Robot Controller through Gestures

ECE 445 Final Report

Eric Zhou(yirui2), Guang Yin(guangy2), Haoduo Yan(haoduoy2)

Group 41- Spring 2022

TA: Hanyin Shao

Abstract

The project proposes a different version of robot controllers other than the traditional joystick, a controller that generates control commands based on the gesture of the user. Instead of pressing buttons and moving joysticks in conventional controllers, the project allows users to control the robot using gestures by wearing a glove with several inertial measurement units (IMU). In this way, the controlling actions can be done in a fine-grind version and also enables the user to get a more intuitive controlling experience.

Contents

1. Introduction

1.1 Problem and Solution Overview

Traditionally, different robots are controlled by specialized controllers from different companies, and it takes time to learn how to use them smoothly and naturally. We propose a gesture control system that builds upon the system, making the process of controlling a robot simple and fun. We plan to design a robot controller which can recognize human gestures and send corresponding commands to the robot. Since two of our team members are in the Illini RoboMaster group, we decided to build our product for the robot borrowed from this group.

1.2 Visual Aid

Fig 1.2 Visual aid

The project consists of a glove and a software program. 6 IMUs are mounted on the glove for gesture recognition, connected to them is a control board mounted on the wristband. It provides feedback and communicates with the robot wirelessly via Bluetooth. The program resides on an intermediate PC, responsible for gesture recording and analysis.

1.3 Advantages

A traditional robot controller mainly uses physical buttons and joysticks to control the robot. We believe our glove will have the following advantages over it:

- Ours takes one hand to operate, while the traditional one takes two.
- Users can customize gestures to be used, making it easier to operate.
- The IMUs we used are more sensitive in orientation.
- It is more suitable for those with trouble moving certain fingers.

1.4 High-level Requirements

- The position and orientation of the user's fingers and arms can be collected through IMUs mounted on the user.
- Users' body movements and gestures can be translated to robot control actions respectively.
- Users can get feedback from the user through a feedback system mounted on the user.

2. Design

2.1 Block Diagram

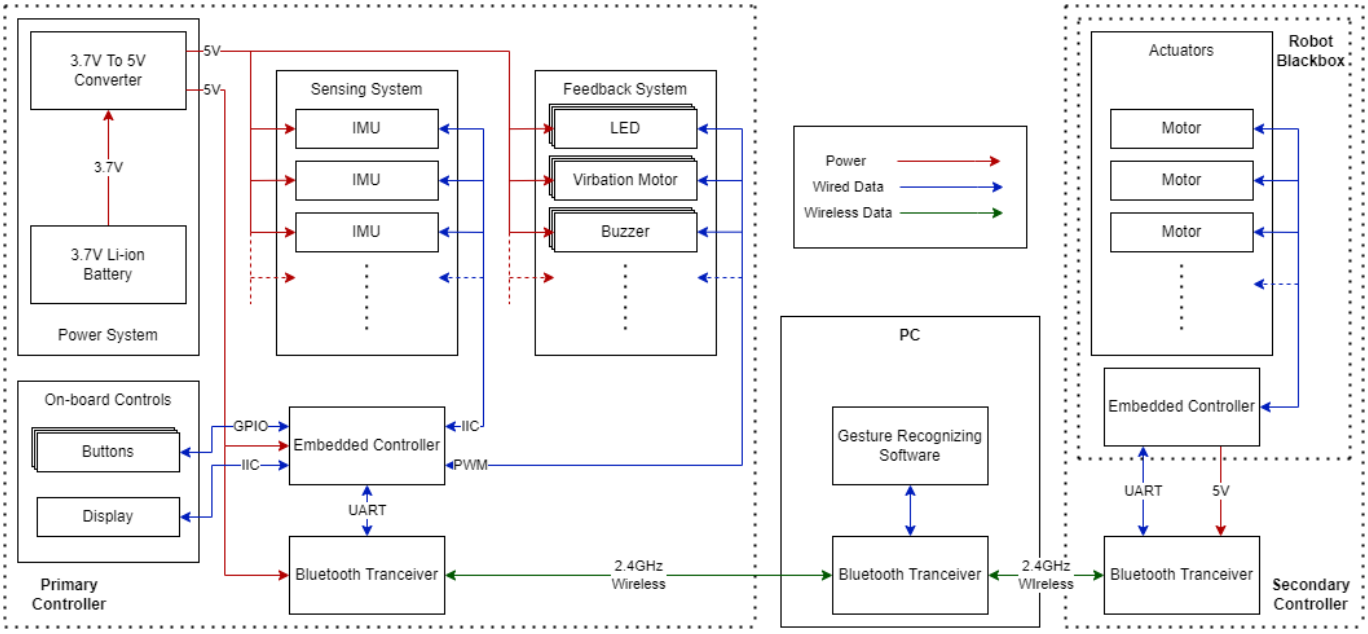

Fig 2.1 Block Diagram

2.1.1 Subsystem Overview and Design Procedure

2.1.1.1 Human Positioning System

The system is mainly used to measure the position and orientation of the fingers and arms of the user and then output these data to the next subsystem. It consists of sensors for orientation measurement, a microcontroller, and a data transmission module. They all have various feasible solutions, and these solutions will be compared below. The orientation of each finger can be divided into the flex of three joints, so resistance of flex sensors indicates the degree of flex for each finger, just like the power glove used by Nintendo Entertainment System. The other solution is to mount IMUs on each finger and measure the Euler angles of fingertips in 3D space.

