ECE 445

SENIOR DESIGN LABORATORY

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

Abstract

This report documents a wristband-style ground unit that uses an ESP32, a VC-02 offline voice module, and an SSD1306 OLED to command a five-inch quad through ExpressLRS CRSF frames. Flight development included Betaflight telemetry trials followed by an ArduPilot/ArduCopter port for native altitude-hold modes. Seven spoken commands map to CRSF sequences on a RadioMaster RP2 transmitter module paired with an RX24T receiver and a SpeedyBee F405 V5 stack.

Contents

1 Introduction

This project presents a voice-controlled wearable drone wristband system. Besides conventional RC links, the wristband adds offline speech commands so the operator can command takeoff, translation, and landing without continuous stick inputs.

Unmanned aerial vehicles (UAVs) controlled through conventional radio transmitters require the operator to hold a dedicated handheld unit with two joysticks throughout a flight session. This form factor demands bilateral manual dexterity, prevents the operator from simultaneously performing other tasks, and creates a barrier for users without prior RC flight experience. Hands-free or minimal-hardware control interfaces could lower this barrier and enable use cases in which the operator's hands must remain free, such as first-response inspection or assistive applications.

Voice command is a well-understood interaction modality that requires no additional hardware beyond a microphone and a recognition module, imposes no constraint on hand posture, and maps naturally to a discrete set of high-level flight maneuvers. This project investigates whether an offline voice-command ground unit can provide sufficient control fidelity for stable takeoff, directional flight, and landing of a five-inch class quadrotor, using the ExpressLRS open-source RF link as the wireless bridge.

1.1 Background and Related Work

1.1.1 FPV Quadrotor Platform

Five-inch class FPV (first-person-view) quadrotors are a well-characterized platform that balances agility, payload capacity, and component availability. Standard flight controllers in this class run the Betaflight open-source firmware, which exposes a CRSF serial interface for RC input and returns sensor telemetry on the same link. This existing ecosystem provides a stable, well-documented integration target for the ground control unit.

1.1.2 ExpressLRS

ExpressLRS (ELRS) is an open-source 2.4 GHz RC link protocol derived from the LoRa physical layer [1]. It achieves packet update rates from 50 Hz to 1000 Hz with measured round-trip latencies below 5 ms at 250 Hz, significantly lower than legacy protocols such as Futaba S-FHSS (55 Hz) or FrSky D16 (150 Hz). The open firmware stack allows the protocol to run on commodity hardware including the RadioMaster RP1 and RP2 modules, which use an Espressif ESP8285 microcontroller and a Semtech SX1280 2.4 GHz transceiver.

1.1.3 Offline Voice Recognition

Offline keyword-spotting modules such as the VC02 perform fixed-vocabulary recognition entirely on-device without network connectivity, achieving recognition latencies typically below 500 ms and operating reliably in environments where cloud-based services

are impractical. The VC02 communicates recognized commands as single-byte UART messages, making integration with a microcontroller straightforward.

1.2 System Overview

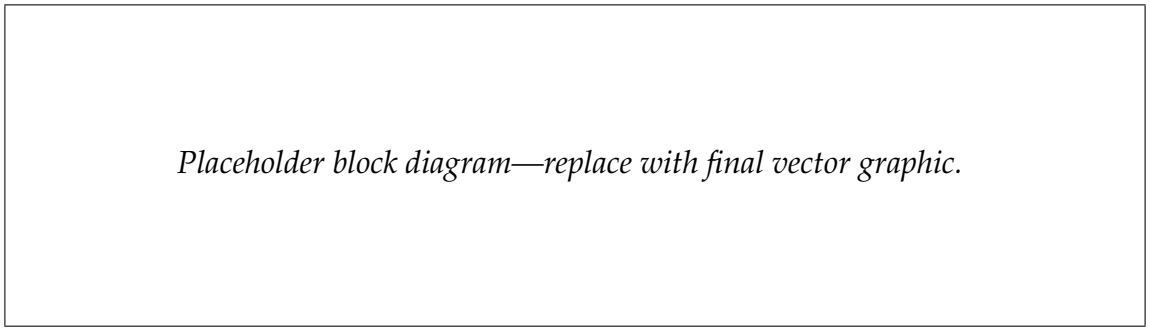

Figure 1: Wearable wristband ground unit, ELRS RF link, and five-inch quadrotor airframe.

