Dwight Look College of ENGINEERING
TEXAS A&M UNIVERSITY

Team 51: Radio Mobile Foxbot
Bi-Weekly Update 4

Brady Lagrone
Sophia Panagiotopoulos
Miguel Segura

Sponsor: Kevin Nowka
TA: Fahrettin Ay

Project Summary

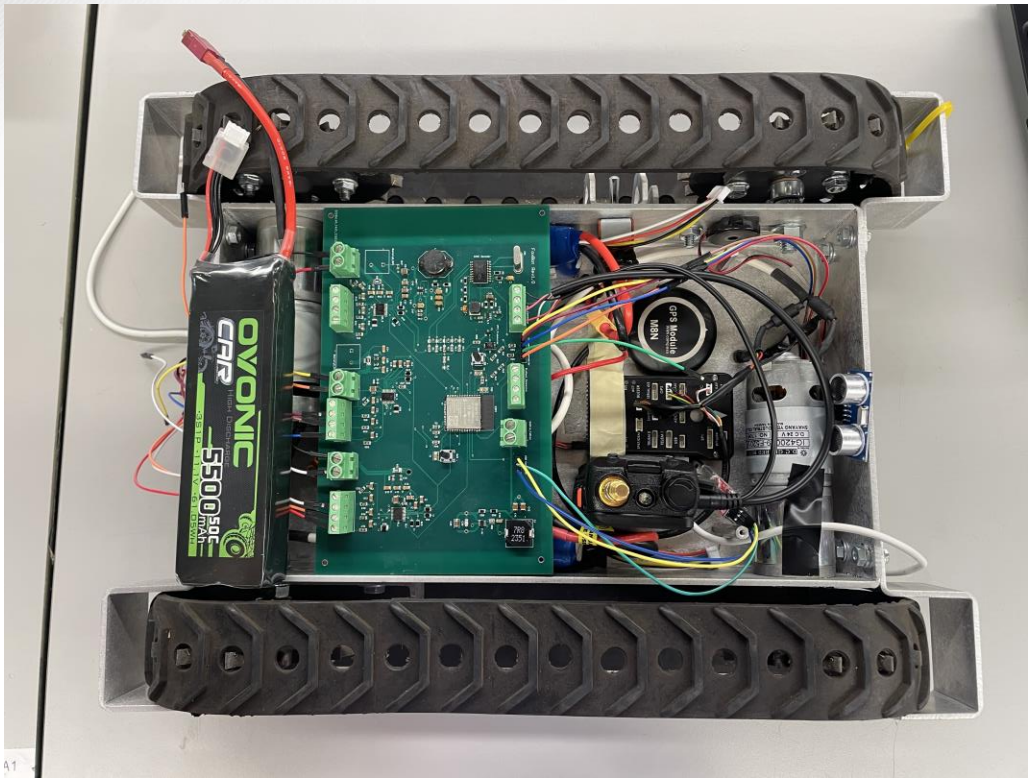
Purpose:

- Amateur Radio Directional Finding (ADRF) is a sport in which an individual tries to locate a hidden transmitting device. Our goal is to “hide” from an adversary Houndbot, who will be trying to locate our robot. Additionally, our robot will increase the potential of ARDF training and aid in the engineering of directional antennas.

Proposal:

- We have created a radio system coupled with a mobile robot that receives DTMF signals and preforms various functions based on the decoded signal.

Integrated System Diagram



- PCB, Radio, and PixHawk general layout integration
- The 5500mAh battery will be placed below PCB in final design with casing made to hold the other internals that consist of the Radio, UltraSonic Sensors Mounts, and GPS Mount.
- Wires will be sorted and changed with finalized integration of design.

Project Timeline

Subsystem Designs and Testing (completed 01/27)	Integration of power and radio subsystem PTT (completed 2/9)	Integration of MCU and Motor Driver (to complete by 2/14)	Integration of power and radio ADC (to complete by 2/14)	All systems have been integrated (to complete by 3/7)	System Testing (to be completed 3/17)	Demo and Report (to complete by 4/14)

Miguel Segura

Accomplishments since last update 25 hrs of effort	Ongoing progress/problems and plans until the next presentation
• Configured proper parameter values to setup into AutoPilot Mode. • Integrated Motors, PixHawk, and ESP32 with each other to have basic communication with each other. • PixHawk fully controls motors with the ESP32 working as a Ground Controller. • Planned each mode of operation with the components needed (with Brady) • Made "mission planner" functions to have the ESP32 working as the Ground Controller.	• AutoPilot works; however, the GPS seems to be having issues with buildings obstructing the Misson Planned path. Further testing needs to be done in an area with less obstruction. • Implement the designated modes we have determined. Alongside the radio transmission schedule.