	Pros	Cons
Flex sensor	1. Cheap ($10.95 each). 2. Easy to use (only need to measure its resistance).	1. Might not be accurate enough for our project, especially when two gestures are similar. 2. Users can feel the resistive force from the flex sensor, which may downgrade the user experience.
Inertial measurement unit	1. Extremely accurate and can directly output Euler angles for gesture recognition. 2. Lightweight and will not interfere with user motions	1. Expensive ($31.9 each) 2. Need calibration after several times of measurements. 3. Output depends on the magnetic field, which may change in different environments.

Flex sensors are easy to use and achieve our target to some degree. However, since IMU can output measurement results that are much more accurate, which is essential for the Gesture Control System that will be explained later, IMU are chosen as our primary sensor.

For the microcontroller unit, originally STM32F4 series are considered. However, while designing our first PCB board, we found out that STM32F4 needs too many peripheral components to operate and it is not easy to program and write code for STM32F4 even though it is extremely powerful. The most inconvenient aspect is that an external Bluetooth module is needed, and designing a Bluetooth module by ourselves is too complicated. As a result, EPS32 is chosen to be an alternative solution.

	Pros	Cons
STM32F4	1. Extremely powerful 2. Professional 3. Nearly all communication protocols are supported	1. Expensive 2. Needs external modules for Bluetooth communication 3. Difficult to write code and program it 4. Has too many pins which make hand soldering nearly impossible
ESP32	1. Bluetooth and WiFf module built-in 2. Cheap 3. Support MicroPython library, so it is much easier to write code and debug 4. Has a small pin number, but pins are connected to the I/O matrix, which enhances flexibility.	1. Computation capability is much less than STM32F4

Since our device and gesture recognition algorithm need low delay and only need to work in a short distance, Bluetooth is preferred for our project compared with other wireless communications like WiFi.

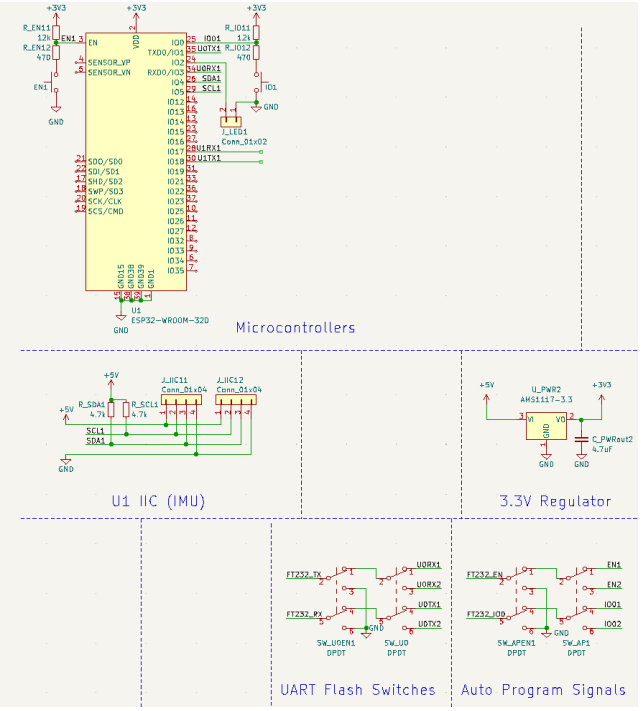

Fig 2.2 Schematics of Human Positioning System

2.1.1.2 Gesture Control System

This system is the software on a PC that will recognize the gestures of the user using the data retrieved by the Human Positioning System. It will then translate the gesture into the corresponding command and send it to the robot through Bluetooth.

For GCS, the most difficult and major task is to design an algorithm to recognize and differentiate gestures. There are two methods that come to our mind: the L2 algorithm and the AI model. The key idea behind the L2 algorithm is simple: compare the current gesture with each gesture in the database, calculate the L2 error between each pair, and output the gesture with the lowest L2 error. The AI model is also suitable and it may work even better than the L2 algorithm. For example, the L2 algorithm may find it hard to recognize two similar gestures, but AI may find the key feature to distinguish them. Also, since the gesture will continue to be the same in a short time, the Markov chain can be extremely useful to increase accuracy in AI models.

	Pros	Cons
L2 algorithm	1. Easy to implement 2. Can manually adjust the sensitivity, allowing an intermediate buffer state. 3. New gestures can be added to the database easily	1. Assumes that user gestures are static, and only use information about orientation angles
AI model	1. Can utilize dynamics like acceleration to recognize gestures 2. By using the Markov chain, the previous prediction can also help the next prediction 3. Universality. In theory, it should perform better than L2 algorithm	1. Hard to train and deploy 2. Performance is greatly influenced by the quality of dataset 3. The prediction labels are fixed, so it would be difficult to add a new gesture 4. Collecting datasets may consume too much time

In the original schedule, both methods are planned to be implemented and tested. But because the L2 algorithm works perfectly and AI model deployment takes too long to be developed and used, the L2 algorithm is chosen to be the solution for gesture recognition in the end.
There three popular orientation representation systems: Euler angles, rotation matrix, and Quaternions. Euler angles are intuitive and Human-readable but suffer from singularities. Rotation matrix works fine in theory, but it needs 9 entries to store orientation, and needs more computing power compared with Quaternions which only have 4 elements for one orientation in 3D space. It is also easy to compute geodesic differences between two groups of Quaternions. So, Quaternions are used as our primary data structure to store IMU orientation information, but Euler angles are also used for debugging.