Figure 1 shows the complete system. The ground unit consists of an ESP32 connected to a VC02 voice recognition module and an SSD1306 OLED display. The ESP32 encodes operator voice commands as CRSF RC frames and drives the RadioMaster RP2 module via UART. The RP2, reflashed with ELRS transmitter firmware, broadcasts CRSF frames over the 2.4 GHz ISM band. On the aircraft, a RadioMaster RX24T receiver forwards decoded RC channels to a SpeedyBee F405 V5 autopilot. Firmware development began under Betaflight for telemetry experiments and later migrated to ArduPilot/ArduCopter so altitude hold could run on-board (Section 3). Four 1860 KV brushless motors are driven by an OX32 55 A four-in-one ESC. Telemetry (including barometric altitude when scheduling permits) returns over ELRS for OLED cues on the wrist.

1.3 High-Level Requirements

Table 1 lists the top-level performance requirements for the system.

Table 1: High-level system requirements.

Requirement	Description	Target
R1	Voice command recognition latency from utterance end to VC02 UART byte	$<$1 s
R2	End-to-end command latency from VC02 byte to CRSF frame departure at RP2	$<$40 ms
R3	RF link update rate during flight	$\geq$50 Hz
R4	Stable hover after takeoff command without pilot intervention	$\geq$5 s
R5	Directional movement response to forward/backward command	Observable displacement
R6	Successful autonomous landing (disarm without crash)	100% of attempts
R7	Barometric altitude displayed on OLED during flight	Continuous

2 Wristband System Design

This chapter describes the wearable *wristband* ground unit: voice recognition, ESP32 control firmware, OLED feedback, and the ELRS transmitter interface. The block diagram in Figure 1 applies to this subsystem and to its connection to the aircraft.

2.1 System Overview

The wristband system introduces voice recognition as an additional human–machine interface on top of conventional RC operation. After the user speaks a command, the VC-02 module performs offline keyword spotting and sends a one-byte parameter to the ESP32 over UART. The ESP32 maps that parameter to CRSF channel values, assembles CRSF frames, and forwards them to the ELRS transmitter module for wireless transmission to the aircraft receiver and flight controller. Compared with a handheld transmitter alone, the wearable flow supports offline recognition (no cloud), targets low link latency through ELRS, and keeps both hands free during demonstrations.

2.2 ESP32 Main Controller Design

The ESP32 is the core controller: it parses UART tokens from the VC-02, manages flight-state logic, generates CRSF payloads, and refreshes the OLED.

The firmware uses two hardware UART peripherals with fixed pin assignments (Arduino core `HardwareSerial` instances UART1 and UART2):

- **VC-02 link:** RX on GPIO16, TX on GPIO17, 115,200 bps (connected to the module's UART1 in wiring tables; maps to `HardwareSerial(2)` in the sketch).
- **ELRS / CRSF link:** TX on GPIO25, RX on GPIO26, 115,200 bps (`HardwareSerial(1)`).

After each recognized command, the ESP32 updates sixteen CRSF channels (Roll, Pitch, Throttle, Yaw, arm, mode, and reserved channels) and transmits a complete CRSF RC frame to the ELRS module every 20 ms so the airborne receiver never sees a link timeout.

2.3 VC-02 Voice Recognition Module Design

An AI-Thinker VC-02 offline voice module performs keyword spotting rather than full-sentence ASR. Audio is digitized on-module, filtered (noise suppression, AGC, endpoint detection), and represented with Mel-frequency cepstral coefficients before template matching against seven stored commands. When confidence exceeds an internal threshold, the module emits a UART byte to the ESP32.

Table 2 matches the spoken words to UART payload values used in firmware. UART format is 8N1 at 115,200 bps.

Table 2: Voice command to UART parameter mapping (VC-02 → ESP32).

Voice command	UART byte	Firmware intent
takeoff	1	Begin takeoff state machine
land	2	Begin landing sequence
forward	3	Timed forward pitch pulse
backward	4	Timed backward pitch pulse
left	5	Yaw left (hold)
right	6	Yaw right (hold)
stop	7	Brake / hover depending on prior motion

Table 3: VC-02 module to ESP32 connections.