Miguel Segura

```c
void set_manual_mode(void) {
    send_command_long(
        MAV_CMD_DO_SET_MODE,
        MAV_MODE_FLAG_CUSTOM_MODE_ENABLED,  // Enable custom mode (param1)
        MANUAL_MODE,                        // Manual mode (param2)
        0, 0, 0, 0, 0                       // Unused parameters
    );

    ESP_LOGI(TAGG, "Set MANUAL mode");
}
```

Manual mode: Works as an OFF switch for the motors

```c
void set_auto_mode(void) {
    send_command_long(
        MAV_CMD_DO_SET_MODE,
        MAV_MODE_FLAG_CUSTOM_MODE_ENABLED,  // Enable custom mode (param1)
        AUTO_MODE,                          // Auto mode (param2)
        0, 0, 0, 0, 0                       // Unused parameters
    );

    ESP_LOGI(TAGG, "Set AUTO mode");
}
```

AutoPilot mode: Has full control of motor system

```c
void start_mission(void) {
    send_command_long(
        MAV_CMD_MISSION_START,
        0, 0, 0, 0, 0, 0, 0                 // All params unused for this command
    );

    ESP_LOGI(TAGG, "Started mission");
}
```

Start Mission: Sends signal to start mission. The mission data is received through the assembled UART telemetry connection of the PixHawk and ESP32

- Functions made for the different modes that need to be established for the ESP32 to communicate with the PixHawk.
- PixHawk has full control of the Sabertooth 2x12 motors with Mixed Motor Control. Obstructions with GPS to fully verify auto pilot capabilities. Further testing required
- Verified old UltraSonic sensor code works with final PCB design with Brady.

```c
void receive_heartbeat(const mavlink_heartbeat_t *heartbeat) {
    uint32_t now = xTaskGetTickCount() * portTICK_PERIOD_MS;
    bool was_connected = pixhawk_connected;

    // Update connection status
    pixhawk_connected = true;
    last_pixhawk_heartbeat = now;

    // Log connection establishment
    if (!was_connected) {
        ESP_LOGI(TAGG, "Pixhawk is Connected");
    }

    // Display heartbeat information every 15 second
    static uint32_t last_heartbeat_display = 0;
    if (now - last_heartbeat_display > 15000) {

        ESP_LOGI(TAGG, "Mode: %s", get_mode_string(heartbeat->custom_mode));
        ESP_LOGI(TAGG, "Status: %s", get_system_status_string(heartbeat->system_status));
        ESP_LOGI(TAGG, "Type: %d, Autopilot: %d", heartbeat->type, heartbeat->autopilot);
        ESP_LOGI(TAGG, "Base Mode: 0x%02X", heartbeat->base_mode);

        last_heartbeat_display = now;
    }
```

Heartbeat Send and Receive: Functions made to maintain and verify PixHawk to ESP connection

Brady Lagrone

Accomplishments since last update 20 hrs of effort	Ongoing progress/problems and plans until the next presentation
• All components powered and operate on built converters • Intgrated radio DTMF of startup and mode selection with ESP • Finished integration of status mode of operation • Planned each mode of operation with the components needed	• Radio and ESP integration with Pixhawk pathing and control • PCB configuration and connections • Developing each mode of operation

Brady Lagrone

Health mode Example

```
Binary: 1010, Decimal: 10
Binary: 1010, Decimal: 10
Binary: 0111, Decimal: 7
This is ultrasonic sensor
Letter: O, Morse Code: ---
Letter: B, Morse Code: -...
Letter: J, Morse Code: .---
  (Word Space)
Letter: D, Morse Code: -..
Letter: e, Morse Code: .
Letter: t, Morse Code: -
Letter: e, Morse Code: .
Letter: c, Morse Code: -.-.
Letter: t, Morse Code: -
This is back in mode 4
```

