ECE 445

Senior Design Laboratory

Final Report

ECE445 Final Report: Voice-Controlled Wearable Drone Wristband System

Team #445

Zhu Zhengyu

(zzhu58@illinois.edu)

CHEN ZHENBO

(zhenboc2@illinois.edu)

CHEN RENANG

(renangc2@illinois.edu)

GUO JINTU

(jintug2@illinois.edu)

TA: Chen Weiye

May 17, 2026

1 Introduction

Drones are widely used in entertainment, photography, and intelligent robotic applica-tions. However, traditional drone controllers usually require complicated joystick opera-tions and significant training, which creates a high learning curve for beginners.

To improve the convenience and intuitiveness of drone control, this project proposes a voice-controlled wearable drone system. Users can control the drone through simple voice commands such as "take off," "land," and "forward," providing a more natural human–machine interaction experience.

The system consists of a VC-02 offline voice recognition module, an ESP32 microcon-troller, an ELRS wireless communication system, and a drone flight controller. Voice commands are recognized by the VC-02 module and transmitted to the ESP32 through UART communication. The ESP32 then converts the commands into CRSF remote-control frames and sends them to the ELRS transmitter for wireless drone control.

In addition, a wearable 3D-printed enclosure was designed to integrate the microphone, display, and control hardware into a compact wristband device, demonstrating the feasi-bility of lightweight and portable voice-controlled drone systems.

2 wristband system design

This project presents a voice-controlled wearable drone wristband system. Based on the conventional wireless remote control functionality of drones, the system introduces voice recognition as an additional human-machine interaction method. It allows users to per-form basic flight control operations through voice commands. This approach reduces the dependence on complex joystick manipulation in traditional remote controllers while improving the convenience and intuitiveness of system interaction.

2.1 System Overview

The wristband system mainly consists of an ESP32 main controller, a VC-02 offline voice recognition module, an OLED display module, and an ELRS wireless transmission mod-ule. The overall system workflow and the functions of each module are summarized as follows:

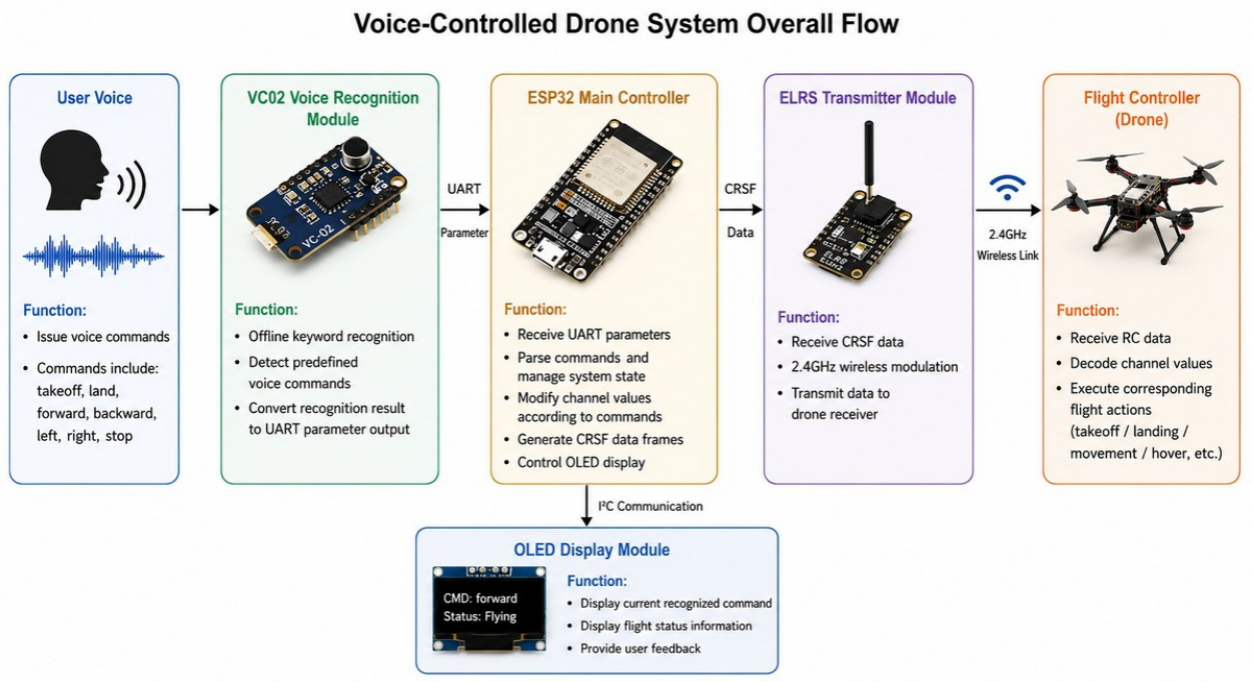

Figure 1: The overall workflow of the wristband system

After the user speaks a voice command, the VC-02 module first performs offline keyword recognition and transmits the corresponding parameter to the ESP32 through UART com-munication. Based on different commands, the ESP32 dynamically modifies the CRSF re-mote control channel values of the drone, and then sends the generated control data wire-lessly to the drone through the ELRS transmission module. In this way, functions such as takeoff, landing, directional control, and hover stabilization can be achieved.