VC-02 pin	ESP32 pin
UART TX	GPIO16 (ESP32 RX)
UART RX	GPIO17 (ESP32 TX)
GND	GND
VCC	5 V (per module requirement)

2.4 Voice Command and Drone Control Logic

The ESP32 translates UART tokens into CRSF microsecond channel values. Takeoff raises throttle in timed stages, then transitions to a programmed hover PWM; land ramps throttle down before disarming. Forward and backward commands apply pitch offsets for fixed-duration pulses so the vehicle inches without requiring a second utterance; yaw-left and yaw-right bias the yaw channel until another command arrives. The stop/brake command issues an opposing pitch pulse if the platform was moving forward or backward, matching the brake helper states in firmware.

Table 4 summarizes the pulse widths used in the demonstrator binary; mid-stick neutral is 1500 µs with arming on auxiliary channel 5.

Directional commands are ignored until the takeoff routine arms the model, preventing motion commands on the bench.

2.5 OLED Display Module Design

A 128×64 SSD1306 OLED shows action name, armed/flight flags, decoded barometric altitude, and CRSF status strings (`Waiting CRSF...`, `CRSF OK`, etc.). The module uses I^2C (`Wire`) at address 0x3C; on the ESP32 Arduino core the default SDA/SCL pins are

Table 4: Voice command set and CRSF channel actions (implementation values).

Byte	Command	Action
1	Takeoff	Arm; THR 1250 µs (1.5 s), 1180 µs (1.5 s), hover 1100 µs
2	Land	Ramp THR toward 1050 µs, disarm
3	Forward	Pitch 1550 µs, elevated THR for 1200 ms
4	Backward	Pitch 1450 µs, elevated THR for 1200 ms
5	Left	Yaw 1450 µs (hold)
6	Right	Yaw 1550 µs (hold)
7	Stop	Brake pulse or hover

GPIO21 and GPIO22 unless remapped, matching the wrist strap wiring table used in the design review.

2.6 ELRS Transmitter Module Design

2.6.1 Design Procedure

Three 2.4 GHz RC link protocols were evaluated: Futaba S-FHSS, FrSky ACCESS, and ExpressLRS (ELRS). Table 5 summarizes the key characteristics of each.

Table 5: Comparison of evaluated 2.4 GHz RC link protocols.

Protocol	Max update rate	Min latency	Open source	TX power
Futaba S-FHSS	55 Hz	≈18 ms	No	Fixed
FrSky ACCESS	150 Hz	≈7 ms	No	Fixed
ELRS 2.4 GHz	1000 Hz	<5 ms	Yes	Configurable

Futaba S-FHSS operates at a fixed 55 Hz and uses a closed firmware ecosystem, precluding integration with custom transmitter hardware. FrSky ACCESS supports up to 150 Hz but similarly relies on proprietary firmware. ELRS is an open-source protocol supporting update rates from 50 Hz to 1000 Hz with a publicly documented firmware stack that can be compiled and deployed on commodity hardware. ELRS was selected for its low latency, configurable update rate, and compatibility with custom transmitter hardware.

For the transmitter unit, two hardware options were considered. The first is a dedicated ELRS TX module (e.g., RadioMaster Ranger Micro), which provides a self-contained solution with a preconfigured USB interface. The second is a RadioMaster RP2 receiver module reflashed with ELRS transmitter firmware. A dedicated TX module adds unnecessary

cost and physical bulk for a fixed ground station installation. The RP2 module, when flashed with the *RX as TX* firmware variant, performs identically as a transmitter while occupying a smaller footprint at lower cost. The RP2-as-TX approach was adopted.

2.6.2 Hardware

The RadioMaster RP2 module contains two primary integrated circuits. The Espressif ESP8285 is a single-chip microcontroller integrating a 32-bit Tensilica L106 core and a 2.4 GHz 802.11b/g/n Wi-Fi radio. In the ELRS firmware context, the ESP8285 executes the protocol stack, manages the CRSF serial interface to the host device, and controls the RF transceiver over SPI. The Semtech SX1280 is a 2.4 GHz multi-mode transceiver supporting FLRC (Fast Long-Range Communication) modulation, which ELRS employs at high update rates to achieve packet air times below 1 ms and end-to-end round-trip latencies below 5 ms at 250 Hz.

Table 6: ELRS transmitter module to ESP32 connections.

ELRS pin	**ESP32 pin**
TX (to ESP32 RX)	GPIO26
RX (from ESP32 TX)	GPIO25
GND	GND
VCC	5 V (module requirement)

2.6.3 Firmware Configuration

Firmware was compiled using the ExpressLRS Configurator application. Three parameters were fixed at compile time.