Initial Startup Sequence

```
I (334) gpio: GPIO[14]| InputEn: 1| OutputEn: 0| OpenDrain: 0| Pullup: 0| Pulldown: 0| Intr:0
Binary: 0000, Decimal: 0
Binary: 0000, Decimal: 0
Binary: 0000, Decimal: 0
Binary: 0000, Decimal: 0
Binary: 0000, Decimal: 0
Binary: 0000, Decimal: 0
Binary: 1010, Decimal: 10
I (35844) I2C: I2C initialized successfully!
I (36444) BQ76920: Wake-up pulse sent to TS1
I (36944) i2c_restart: Scanning I2C bus...
I (36944) i2c_restart: Found device at address: 0x08
I (36964) i2c_restart: Scan complete.
I (37464) gpio: GPIO[33]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 1| PuI (37464) MAVLINK_ESP32: UART initialized
Battery Monitor System is good
Letter: B, Morse Code: -...
Letter: M, Morse Code: --
  (Word Space)
Letter: G, Morse Code: --.
Letter: o, Morse Code: ---
Letter: o, Morse Code: ---
Letter: d, Morse Code: -..
Binary: 1010, Decimal: 10
Not an actionable task
This is back in mode 4
```

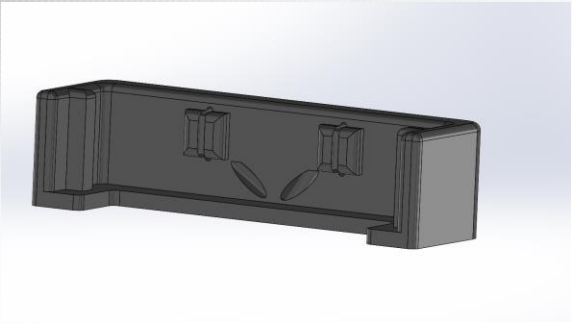
Modes of Operation:

- Users sends DTMF signal for each
- Need to send heartbeat to Pixhawk each mode
- Check Ultrasonic During modes of operation and alert if object
    - First mode (Beep on off moving on path PTT also) (20s)
    - Second mode (Emitting signal, pause ”hide”, repeat) (10s, 10s)
    - Third Mode (Beep on off Stationary)(20s)
    - Fourth Mode (Continuous PTT and stationary) (20s)
    - Health mode (Check Bot)
        - Battery, GPS, health, ultrasonic check
        - Need to have a reset value
- KJ5HZT (call sign)

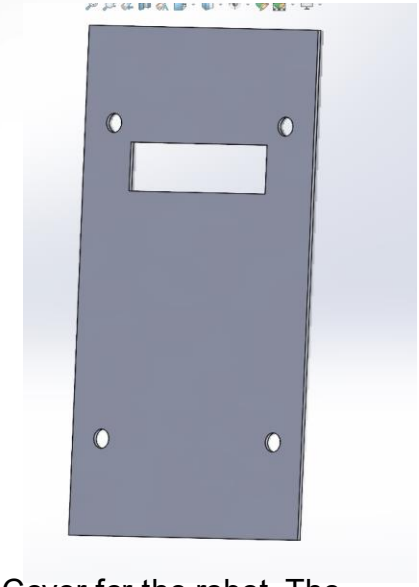
Sophia Panagiotopoulos

Accomplishments since last update 20 hrs of effort	Ongoing progress/problems and plans until the next presentation
- Finished soldering PCB and testing that all circuits are functional. - Radio can arm and disarm the Pixhawk, beginning motor integration - Tested radio control outside for arming and disarming Pixhawk and for reading battery health. - Made 3D printed battery connector for radio to enable charging through the PCB. - Made 3D printed cover for robot chassis and PCB case	- Continue working integrating radio with Pixhawk for motor control. - Finalize modes of operation. - Test system outside, making sure that GPS can be read. - Test battery connection between printed part and 7.4V line. - Continue implementing packaging for PCB and overall robot chassis.