2.1.1.3 Robot Feedback System

The system will read the warning messages from the robot's own controller and translate these warnings to actions of the feedback system. It will then command the feedback systems on the user to respond accordingly.

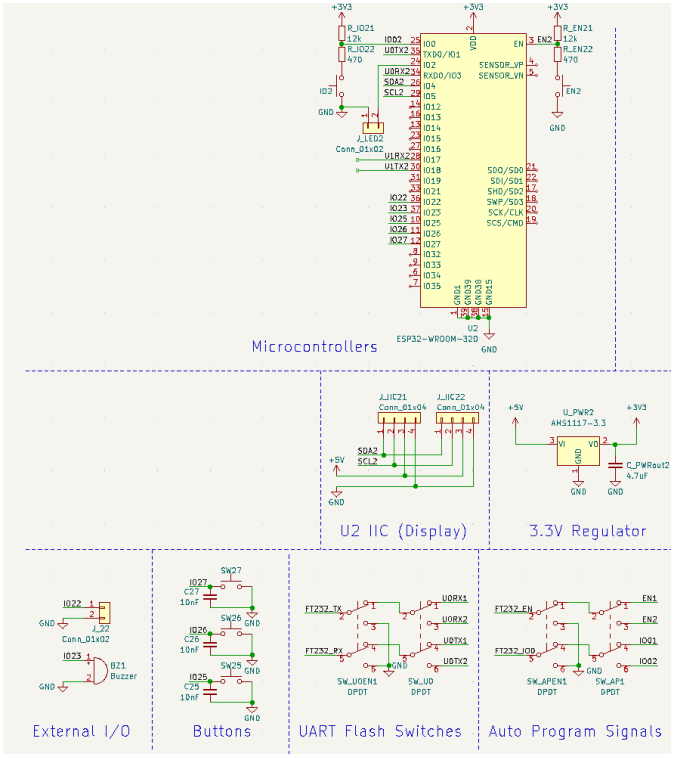

Fig 2.3 Schematics of Robot Feedback System

2.2 Design Details

2.2.1 Primary Controller

The primary controller consists of a power source, an embedded processor, multiple IMUs, multiple actuators, and a transceiver.

The power source will be using a 3.7V Li-ion battery and a 3.7V to 5V voltage converter to power the whole system.

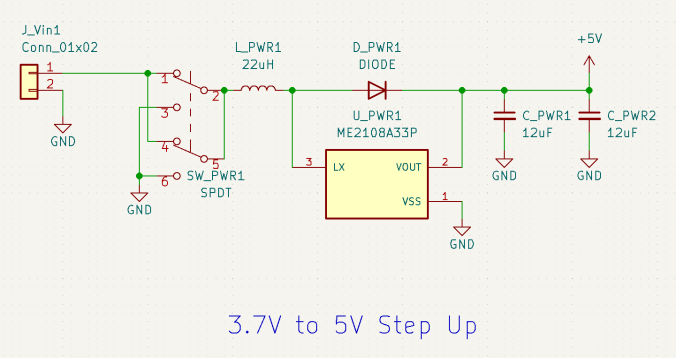

Fig 2.4 3.7V to 5V Step-Up Circuit

The sensing system will have multiple (at least 4 and at most 8) IMUs connected to the embedded controller through IIC communication. The data of the IMUs should be read from the IMUs at least 47Hz (detailed calculations are included in the 2.4.2 IIC communication bandwidth section).

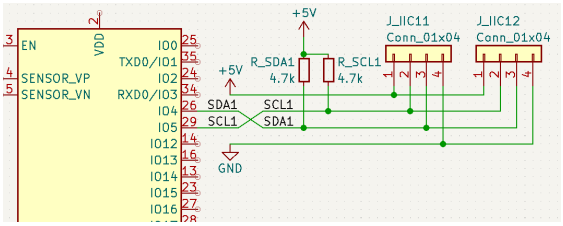

Fig 2.5 IIC Communication Socket

The feedback system will have multiple components that help to display messages in some ways. It includes visual feedback (LED and display), hearing feedback (buzzer), and body feedback (vibration motor). These commands are sent to the display using IIC protocol, and to other devices with PWM or analog output pin.

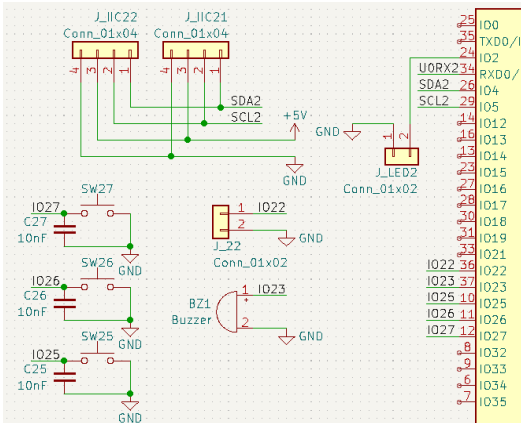

Fig 2.6 Feedback System

The system should also be able to send the angles read/calculated via a 2.4GHz Bluetooth link. It can help identify the position and orientation of the fingers and arm of the user.