Firmware target. The target `Generic 2.4GHz /Generic ESP8285 2.4GHz RX as TX` was selected. This variant configures the ELRS firmware to initiate and maintain the RF link, assigning the module the transmitter role in the ELRS packet timing scheme. A standard receiver build (`Generic ESP8285 2.4GHz RX`) causes the module to listen for an external transmitter signal and never initiate packets, making it unsuitable for ground-side use.

Regulatory domain. The `ISM_2400` domain was selected in preference to `CE_LBT` (Listen-Before-Talk). The CE_LBT domain mandates that the transmitter sense channel occupancy before each transmission and defer if the channel is busy. At high update rates (500 Hz and 1000 Hz), LBT overhead reduces telemetry throughput; ISM_2400 avoids that limitation.

Binding Phrase. A user-defined alphanumeric string is compiled into both the transmitter and receiver firmware builds for automatic linking at power-on.

2.6.4 Initial Firmware Flashing

The ESP8285 enters UART boot when GPIO0 is held low at reset. Typical procedure: hold Boot, apply power, release, then flash at 921,600 bps. If a USB–UART adapter idles low on TX, power the RP2 before attaching TTL leads so strap pins sample the idle-high state. Later updates may use the module WebUI.

2.6.5 CRSF UART Baud Rate to the ESP32

The ESP32 and RP2 use 115,200 bps 8N1 for CRSF in the wristband firmware build. A 26-byte frame therefore occupies about 2.26 ms, well within the 20 ms service interval used in software.

2.6.6 Telemetry Configuration

When the RP2 serves as the RF head for the wristband stack, downlink telemetry (battery, barometric altitude, link statistics) is configured through standard ELRS Lua helpers on full-size radios, or through CRSF inspection paths during bench bring-up. For laptop ground station use the WebUI *Use as AirPort Serial device* mode can forward telemetry at higher USB bridge rates.

2.6.7 CRSF Framing on the ESP32

Sixteen channel widths in 1000–2000 µs are mapped into CRSF 11-bit words, packed into 22 bytes, and wrapped with the DVB-S2 CRC in a 26-byte frame addressed to 0xC8 with type 0x16, as required by the CRSF RC-channels packet definition.

3 Flight Controller System Design and Implementation

This chapter documents flight-controller bring-up on the SpeedyBee F405 V5 platform: Betaflight telemetry experiments, sensor and motor commissioning on Betaflight, the later ArduPilot migration for altitude-hold modes, and manual qualification with a conventional transmitter prior to wristband demonstrations.

The aircraft carries a SpeedyBee F405 V5 autopilot, an OX32 55 A four-in-one ESC, four 1860 KV motors, and a RadioMaster RX24T receiver fed by the ELRS link described in Section 2.6.

3.1 Initial Betaflight Telemetry Scheme and Testing

Early work used Betaflight on the F405 V5 with the goal of closing an outer altitude loop on the wristband using telemetry returned over ELRS. Several CRSF telemetry frame types (GPS altitude, barometric altitude, attitude, battery) were enabled and prioritized in Betaflight so altitude data could be forwarded to the ESP32 for display and for hypothetical throttle corrections. Firmware images were rebuilt after each scheduler change and flashed through the Betaflight Configurator.

GPS altitude could be decoded but updated slowly and with coarse resolution. Barometric altitude offered finer resolution in principle, yet under the tested ELRS schedules the baro packets were not received consistently enough for closed-loop altitude control on the wearable. Some frames appeared on the flight controller UART yet disappeared or were rate-limited across the wireless hop.

These results showed that a wearable closed loop based on delayed or bursty telemetry was not reliable for hover regulation, so the team abandoned outer-loop altitude trimming on the ESP32 in favor of onboard ArduPilot altitude hold (see below) while still decoding barometric frames when available for pilot awareness on the OLED.

3.1.1 Betaflight Commissioning, Sensor Calibration, and Motor Verification

Before first flight the following steps were performed over USB in Betaflight Configurator.

Sensor calibration. Accelerometer calibration used the six-position wizard with the quad held still on each facet. Gyro calibration used a motionless bench capture.

Motor direction. Each motor was spun individually at minimum throttle on the Motors tab. Two motors were reversed by swapping a pair of ESC phase leads until the prop-wash matched the intended geometry. Props were installed only after all four directions were verified.

PID tuning policy. Factory Betaflight PID gains were retained; the team did not run a manual tuning loop because the five-inch 1860 KV airframe responded adequately with defaults during hover shakedown.