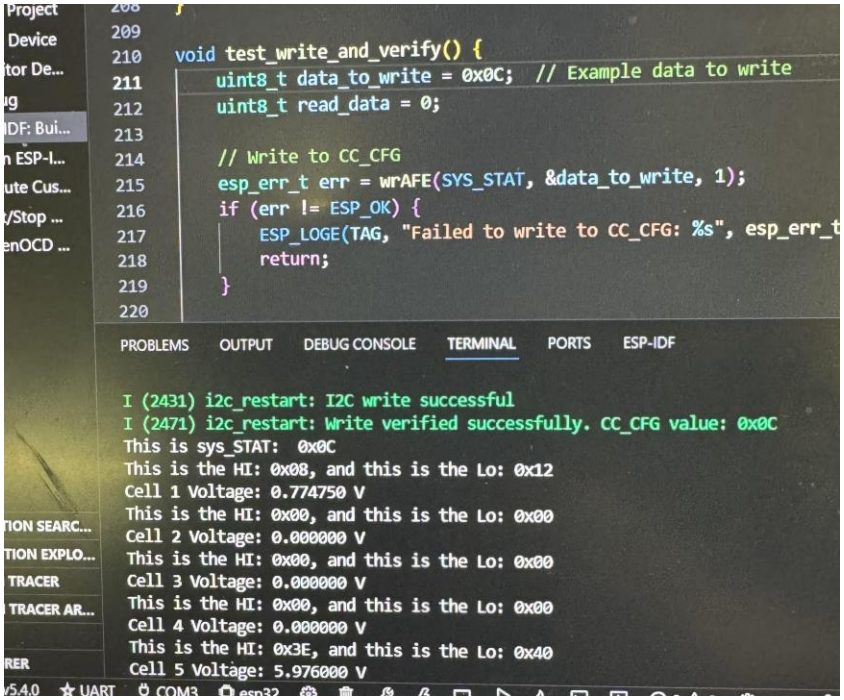
Sophia Panagiotopoulos



Battery component that will allow the 7.4V line on our PCB to charge the Foxbot radio



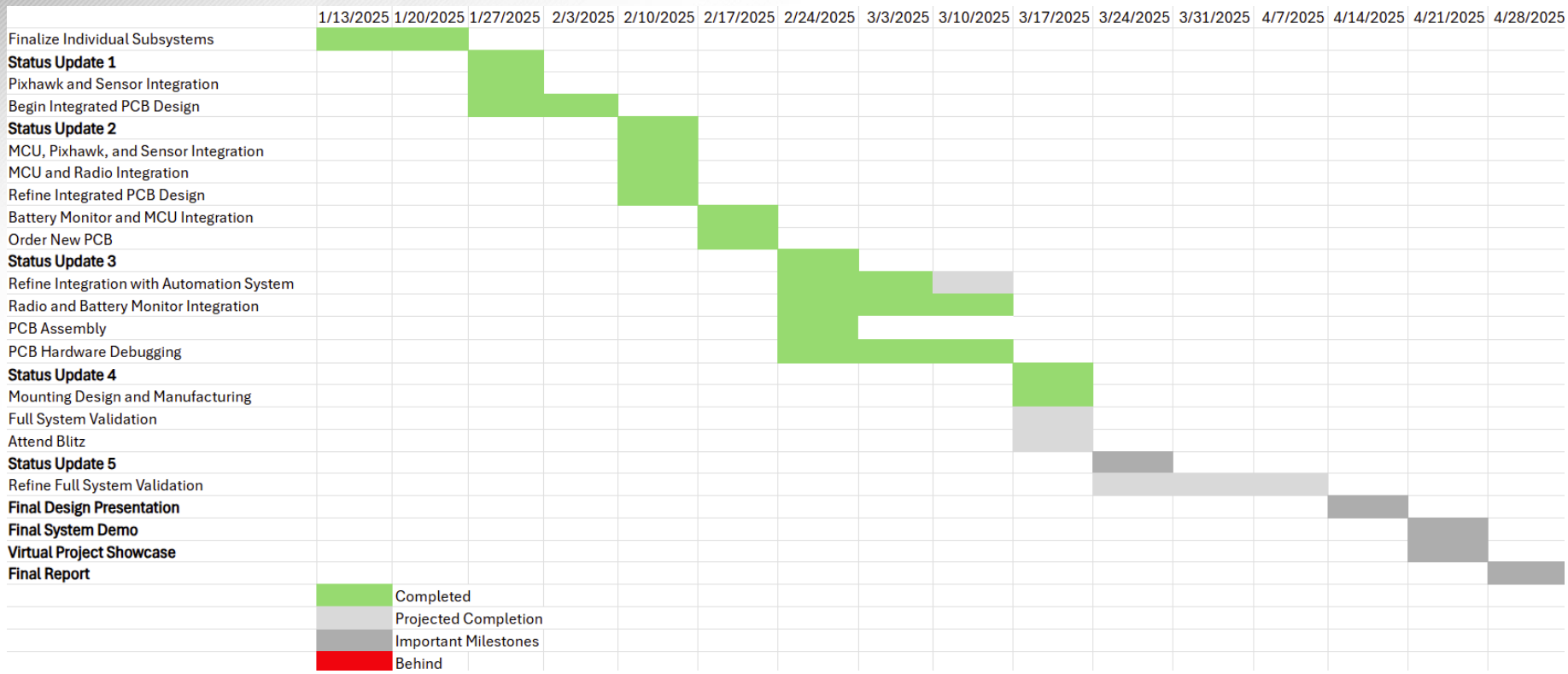
Cover for the robot. The opening is for wires to come out from under the cover and into the PCB