2.2.2 Gesture Recognizing Software

The program will take in the Quaternions for each IMU broadcasted by the Bluetooth module on the primary controller, compare the current gesture data with those in the database by using the L2 algorithm that we developed, predict the best match gesture, and output the corresponding command to the robot through a 2.4 GHz Bluetooth datalink.

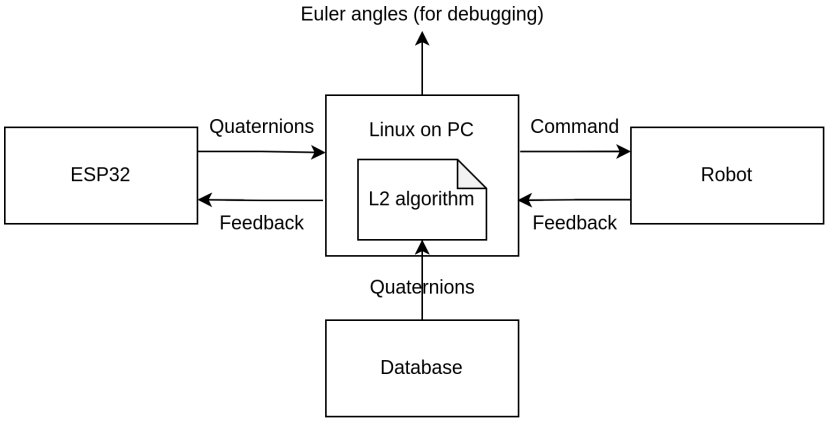

Fig 2.7 General Data Flow

Note that GCS on PC also outputs Euler angles but only for debugging purposes. The Python code to convert Quaternions to Euler angles is included in Appendix B.

The key part of gesture recognition is the L2 algorithm, and its visualization is below.

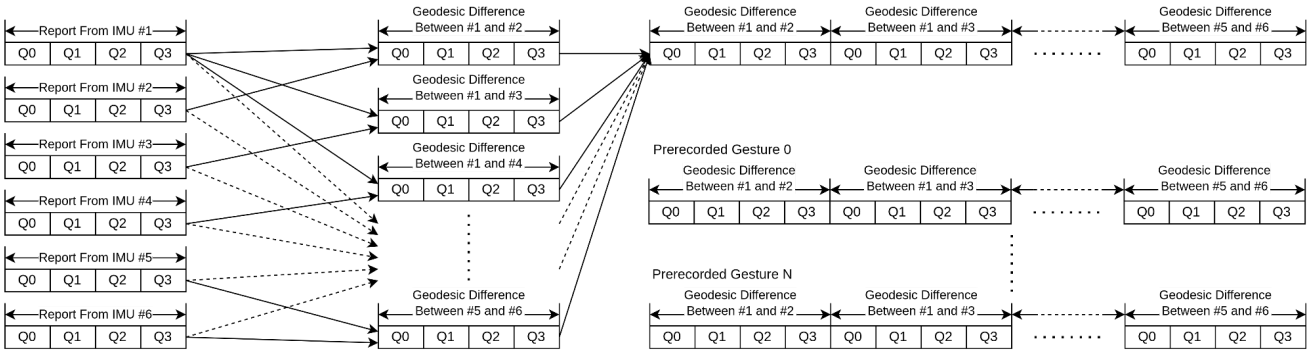

Fig 2.8 L2 Algorithm Visualization

The L2 algorithm can be divided into 5 steps.

1. Get 6 groups of quaternions for each IMU mounted on the primary controller.
2. 15 groups of quaternions are gathered by calculating the geodesic distances between every possible pair of the quaternion groups.
3. Concatenate these 15 geodesic differences into one feature vector that contains all the information needed for user gestures.
4. Calculate the L2 error between the current user gesture feature vector and each gesture feature vector in the prerecorded database. These L2 errors represent the similarity between vectors.
5. Predicted gesture is chosen based on two criteria. Firstly, the L2 error between the current gesture and the predicted gesture needs to be smaller than a hyperparameter threshold; and secondly, if multiple gestures meet the requirement at the same time, the gesture in the database with the lowest L2 error will be chosen.
6. After this arbitrating process, if no gesture is chosen because all the L2 errors are above the threshold, the algorithm will output no prediction.

To further explain the logic behind the algorithm, an example is shown below.

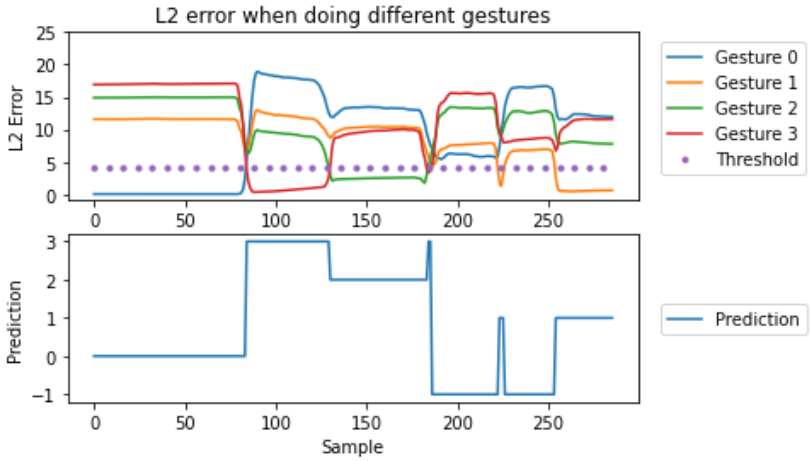

Fig 2.9 An Example of L2 Algorithm

In the beginning, between samples 0 and 80, Gesture 0 has the lowest L2 error which is also below the threshold, so gesture 0 will be chosen as our prediction. Then, in sequence, we predict gesture 3 and gesture 2. In the period

that covers samples 170 to 220, even though Gesture 0 has the lowest L2 error, its L2 error is above the threshold, so the algorithm judges that the user is doing a gesture that is not in the database and outputs gesture -1, indicating that there is no matching gesture found in the database.