Compared with the traditional remote control solution, this system has the following features: Supports offline voice recognition, no need for network connection; Utilizing the ELRS wireless link results in a lower communication delay; The wearable structure enhances portability;Provide a solid development foundation for subsequent functional expansions.

2.2 ESP32 Main Controller Design

As the core controller of the entire system, the ESP32 is responsible for voice command parsing, flight state switching, CRSF remote control data generation,

and OLED status display.

In the system, the ESP32 mainly utilizes two hardware UART interfaces. GPIO16 and GPIO17 are configured as the serial communication pins for the VC02 voice recognition module through UART1, while GPIO25 and GPIO26 are configured as the serial commu-nication pins for the ELRS transmitter module through UART2.

After receiving UART parameters transmitted by the VC-02 module, the ESP32 calls the

corresponding control functions and dynamically modifies CRSF remote control channel values including Roll, Pitch, Yaw, and Throttle. The ESP32 then continuously generates CRSF data frames and transmits remote control data to the ELRS transmitter module with an update period of approximately 20 ms, ensuring that the flight controller can stably receive control commands.

2.3 VC-02 Voice Recognition Module Design

To implement offline voice control functionality, the system adopts the VC-02 offline voice recognition module developed by AI Thinker.

The VC-02 performs voice detection using an offline Keyword Spotting approach. Unlike cloud-based continuous speech recognition systems, the VC-02 does not fully interpret natural language sentences. Instead, it uses pretrained keyword models to match and classify predefined command words.

After the user speaks a voice command, the onboard microphone first captures the audio signal. The analog voice signal is then converted into digital data through an ADC. After digitization, the voice data undergoes several preprocessing procedures, including envi-ronmental noise suppression, automatic gain control, and endpoint detection, in order to improve recognition stability and accuracy.

After preprocessing, the VC-02 performs feature extraction on the voice signal. The sys-tem mainly utilizes Mel Frequency Cepstral Coefficients to represent the speech spec-trum while preserving the major frequency characteristics of human voice signals. The extracted voice features are then input into the offline keyword recognition model and matched with predefined keyword templates. When the matching confidence exceeds a predefined threshold, the command is considered successfully recognized.

In the final implementation, seven English voice commands were defined:

takeoff: the drone takes off and stabilizes in the air

land: the drone slowly descends and eventually stops the motors

forward: the drone continuously performs slight forward movement

backward: the drone continuously performs slight backward

movement **left**: the drone rotates left in place

right: the drone rotates right in place

stop: the drone stops the current action, returns to a level attitude, and hovers

After a command is successfully recognized, the VC-02 transmits the corresponding pa-rameter to the ESP32 through UART1 TX. The mapping relationship between voice com-mands and UART parameters is summarized in Table 1.

Table 1: Voice Command and UART Parameter Mapping

Voice Command	UART Parameter Mapping
takeoff	1
land	2
forward	3
backward	4
left	5
right	6
stop	7

UART communication is used between the VC-02 module and the ESP32, with the com-munication baud rate configured at 115200 baud. The hardware connection relationship of the system is shown below:

Table 2: Connection Between VC-02 and

ESP32	VC-02 Pin	ESP32 Pin
UART1 TX		
GPIO16	UART1 RX	
GPIO17		
GND	GND	
VCC	5 V	

2.4 Voice Command and Drone Control Logic

After receiving the corresponding UART parameter, the ESP32 dynamically modifies CRSF remote control channel values according to different voice commands, allowing the drone to perform different flight actions.

Among these commands, “takeoff” mainly increases the Throttle channel (CH3) value so that the drone enters the takeoff state. After takeoff, the system automatically adjusts the throttle back to a preset hover value to maintain stable flight. The “land” command gradually decreases the Throttle channel (CH3) value, allowing the drone to descend and eventually stop the motors.

The “forward” and “backward” commands are mainly used to modify the Pitch channel (CH2). Specifically, “forward” increases the Pitch value, while “backward” decreases the Pitch value. The “left” and “right” commands are used to modify the Yaw channel (CH1). In this case, “left” decreases the Yaw value, while “right” increases the Yaw value.

