SIMPLE SWITCHER Power Converter 150 kHz 0.5A Step-Down Voltage Regulator Datasheet," TI.com. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2594.pdf. [Accessed: 03-May-2023].

Appendix ARequirement and Verification Table

Table 2 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Button & Trigger Requirements a. When the button & trigger are pressed or pulled, the signal should be sent to the microcontroller and the microcontroller will then input the signal to the PC through the USB port with a delay time no more than 50ms. b. Each button or trigger should work independently without interference.	1. Button & Trigger Verifications a. Use 60Hz video to record the trigger and the motor in the same screen. Verify that the number of frame from the time the trigger is pulled and the time the motor started to vibrate is less than 5 frames(50ms) b. Verify the function of each button while some other buttons or triggers are held.	**Y**
2. Linear Vibration Requirements a. The motor must vibrate immediately after the trigger is pulled. The delay time needs to be under 50ms. b. The motor has to continue vibrating while the trigger is held.	2. Linear Vibration Verifications a. Use 60Hz video to record the trigger and the motor in the same screen. Verify that the number of frame from the time the trigger is pulled and the time the motor started to vibrate is less than 5 frames(50ms) b. Record the sound of the motor while the trigger is held. This can verify that the motor is continuously vibrating while the trigger is held.	**Y**
3. Gyroscope Requirements a. The aiming delay on screen has to be less	3. Gyroscope Verifications a. Use 60Hz video to record the trigger and the motor	Y

than 50ms. b. The rotation direction of the character in game must be the same as the rotation direction of the gun controller.	in the same screen. Verify that the number of frame from the time the trigger is pulled and the time the motor started to vibrate is less than 5 frames(50ms) b. Face the game character to a wall. Make one shoot. Rotate the controller and make another shoot. Check and see whether the rotation is in the same direction of the controller.	