2.2.3 Warning Translation Software

The program will take in the warning singles transferred by the transceiver of the secondary controller and translate these warning signals into actions of the feedback system, including flashing LED, running vibration motors, and playing sounds through buzzers. Then the system will send them to the primary controller using a 2.4GHz Bluetooth datalink.

2.2.4 Secondary Controller

The controller will be powered through the batteries on the robot.

The controller will receive the control commands from the PC via a 2.4GHz Bluetooth data link and communicate with the robot's own controller through a wire to let it execute the commands.

The controller will also be used to read the warnings from the robot's own controller using wired connections, then send out these warnings to the PC via a 2.4GHz Bluetooth datalink.

Appendix A contains the instruction set for the robot that we use.

2.2.5 PCB Design

Below is the front and back view of the first version PCB. Although most parts of the board perform well in testing, some external modifications are done to the board (e.g. capacitors across the button, floating power converter, etc.) to make it function correctly overall. It serves as a fully functional development board during the early stages of the development, but it is soon replaced with a fully functional second version PCB.

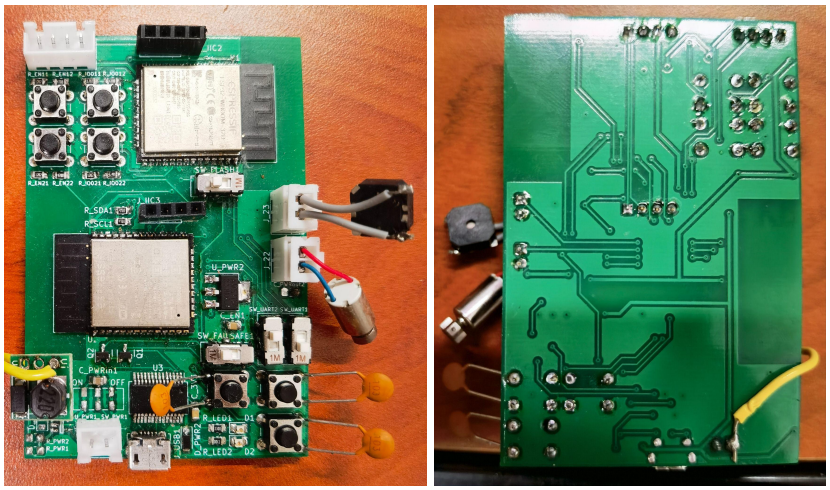

Fig 2.10 First Version PCB Overview

There are several reasons for redesigning the PCB.

- Power system failure. The power system cannot provide any power to the system, and instead, it smokes and burns. This indicates that the system needs a completely new design, probably with another chip that has large commercial use, as there will be more schematics and designs for us to refer to.
- Bad mounting method for buzzer and vibration motor. Now the two devices are not surface mounted. They are connected through JST-XH connectors. This makes them hang in the air and thus be easily broken during the operating process.

- Switches that control the flow of data. I designed the switches to be surface-mount types, but it turns out the footprint I downloaded is different from the actual footprint, the soldering pads are closer together than expected. Also, the switches are designed too close to each other, making surface mount almost impossible.
- Instructions on the silkscreen are not sufficient. Now there is only instruction on the power system that let the user know which side of the switch indicates power on and which side indicates power off. The UART and auto-program switches do not have an indication of the functionality on the silkscreen.
- The backside of PCB is not utilized. Some of the ICs can potentially be transferred to the backside of the PCB to make the design more compact.

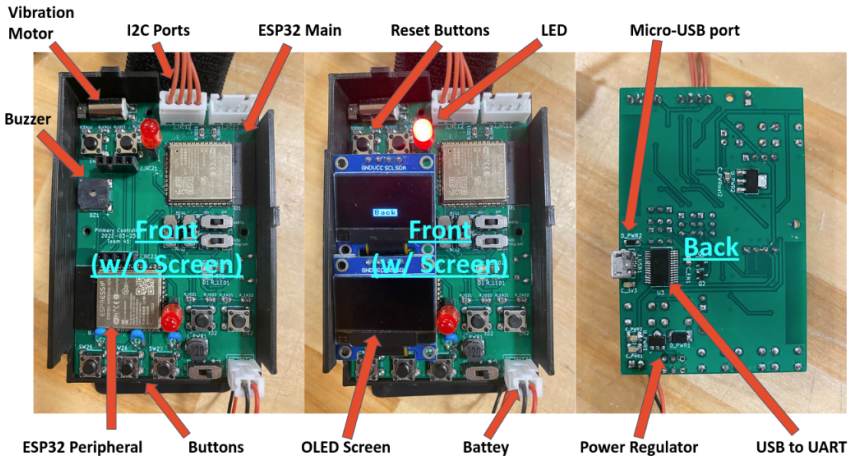

Fig 2.11 Second Version PCB Overview

2.2.6 PCB Case

In order to protect users from any possible electrical damage and make our project more engaging, a case is needed to contain our PCB board.