The “stop” command is used to restore the drone to a stable hover state. Once this com-mand is received, the system immediately resets the Roll, Pitch, and Yaw channels to their preset hover baseline values while maintaining a stable hover throttle value.

Table 3 shows the baseline channel values used during hovering.

Table 3: Baseline Hover Channel Values

Channel	Value
Roll	1535
Pitch	1460
Throttle	1496
Yaw	1500

Since the system controls movement by continuously applying channel offsets, the drone will keep performing the corresponding action after receiving commands such as “for-ward”, “backward”, “left”, or “right” until the “stop” command is received. This ap-proach reduces the need for repeatedly triggering voice commands and improves the stability of continuous flight control.

2.5 OLED Display Module Design

To make system debugging and status observation more convenient, an OLED display module was added to the system for real-time display of recognized voice commands and current drone status information.

The system uses an SSD1306-based OLED display with a resolution of 128×64. Compared with traditional LCD displays, OLED displays have advantages such as lower power consumption, smaller size, and clearer display performance, making them suitable for wearable applications.

The OLED communicates with the ESP32 through the I^2C communication protocol. In the system, GPIO21 is used as the SDA data line, while GPIO22 is

used as the SCL clock line.

The hardware connection relationship is shown in Table 4.

Table 4: Connection Between OLED and

ESP32 OLED Pin	ESP32 Pin
VCC	3.3 V
GND	GND
SDA	GPIO21
SCL	GPIO22

During operation, the ESP32 dynamically updates the OLED display according to the current recognition result. For example, when the system recognizes the "forward" com-mand, the OLED displays the corresponding flight status. When the system is idle, mes-sages such as "Ready" and "safe" are displayed. Experimental results show that the OLED module can provide near real-time feedback of the current system status, which improves both debugging efficiency and user interaction experience.

2.6 ELRS Transmitter Module Design

To achieve wireless remote communication between the wristband system and the drone, the project uses an ELRS (ExpressLRS) wireless transmitter module for data transmis-sion.

ELRS is a low-latency wireless remote control protocol based on LoRa modulation tech-nology. It is widely used in drone systems because of its high refresh rate, long commu-nication range, and strong anti-interference capability.

In the system, the ESP32 dynamically generates CRSF remote control data frames accord-ing to the current flight state and transmits them to the ELRS transmitter module through UART communication. The ELRS module then wirelessly sends the control data to the drone receiver, where the flight controller parses the received commands and performs the corresponding actions.

UART communication is used between the ESP32 and the ELRS transmitter module,

with the baud rate configured at 420000 baud. The hardware connection relationship is shown below.

Table 5: Connection Between ELRS Module and

ESP32 ELRS Pin	ESP32 Pin
TX	GPIO25
RX	GPIO26
GND	GND
VCC	5 V

During operation, the ESP32 continuously transmits CRSF remote control channel data to the ELRS module, allowing real-time drone control. Experimental testing showed that the wireless communication link could stably transmit voice control data with relatively low overall control latency, which meets the requirements of real-time drone flight con-trol.

3 Flight Controller System Design and Implementation

This section describes the design and debugging process of the flight controller system, which is responsible for the drone' s stability and hover performance. The work mainly in-volved telemetry testing under Betaflight, migration to ArduPilot/ArduCopter firmware, and the final implementation of the onboard altitude hold control scheme.

3.1 Initial Betaflight Telemetry Scheme and Testing

At the beginning of the project, the system used a SpeedyBee F405 V5 flight controller running Betaflight firmware. The original plan was to use telemetry feedback through the ExpressLRS (ELRS) link so that altitude information from the drone could be transmitted back to the wearable controller. The wearable system would then use the received altitude data to calculate throttle corrections according to different voice commands.

To test this idea, several CRSF telemetry frames were examined, including GPS alti-tude data, barometer altitude data, attitude data, and battery sensor data. The teleme-try scheduling logic inside Betaflight was modified so that altitude-related information could be transmitted with higher priority. After each modification, the firmware was recompiled and flashed to the flight controller for testing.

During testing, several problems appeared. Some telemetry frames could be generated correctly on the flight controller side but were not forwarded reliably through the ELRS wireless link. GPS altitude data could be received successfully, but the update rate was relatively slow and the altitude value only had integer-meter resolution, which was not sufficient for responsive altitude control.

Barometer altitude data was expected to per-form better because of its higher resolution, but under the tested ELRS setup it could not be transmitted consistently.

These issues showed that using telemetry feedback as a real-time control signal was not reliable enough for stable altitude holding. Because of this, the original external closed-loop control approach based on telemetry feedback was eventually abandoned.