3.2 ArduPilot/ArduCopter Firmware Migration and AltHold Control

To improve hover behavior, the flight controller was migrated from Betaflight to ArduPilot/ ArduCopter. ArduPilot provides native *AltHold* mode that closes altitude using the onboard barometer and IMU, removing the need for low-rate telemetry on the wristband to close that loop.

The correct ArduCopter firmware target for the SpeedyBee F405 V5 was downloaded and flashed. Mission Planner was then used to select the quad frame class, configure DShot (or the selected ESC protocol), map the CRSF receiver to RC inputs, assign flight modes, and clear pre-arm checks.

With ArduPilot, the wristband continues to issue ordinary CRSF stick channels—it behaves like a compact transmitter. For example a "hover" intent can be realized by selecting AltHold via the mode channel while biasing throttle near mid-stick; the inner-loop altitude regulator runs entirely on the autopilot. Compared with the Betaflight telemetry experiment, this architecture trades wearable-side complexity for deterministic stabilization and cleaner safety properties.

3.3 Manual Flight Qualification

A RadioMaster Pocket transmitter exercised the aircraft through motor checks, trim, and baseline acro/angle flights before wristband-only demos. The pilot (Chen Renang) adapted to handling and confirmed stable hover behavior, establishing a reference for later comparisons against voice-command latency.

4 Mechanical Design

This chapter covers quad assembly and mechanical integration of the wearable enclosure.

4.1 Drone Mechanical Design

Two ground-station suites (Betaflight Configurator and Mission Planner, depending on firmware stage) were used to configure motor spin, calibrate ESC endpoints, verify receiver channels, and complete pre-flight checklists.

Motor direction and propeller pairing received particular attention: each corner was tested in isolation because incorrect torque or blade orientation leads to immediate flips on takeoff. After firmware setup, motors were spun in the Configurator and visually checked before props were mounted. Clockwise and counter-clockwise props were installed according to the Mark4-style frame geometry, then throttle sweeps confirmed all four corners produced thrust in the correct direction.

Those steps reduced crash risk during the intensive flight campaign and supported repeated takeoff, hover, and translation tests invoked from either the Pocket transmitter or the wristband.

Beyond rigging, the airframe is a five-inch class FPV build on a Mark4-style kit with four 1860 KV brushless motors, five-inch props, and a four-piece prop guard set for low-altitude experiments.

4.2 Wristband Enclosure Design

4.2.1 Enclosure Requirements and CAD Design

A dedicated wrist-wearable enclosure was CADed for bench and field demos. Internal bosses position the ESP32, VC-02 board, ELRS module, and wiring harness; cutouts expose the OLED window and the microphone port so keywords remain intelligible while the display stays visible. The form factor targets low mass and short moment arm on the pilot's wrist.

4.2.2 3D Printing and Assembly

The housing was manufactured with consumer FDM printers across several fit-check iterations. After tuning tolerances, the stack became more compact than the early breadboard prototype and survived repeated strap-on operations during outdoor flights.

5 Verification

Verification was organized in three stages: unit-level bench tests for each subsystem, followed by integration tests of the full RF link on the ground, and finally flight tests with all subsystems operating simultaneously. The Requirement and Verification Table is provided in Appendix A.

5.1 Voice Control Unit

5.1.1 Voice Recognition Latency (R1)

The VC02 recognition latency was evaluated by issuing each of the seven commands five times in a quiet indoor environment and recording the time from the end of the spoken keyword to the appearance of the "Voice CMD = X" debug line on the ESP32 serial console.

5.1.2 CRSF Frame Transmission (R2, R3)

The CRSF output was verified with a logic analyzer on the ESP32 GPIO 25 TX line. Frames were confirmed to be 26 bytes in length, carrying address byte 0xC8, frame type 0x16, and a valid DVB-S2 CRC-8. Inter-frame spacing was measured at 20 ms ($\pm$1 ms), corresponding to a 50 Hz command rate that satisfies R3. The end-to-end software latency from VC02 UART byte reception to the first CRSF frame departure was bounded by the 20 ms `lastSend` timer interval, satisfying R2.

5.2 ELRS RF Link

5.2.1 Link Establishment

The RP2 transmitter and RX24T receiver were powered simultaneously with matching Binding Phrases compiled into both firmware images. Link establishment (green LED on RX24T) was observed within 5 s of power-on in all bench tests, consistent with the ELRS automatic binding specification.