Output showing battery cells can be read, allowing for battery health updates via radio control

Finished PCB with full functionality

Execution Plan



Validation Plan

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
3.2.1.1	Transmisison Range	Radio on foxbot is able to pick up signals from user radio within the specified foxhunt region	Radio on the foxbot is able to recognize a sent signal from the user radio by outputting the tone from its speaker.	TESTED	Sophia
3.2.1.2	Operation Time	The foxbot will operate for at least 30 mins	The batteries are able to run the system for 1 hour when left alone.	TESTED	Brady
3.2.1.3	DTMF Decoding Accuracy	A binary value, output of the decoder, will correspond to the keypad number sent by the user radio	LEDs will show the bits that are high or low, indicating the value in binary.	TESTED	Sophia
3.2.1.4	Intutiveness of System	The system shall be straightforward and the user shall be able to troubleshoot if there is an issue	It should take the user not more than 10 minutes to load the path and start up the foxbot.	UNTESTED	Full Team
3.2.1.5	Pathing Accuracy	The foxbot will stay on the user decicded path and will avoid obstacles including trees, ditches, and moving objects.	Place the foxbot in a location and it should be able to determine its path while staying in the programmed range. It will be able to reroute if any obstacles are detected.	UNTESTED	Miguel
3.2.1.6	Morse Decoding Accuracy	The user radio will be able to pick up the transmitted signal from the foxbot radio and decode it using a morse decoding app.	App on phone will be able to decode the sent signal corresponding to a voltage reading or some other message.	TESTED	Sophia
3.2.2.1	Mass	The foxbot will not exceed 7lbs	Weigh the system once all of the componets are mounted.	UNTESTED	Full Team
3.2.2.2	Mounting	to the top of the chassis	be held down securely to ensure it can withstand hostile movements.	TESTED	Full Team
3.2.2.3	System Packaging	The radio, PCBs, and MCU will be held in custom protective cases	Cases should hold the components and protect them from enviornmental factors like heat, humidity, and water.	UNTESTED	Full Team
3.2.3.1	Input Voltage (Radio)	The Baofeng will receive an input voltage fo 7.4 V at a current of 1780mA	Use a multimeter to validate input voltage levels.	TESTED	Brady
3.2.3.2	Input Voltage (ESP)	The ESP will receive an input voltage of 3.3V at a current of 160mA	Use a multimeter to validate input voltage levels.	TESTED	Brady
3.2.3.3	Voltage Monitoring	The voltage of the batteries will be read by a GPIO pin of the ESP and will detatch the battery from the battery monitor if the voltage becomes too low.	The voltage is read by the ESP and shows up in the terminal. LEDs can be used to show the battery level as a percentage of a fully charged rating. A trasistor will be activated to break the circuit when the battery levels drop below a usable voltage.	TESTED	Brady and Miguel
3.2.4.1	Enviornmental Resistance	The foxbot shall be able to traverse flat terrain and withstand temperatues in the range of 5°C to 40°C	The system will be placed in various enviornments ensuring the monitors can traverse the terrain.	UNTESTED	Full Team
	Full System Demo	The foxbot will self automate a loaded path from the pixhawk while communicating with the user through DTMF signals and morse. It will hide from the houndbot and will be able to run for 1 hour while avoiding obstacles	Foxbot is placed at a starting location, follows automated path with sending and receiving signals, and avoid getting caught by the houndbot.	UNTESTED	Full Team