3.2 ArduPilot/ArduCopter Firmware Migration and AltHold Control

To improve system stability, the project later switched from Betaflight to ArduPilot/ArduCopter firmware. Compared with the previous approach, ArduPilot already includes a built-in AltHold mode that uses onboard barometer and IMU sensor data for altitude stabiliza-
tion. This allowed the altitude control loop to run directly inside the flight controller

rather than on the external wearable device.

The migration process required finding the correct ArduCopter firmware target for the SpeedyBee F405 V5 flight controller, downloading the firmware, and flashing it onto the board. After flashing, the flight controller was configured again in Mission Planner, in-cluding frame type selection, ESC protocol configuration, CRSF receiver input mapping, flight mode setup, and pre-arm safety checks.

In the final design, the wearable voice-control module no longer sends altitude values or text-based control instructions to the flight controller. Instead, it behaves similarly to a standard RC transmitter by sending CRSF remote-control channel values through the ELRS link. ArduPilot interprets these values as normal RC inputs.

For example, when the user says the "hover" command, the system switches the flight mode channel to AltHold mode while setting the throttle channel near the middle posi-tion. The actual altitude stabilization is then handled internally by the ArduPilot control algorithms running on the flight controller.

Compared with the previous telemetry-based approach, this method is more practical and stable because the hover function no longer depends on delayed telemetry feed-back. The flight controller can directly use onboard sensor data for real-time stabilization, which improves both flight safety and hover performance.

4 Mechanical Design

4.1 Drone Mechanical Design

During the development and assembly process of the drone platform, two different ground-station software platforms were used for flight-controller configuration and hardware verification. These tools were mainly used to configure motor rotation directions, cali-brate ESC outputs, verify receiver input signals, and test overall flight-controller func-tionality before actual flight testing.

Particular attention was given to the motor direction setup and propeller installation process. Since quadrotor stability strongly depends on the correct

combination of mo-tor rotation direction and propeller orientation, each motor was individually tested after firmware configuration. Incorrect motor direction or propeller placement could easily result in immediate instability or loss of control during takeoff. Therefore, the motor outputs were repeatedly checked through the ground-station software before flight test-ing.

After verifying motor rotation directions, corresponding clockwise and counterclockwise propellers were installed according to the frame layout configuration. Additional testing was then performed to ensure that all four motors generated thrust in the correct direction and responded properly to transmitter inputs.

The drone structure uses a Mark3 carbon fiber quadcopter frame to provide lightweight construction and sufficient structural strength during flight. High-strength 5-inch plastic propellers were selected to improve durability and reduce the risk of damage during repeated testing. In addition, four propeller protection guards were installed around the frame to reduce collision damage and improve operational safety during indoor testing and debugging.

The ESC and flight controller system were manufactured by SpeedyBee and connected through a 10-pin cable interface, simplifying the wiring structure and improving assembly reliability. The power connector and motor wires were all permanently fixed using solder joints to ensure stable electrical connections during high-current operation and flight vibration.

The receiver connections, including GND, VCC, TX, and RX, were soldered directly to the UART6 interface of the flight controller for reliable CRSF communication. A 1000 μF electrolytic capacitor was additionally installed at the power input to suppress voltage fluctuations and reduce electrical noise generated by the motors and ESCs.

To improve cable management and reduce interference, the motor wires were intentionally shortened and adjusted to appropriate lengths, then bundled and secured using insulating tape. This reduced unnecessary wire movement and improved the overall cleanliness of the internal structure.

The battery was mounted above the center of the frame using a battery strap and firmly secured in place. Positioning the battery near the center of gravity helped improve flight balance and overall stability during hovering and directional movement.

4.2 Wristband Enclosure Design

4.2.1 Enclosure Requirements and CAD Design

To make the wearable voice-control system more suitable for testing and demonstration, a dedicated enclosure was designed and manufactured using 3D printing. The enclosure was designed to hold the main electronic components of the wristband system. Openings were reserved for the OLED display and microphone input so that the display remained visible and the voice module could still receive commands clearly.

Since the system is intended to be worn on the wrist, the enclosure was designed with a compact form factor and lightweight structure. Internal mounting positions were also added in the CAD model to help secure the circuit boards and reduce movement of the components during operation.

4.2.2 3D Printing and Assembly

The enclosure was fabricated using FDM 3D printing. Several print iterations were tested in order to check component fit, structural strength, and assembly tolerance. After the final assembly, the wearable module became significantly more compact and organized compared with the earlier breadboard-style prototype.