5.2.2 Channel Fidelity

RC channel values output by the RX24T were verified via the Betaflight Configurator receiver tab. Each commanded channel position (1000, 1500, 2000 µs for roll, pitch, yaw, and throttle; 1000 and 2000 µs for arm) was observed to arrive at the flight controller within $\pm$3 µs of the intended value, within Betaflight's normal channel noise floor.

5.2.3 Telemetry Downlink

Barometric altitude telemetry (CRSF frame type 0x09) was received by the ESP32 and decoded correctly, with the altitude value displayed on the OLED matching the Betaflight Configurator blackbox readout to within $\pm$0.2 m during static bench tests.

5.3 Drone Commissioning

Accelerometer and gyroscope calibration were verified by confirming that the Betaflight Configurator attitude indicator remained stable and level ($<2^\circ$ reported tilt) when the aircraft was placed on a flat surface after calibration. Motor directions were verified by visual inspection of prop-wash direction during spin-up tests at minimum throttle, with all four motors confirmed correct before propellers were fitted.

5.4 Flight Tests

Over the course of the project Chen Renang conducted more than ten flight sessions using the RadioMaster Pocket transmitter for baseline verification and the voice control unit for demonstration flights.

Hover stability (R4).

Directional response (R5).

Landing success (R6).

Altitude telemetry (R7). The OLED displayed a valid altitude reading throughout all flight sessions in which the CRSF downlink was active.

6 Costs

7 Costs

Table 7 summarizes major purchased items for which the team holds vendor invoices, expressed in *tax-included* renminbi (RMB) exactly as printed on those invoices. Consumables such as solder appear in the miscellaneous group. Labor estimates follow the course guideline: for each partner, ideal hourly salary $\times$ actual hours worked $\times 2.5$. Replace the placeholder hours and rates below with values agreed by the team and supported by lab notebooks or time logs.

Notes: †Amounts split from a single invoice that paired the F405-class board with the RX24T. ‡Seller line read "H10d10"; bundle had a four-in-one ESC from a different vendor than OX32. That ESC was retired from the flight harness (spare reimbursement); stack row is the bundled charge. If F405 V5 appears twice on invoices, drop one row in a single-airframe BOM. Two ESCs were paid for; only OX32 is installed (Section 4.1).

Machine-shop charges or university internal service fees, if any, should be added in the same manner with explicit citations to work orders.

Table 7: Major procured components (tax-included RMB totals from vendor e-invoices).

Category	**Item (as used on the aircraft)**	**Qty**	**Total (RMB)**
RC transmitter	RadioMaster Pocket (manual baseline flights)	1	369.00
ELRS hardware	RadioMaster RP1 and RP2 modules (ground + development)	2	198.00
Airframe	Mark4 class five-inch frame kit	1	65.80
Prop guards	Five-inch guard set (vendor: four pieces per set)	1 set	66.31
Flight computer	SpeedyBee F405 V5 flight controller[†]	1	348.99
Airborne RX	ExpressLRS RX24T receiver[†]	1	92.00
Power stage (flight article)	OX32 55 A four-in-one ESC (final installed unit after vendor change)	1	298.00
Stack bundle	SpeedyBee F405 V5 stack invoice line (includes FC plus original four-in-one ESC[‡])	1	152.99
Motors	1860 KV brushless outrunner	4	263.80
Battery	Six-cell (6S) 22.2 V, 1400 mAh class LiPo	1	168.00
Power connector	AMASS XT60 pigtail pair	1	2.75
Hardware kits	M3 screw assortments (two vendor line items combined)	2 kits	5.73
Miscellaneous laboratory support			
Solder	0.8 mm leaded wire spool	1	16.90
Battery charger	“T6” class charger for field packs	1	169.15
Microcontroller	STM32F103C6T6 module (invoice line item)	1	16.98

Table 8: Labor cost worksheet (placeholder—replace hours and salary assumptions).