The requirements for the case are listed below:

1. Screens, LEDs, I2C ports, buttons, switch, battery charge port, and Micro-USB port should be exposed to the user, but other electronic components should not.
2. Should include a small area to contain the battery inside.
3. Should have two mounting points, so the device can be mounted on arms.

Besides, 6 tiny cases are also printed to contain the IMUs.

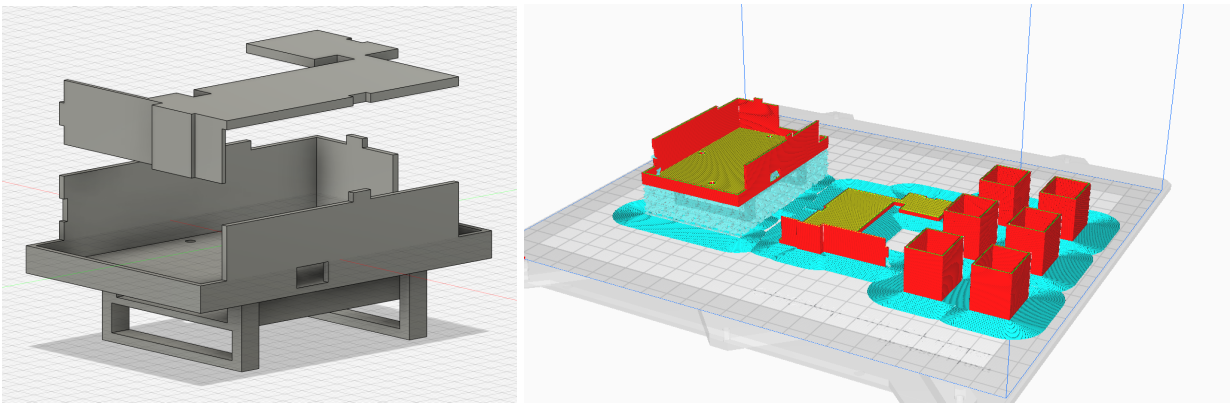

Fig 12. Cases for PCB Board and IMUs in 3D View

Fig 13. Final Physical Appearance for the Project

2.3 Requirements & Verification Tables

2.3.1 Primary Controller

Requirements	Verifications
1) After using 4.7kΩ resistors to pull up the IIC communication line, the voltage at the power side of the pull-up resistor should be at least 3.7V to make the IIC correctly functioning. 2) After a prolonged running of the microcontroller, the microcontroller, IMUs, and the feedback systems should not exceed the temperature of 40 degrees Celsius to prevent injuries to the human body. 3) After sensor fusion, the output angle of the user should be a valid configuration, and the deviation of the measured angle from the true angle should not exceed 5 degrees.	1) After powering up the microcontroller, use a multimeter to probe the voltage between the power side of the resistor and the ground. Make sure that the voltage is at least 3.7V. 2) Run the whole system normally without actually wearing the equipment, let the device run for about an hour, then measure the temperature of the whole system through an infrared thermometer, and make sure the temperature is below 40 degrees Celsius. 3) Take an individual IMU, place it on surfaces of the known slope, and read the values output from the program for testing pitch and roll. For yaw testing, place a compass along with it to check if the readings deviate at most 5 degrees.

2.3.2 Gesture Recognition Software

Requirements	Verifications
1) The half-duplex communication between the embedded processor on the user and PC is fluent and without any error present.	1) Use a wired connection along with a wireless connection, run the whole system and place the IMUs in arbitrary positions. Make sure readings from the processor and readings from the PC are the same. Then send feedback system specified commands, observing if the commands sent out are expected.

2) The software is able to recognize some gestures that are essential for the whole system to run.	2) Wear the system and calibrate it first by pressing the calibration button on board while making a fist. Make sure the software is displaying that the user is making a fist. Then extend the fingers and make sure the software recognizes that behavior.
3) The half-duplex communication between the PC and robot is fluent and without any error present.	3) Using the same method as 1), attaching wire connection between these devices, make sure the data transmitted is the same.

2.3.3 Secondary Controller

Requirements	Verifications
1) The system can successfully communicate with the robot's own controller, including sending commands to the controller and receiving warnings from the controller.	1) When a Bluetooth connection is established between PC and secondary controller, test via Bluetooth, using a wired connection otherwise. It should be able to send commands to the secondary controller to command the robot to move, turn, and rotate its gimbal, making sure the robot works as expected. Try to bump into walls or make the robot overload, see if any warning message is recognized by the secondary controller.

2.4 Tolerance Analysis

2.4.1 Accelerator Range

According to K. M. DeGoede et al.[1], the fasted motion that can be produced by human hands and arms are about to move $33 \pm 3\ cm$ in 226ms on average when intercepting incoming objects. When an object starts at rest, the distance traveled by any object under constant acceleration is

$$d = v_{init}t + \frac{1}{2}at^2 = \frac{1}{2}at^2$$

To simplify, if the process of accelerating and decelerating are considered to have the same acceleration, then the maximum speed happens when the total distance is half-traveled, and when reaching that point, the time used is exactly half of the total time. Then

$$d = \frac{1}{2}at^2 = \frac{1}{2}a(\frac{t_{total}}{2})^2 = \frac{1}{8}at_{total}^{\ 2}$$

$$\Rightarrow a = \frac{8d}{t_{total^2}} = \frac{8\times(33+3)\times10^{-2}m}{(226\times10^{-3}s)^2} = 56.39ms^{-1}$$

According to the datasheet of the WT901[2], its accelerometer can measure up to ±16g maximum, about $156.8m/s$, which is more than two times larger than the maximum velocity that a human can reach. So its accelerometer can read all user inputs correctly.