The completed enclosure also improved the practicality of the system during demonstra-tions and flight testing. By integrating the electronics into a single wearable structure, the overall system became easier to carry, operate, and test in real flight conditions.

5 Quantitative Results

Table 6: Voice Command Mapping to Drone Control Channels

Voice Command	Parameter	CH1 Roll	CH2 Pitch	CH3 Throttle	CH4 Yaw	CH5 Mode	Duration / Result
takeoff	1	1499	1499	1663	1500	1500	Hover after 1.5 s
land	2	1499	1499	1350	1500	1500	Safe after 1.5 s
forward	3	1499	1619	1540	1500	1700	Hover after 1.5 s
backward	4	1499	1379	1540	1500	1700	Hover after 1.5 s
left	5	1379	1499	1540	1500	1700	Hover after 1.5 s
right	6	1619	1499	1540	1500	1700	Hover after 1.5 s
lock	8	1500	1500	988	950	1500	Maintain for 6 s
unlock	9	1500	1500	988	2050	1500	Maintain for 8 s
up	10	1499	1499	1663	1500	1700	Hover after 0.5 s
down	11	1499	1499	1350	1500	1700	Hover after 0.5 s

Table 7: Hover State Channel Values

CH1 Roll	CH2 Pitch	CH3 Throttle	CH4 Yaw	CH5 Mode
1499	1499	1496	1500	1700

Table 8: Safe State Channel Values

CH1 Roll	CH2 Pitch	CH3 Throttle	CH4 Yaw	CH5 Mode
1500	1500	988	1500	1500

Table 9: Voice Recognition Accuracy

Command	Trials	Successful Recognition	Accuracy
Take off	10	8	80%
Land	10	9	90%
Forward	10	7	70%
Backward	10	7	70%
Left	10	8	80%
Right	10	8	80%
Up	10	9	90%
Down	10	9	90%

Table 10: Voice Module to Drone Communication Success Rate

Command	Trials	Successful Transmission	Success Rate
Take off	10	10	100%
Land	10	10	100%
Forward	10	10	100%
Backward	10	10	100%
Left	10	10	100%
Right	10	10	100%

6 Accomplishments

1. Built and configured the basic drone hardware system, including the frame, motors, ESCs, and flight controller.

2. Established the ELRS wireless communication link and successfully transmitted CRSF control data from ESP32 to the drone.

3. Implemented the VC-02 offline voice recognition system with multiple custom voice commands.

4. Completed the full voice-control process from voice input to wireless drone con-trol.

5. Tested different flight-control solutions and finally used ArduPilot AltHold mode to improve flight stability.

6. Designed and manufactured a wearable 3D-printed wristband case for the control system.

7. Integrated all hardware and software modules and successfully demonstrated basic voice-controlled drone flight functions.

7 Uncertainties and Limitations

1. The wearable mechanical structure is still relatively large and can be redesigned into a more compact and lightweight form in the future.

2. The drone control response is not sensitive enough in some situations, especially dur-ing fast movement and direction changes.

3. The voice recognition system may occasionally recognize the wrong command.

4. In some environments, the VC-02 module may fail to detect voice commands cor-rectly.

5. The drone flight stability still needs improvement, especially during hovering and sudden movements.

6. Communication delay and control instability may sometimes cause unexpected drone behavior or possible loss of control.

8 Future Work and Improvements

1. The drone control response is sometimes not sensitive enough because the current system only modifies fixed RC channel values. In the future, smoother control algorithms and dynamic PWM adjustment can be implemented to make the drone movement more responsive and natural.

2. The VC-02 module may occasionally recognize incorrect commands because offline keyword matching is sensitive to pronunciation and environmental noise. Future work can improve this by redesigning the command set, using shorter and more distinguish-able keywords, and adding repeated-command confirmation for critical operations such as takeoff and landing.

3. Voice detection performance decreases in noisy environments because the current mi-crophone has limited noise resistance. A higher-quality microphone module and addi-tional digital noise filtering algorithms can be added to improve recognition accuracy.

4. The drone flight is still not stable enough during hovering and movement. Future improvements can include further PID tuning in ArduPilot, better calibration of the IMU sensors, and optimization of the drone weight distribution.

5. Communication delay and occasional control instability may affect flight safety. Future versions can optimize UART transmission timing and CRSF packet handling to reduce latency and improve communication reliability.

6. The current system may still have the risk of losing control during unexpected sit-uations. Additional safety functions such as emergency motor stop, automatic landing when signal is lost, and low-battery protection can be added in future designs.

7. More intelligent functions can also be added in the future, such as gesture control using IMU sensors, autonomous waypoint navigation, and obstacle avoidance using cameras or distance sensors.