Partner	**Role focus**	**Hours**	**Rate (USD/h)**	**Cost (USD)**
Zhu Zhengyu	Wristband firmware / ESP32	—	—	—
Chen Zhenbo	RF link / ELRS	—	—	—
Chen Renang	Flight test / commissioning	—	—	—
Guo Jintu	Mechanical / integration	—	—	—
			Total labor (hours $\times$ rate $\times 2.5$)	—

8 Conclusions

This project demonstrated that a voice-command interface built on commodity offline recognition hardware can control a quadrotor drone through the full mission cycle of takeoff, directional flight, and landing without a conventional RC transmitter. The ESP32 and VC02 combination achieved the required command latency budget, the ELRS link maintained reliable packet delivery at 50 Hz across all flight sessions, and the SpeedyBee F405 V5 platform proved stable on factory PID defaults without manual tuning.

8.1 Summary of Accomplishments

The following objectives from the high-level requirements were met:

- A seven-word offline voice command vocabulary was implemented and mapped to flight-controller CRSF channel sequences with correct DVB-S2 framing and timing.
- The ELRS RF link was established using a RadioMaster RP2 reflashed as a CRSF transmitter, achieving automatic pairing via Binding Phrase without physical button binding.
- Stable hover was maintained for multiple seconds after each takeoff command, and directional pulse maneuvers produced observable displacements.
- Barometric altitude telemetry was decoded from the CRSF downlink and displayed on the OLED throughout flight, providing the ground operator with real-time situational awareness.
-

8.2 Ethical Considerations

This project involved repeated outdoor and indoor flight of a motorized aerial vehicle with rotating blades capable of causing injury. The IEEE Code of Ethics [2] requires engineers to hold the safety of the public paramount and to be honest about the limitations of their work. In adherence to these principles, all flight tests were conducted in restricted, supervised areas with bystanders kept outside the rotor arc. The voice control system incorporates a software interlock that rejects directional commands unless the takeoff sequence has already been completed, preventing unintended ground-level motor commands. Future deployment in public spaces would require regulatory compliance with local UAV operating rules and additional hardware failsafes such as a geofence or automatic return-to-home function.

The project has broader relevance in the context of accessible human–machine interfaces. Voice-driven UAV control reduces the physical skill barrier for operators with limited manual dexterity and opens potential applications in disaster response and remote inspection where operator hands may be otherwise engaged. At the same time, increased accessibility of UAV control raises societal concerns around privacy and airspace safety that future iterations of this work should address through both technical design choices and engagement with regulatory frameworks.

8.3 Future Work

Several improvements would increase the practical utility of the system.

Higher command update rate. The current 50 Hz CRSF output rate is sufficient for stable flight but is below the 250 Hz or 500 Hz rates achievable with ELRS. Increasing the rate would reduce the latency between a voice command being received and the first corresponding CRSF frame reaching the aircraft.

Continuous control modality. Replacing discrete pulse commands with proportional control—for example, by mapping the operator's head orientation or a wrist-worn IMU to roll and pitch channels in real time—would enable more precise maneuvering and altitude hold.

Safety failsafes. A hardware kill-switch input to the ESP32 and a CRSF failsafe channel configuration on the flight controller would ensure the motors disarm if the voice link or USB power to the ground unit is interrupted unexpectedly.

Outdoor range characterization. All verification flights were conducted at short range. A systematic link-budget test at increasing distances would characterize the operational envelope of the RP2-based transmitter at the ISM_2400 regulatory domain power level.

References

[1] ExpressLRS Contributors. “”ExpressLRS: Open Source RC Link”,” Accessed: May 1, 2026. [Online]. Available: https://github.com/ExpressLRS/ExpressLRS

[2] IEEE. “”IEEE Code of Ethics”,” Accessed: May 1, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A Requirement and Verification Table

Table 9 lists every requirement from Table 1 alongside the verification method used and the result.

Table 9: Requirement and Verification Table.

Req.	Verification method	Target	Result
R1	Time five utterances of each command; record UART byte timestamp via ESP32 serial monitor.	$<$1 s	*TODO*
R2	Logic analyzer on GPIO 25; measure time from VC02 byte in ISR to frame first bit on wire.	$<$40 ms	Pass ($\leq$20 ms)
R3	Logic analyzer inter-frame period on GPIO 25 TX line.	$\geq$50 Hz	Pass (50 Hz)
R4	Record hover duration after each takeoff command across all flight sessions.	$\geq$5 s	*TODO*
R5	Observe drone displacement during forward/backward voice commands in open area; estimate distance by reference marker.	Observable	*TODO*
R6	Count landing attempts vs. successful disarms (motors stop, craft does not invert or crash).	100%	*TODO*
R7	Confirm OLED altitude reading is non-zero and updating during each flight session.	Continuous	*TODO*