2.4.2 IIC Communication Bandwidth

According to NXP Semiconductors[3], "when using serial, 8-bit oriented, bidirectional data transfers can make at up to 100kbit/s in the standard mode." Since the standard model is the slowest mode, the bandwidth of IIC communication is at least 100kbit/s. According to the datasheet of WT901[2], one query to one of the IMUs will need the controller to send out 3 bytes (24bits) first to select the IMU using the IIC address, then the IMU will respond with a data packet of 8 bytes (64bits) for each set of parameters queried. So one query will have

$$24\,bits + 64bits = 88\,bits$$

The readings that the controller needs to figure out the position and orientation of the IMUs are accelerometer readings, gyroscope readings, and magnetometer readings. Combined together, for each query, one IMU needs

$$88bits \times 3 = 264bits$$

Then the minimum querying frequency IIC protocol will allow is

$$floor(\frac{100kbit/s}{264bits}) = floor(378.78Hz) = 378Hz$$

Considering that the system has at most 8 IMUs to query, in the worst case, each one of them will have a minimum querying frequency of 47Hz. According to the WT901 datasheet, the sampling frequency can be anywhere between 0.5Hz to 200Hz for each sensor, and 47Hz falls in the range.

2.4.3 IMU Sampling Frequency

For the 47Hz worse-case querying frequency, the maximum angular velocity that the sensor can detect defined by Nyquist sampling frequency is

$$\frac{47Hz}{2} \times 2\pi = 147.65rad/s = 23.5\,revolution/s$$

This would be more than sufficient for detecting the motion of the human body because humans cannot exceed this range under normal conditions.

Addon to that, the gyroscope of the WT901[2] can accept readings up to $\pm$ 2000 deg/s. So the motions can be captured by the IMU successfully even in situations where the IMUs are not mounted on the user.

2.4.4 Bluetooth Communication Bandwidth

According to the data provided by Espressif Systems [4], the ESP32 module has a transmission rate of up to 90KB/s.

For the angle data that is sent by the primary controller, in our design, up to eight angles are sent from the primary controller to the PC. Each of the angles can be encoded as a 2-byte integer by using the formula below

$$angle_{encoded} = int(angle_{raw,radian}/2\pi \times 32768)$$

Using this formula will cause the precision of the data to drop to $1.92 \times 10^{-4} rad = 0.011deg$, which is negligible. Then eight angles are encoded in 16 bytes, which means the HC-05 can send the data up to

$$\frac{90KB/s}{16\,byte \times 8\,bit/byte} = 720Hz$$

The frequency is more than enough compared to the potential query frequency of 47Hz.

For control data sent to the robot, this theoretical speed is also high enough that the data transmission would also use less than 1% of the total capacity.

3. Cost

3.1 Components

Vender	Item	Count	Unit Price	Total Price	Details
Digikey	330Ω Resistor	4	0.1	0.4	LED
	1kΩ Resistor	10	0.1	1	UART Bus
	2kΩ Resistor	5	0.1	0.5	UART Bus
	4.7kΩ Resistor	20	0.27	5.4	2-16 for IIC Bus
	330kΩ Resistor	5	0.1	0.5	DC-DC Step-up circuit
	1.05MΩ Resistor	5	0.1	0.5	DC-DC Step-up circuit
	1μF Capacitor	8	0.3	2.4	4 for Voltage Regulator; 1 for Controller
	4.7μF Capacitor	6	0.45	2.7	DC-DC Step-up circuit
	4.7μH Inductor	5	0.1	0.5	DC-DC Step-up circuit
	MBR0520 Schottky	4	0.43	1.72	1 for DC-DC Step-up circuit; 1 for Controller
	28DIP Socket	2	1.44	2.88	Controller
	16MHz Oscallitor	4	0.4	1.6	Controller
	DPDT Switch	10	0.99	9.9	UART
	LED	4	0.57	2.28	LED
	Push Button	10	0.168	1.68	Controller
	USB Socket	2	0.96	1.92	USB
	12kΩ Resistor	4	0.1	0.4	KEY
	470Ω Resistor	4	0.1	0.4	KEY
	ESP32-WROOM -32D	4	4.2	16.8	Controller
	NFET	4	0.22	0.88	Controller

Mouser	MCP1663 Regulator	3	1.38	4.14	DC-DC Step-up circuit
Amazon	USB-UART/FT232RL	3	5.99	17.97	2 for flash; 1 for FT232rl chip
	WT901	5	31.9	159.5	IMU
	1000mAh Li-ion Battery	2	12.99	25.98	Battery
Total				261.95	

Table2 : Component Costs

3.2 Labor

Name	Hourly Rate ($)	HoursWorked	Total ($)	Total * 2.5 ($)
Haoduo Yan	40	120	4800	12000
Eric Zhou	40	120	4800	12000
Guang Yin	40	120	4800	12000
Total				36000

Table3 : Labor Cost

4. Conclusion and Further Resources

4.1 Accomplishments

Overall, the product meets our expectations in the project proposal and design document. In our final demo, the robot could move as instructed, and the gimbal could aim smoothly. We tried several different gestures, and the algorithm identified them correspondingly with no error. Moreover, it fits most people's hands. We have not had short circuit or overheating issues in the hardware part.

4.2 Challenges

between the glove and the robot was not stable, so we examined the Bluetooth module first. We found it hard to connect because the robot was not designed for Bluetooth connection. It initially used a 2.4GHz remote controller, so the Bluetooth module did not work perfectly. Also, it could be affected by other Bluetooth devices.

The readings of IMUS are influenced by the surrounding environment's magnetic field. In our tests, it caused the robot to move imprecisely. There was also a bug in our code related to the orientation. The degree went from 359 ° to 0° suddenly as we faced north, which caused the robot to be out of order.

Another change we made was to add another ESP32 chip to the PCB board because we found that one was not enough to support the collection of data and the operation of the feedback system. In our final design, one processor worked to process the data from IMUs, and the other controlled the feedback system.

4.3 Ethical & Safety

Our product utilizes Bluetooth protocol for wireless communication between the controller and the robot, and user body data is collected to analyze and recognize user gestures and body movements. We commit to protecting and respecting user privacy, complying with section 7.8.I-1 of IEEE's guidelines [6].

For specific tasks, the project requires the use of personal protective equipment. We guarantee to take our professional responsibilities to pursue high standards on our project. Specifically, we refer to the Division of Research Safety at the University of Illinois [7].

Since our product directly contacts human skin, we make sure the wearable device we designed does not pose a threat to the user. There are no exposed electrical components, and we will keep our circuit operating under a low voltage.

4.4 Future Work

We plan to improve our project in the following ways:

- We will try a more powerful processor (possibly STM32) and remove the intermediate PC. The STM32 family has more libraries that we may make use of.
- Our software currently runs on the PC's terminal. Since we plan to remove it, we will need to develop another UI on the control board. This allows users to record more gestures, and more robots will be compatible with our glove.
- We are using 6 IMUs to do gesture recognition. We would like to add more of them to expand the range of applications.
- In our further study, we found that a method based on convolutional neural networks (CNN) can improve the accuracy and reliability of hand gesture recognition[8]. Therefore, our objective is to explore the neural network-based approach and revise our algorithm.

References

[1] K. M. DeGoede, J. A. Ashton-Miller, J. M. Liao, N. B. Alexander, “How quickly can healthy adults move their hands to intercept an approaching object? Age and gender effects, ” The Journals of Gerontology: Series A, vol. 56, no. 9, 2001.

[2] Wit Motion, “WT901 datasheet”. Available: https://github.com/WITMOTION/WT901/blob/26527441cfbc71e1ffd86c9e0fe0ef500a31a767/WT901%20Datasheet.pdf [Accessed: 23-Feb-2022]

[3] NXP Semiconductors, “IIC-bus specification and user manual.” Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf [Accessed:23-Feb-2022]

[4] Espressif Systems, “Bluetooth LE & Bluetooth”. Available: https://espressif-docs.readthedocs-hosted.com/projects/espressif-esp-faq/en/latest/software-framework/ble-bt.html#:~:text=The%20maximum%20throughput%20of%20Bluetooth,is%20about%2090%20KB%2Fs. [Accessed:23-Feb-2022]

[5] M. Zhu, Z. Sun, Z. Zhang, Q. Shi, T. He, H. Liu, T. Chen, and C. Lee, “Haptic-feedback smart glove as a creative human-machine interface (HMI) for virtual/augmented reality applications,” Science Advances, vol. 6, no. 19, 2020.

[6] “IEEE code of Ethics,” IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 21-Feb-2022].

[7] Division of Research Safety, “Electrical Safety in the Research Laboratory”. Available: https://www.drs.illinois.edu/Page/SafetyLibrary/ElectricalSafetyInTheResearchLaboratory [Accessed:23-Feb-2022]

[8] Q. Gao, J. Liu, and Z. Ju, “Hand gesture recognition using multimodal data fusion and multiscale parallel convolutional neural network for human–robot interaction,” Expert Systems, vol. 38, no. 5, Jan. 2020, doi: 10.1111/exsy.12490.

Appendix A

Example Command Set Used for Demo

Hold	Stop all action
Chassis	Move forward or back
	Turn left or right
	Rotate clockwise or counter clockwise
Gimbal	Move up or down
	Move left or right
Shooter	Shoot

Appendix B

Python Code to Convert Quaternions to Euler Angles

```python
def Q2Euler(Q: Quaternion):
    qw, qx, qy, qz = Q.elements
    sinr_cosp = 2 * (qw * qx + qy * qz)
    cosr_cosp = 1 - 2 * (qx * qx + qy * qy)
    roll = np.arctan2(sinr_cosp, cosr_cosp)

    sinp = 2 * (qw * qy - qz * qx)
    if np.abs(sinp) >= 1:
        pitch = np.copysign(np.pi / 2, sinp)
    else:
        pitch = np.arcsin(sinp)

    siny_cosp = 2 * (qw * qz + qx * qy)
    cosy_cosp = 1 - 2 * (qy * qy + qz * qz)
    yaw = np.arctan2(siny_cosp, cosy_cosp)

    return np.array([roll, pitch, yaw])
```