Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Team 51: Radio Mobile Foxbot
Bi-Weekly Update 2

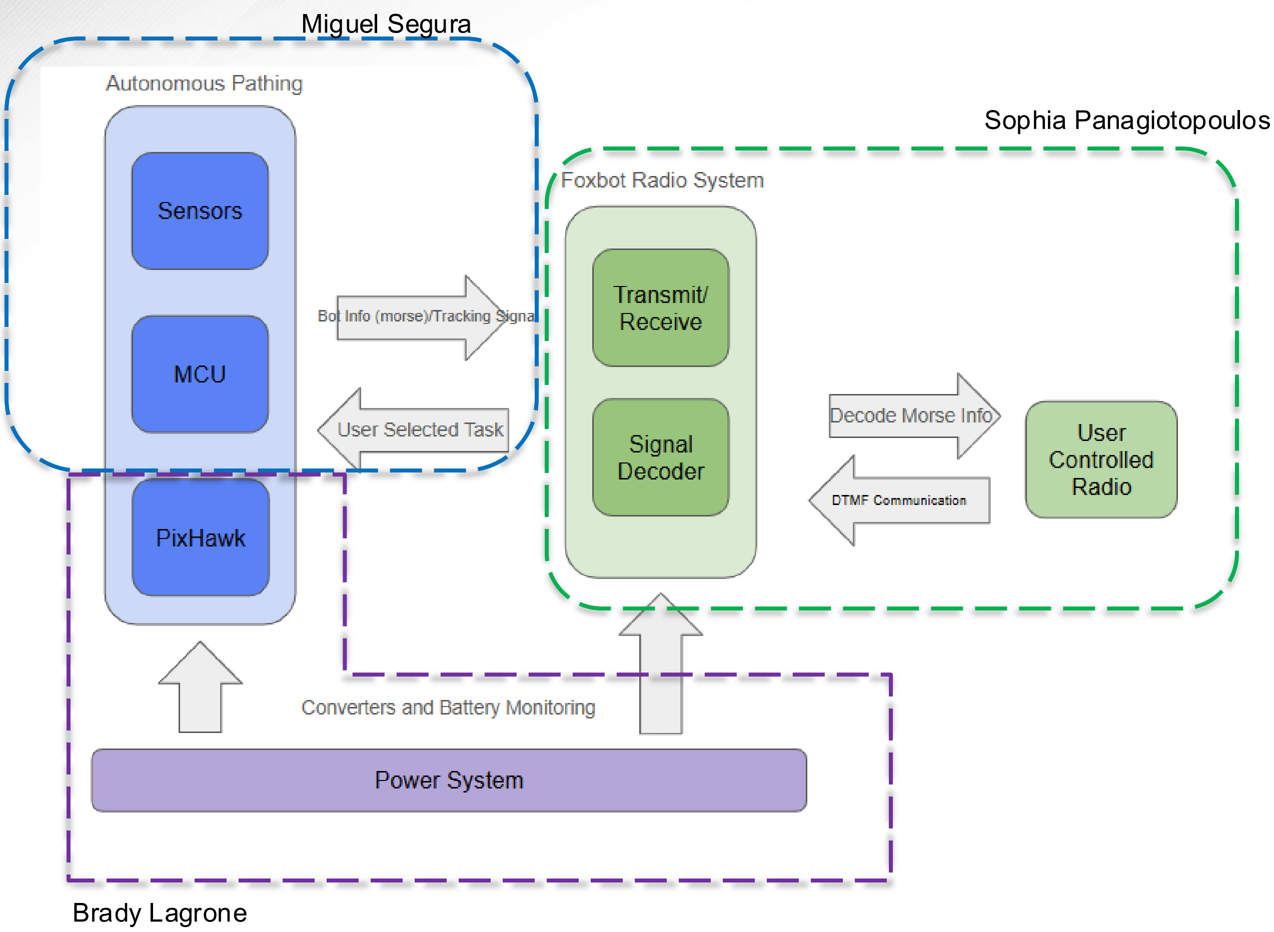
Brady Lagrone
Sophia Panagiotopoulos
Miguel Segura

Sponsor: Kevin Nowka and David Gent
TA: Fahrettin Ay

Project Summary

- Purpose:
- Amateur Radio Directional Finding (ARDF) is traditionally done with a stationary transmitter, limiting the training abilities.
- Transmitted signals are non-adjustable with little variability for the user.
- Our Proposal:
- We will have a mobile robot chassis transmitting user selected signals.
- This will increase the potential of ARDF training.

Project/Subsystem Overview



Subsystem Goal Overview:

Autonomous Pathing System:

- Robot can arrive at user decided destination while avoiding objects by using sensors and Pixhawk.
- MCU will send analog signal to radio transmit while controlling PTT function.
- MCU will receive and control batteries properties

Power System:

- Lippo batteries will be converted for component ratings.
- Battery monitor will send alert to MCU when voltage ratings are low.

Radio System:

- User can send DTMF tone to Foxbot which can be decoded to determine user selected settings (transmit for houndbot or battery health).
- Radio will transmit the formulated signal by the MCU. If the user has requested battery health, morse will be decoded.

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)	Final Integration(to complete by 3/7)	System Testing (to be completed 3/17)	Demo and Report (to complete by 4/14)

MCU Subsystem

Miguel Segura

Accomplishments since last update ~16 hrs of effort	Ongoing progress/problems and plans until the next presentation
• Armed the PixHawk to Mission Planner to provide the autonomous design. • Wrote code to make LEDs interact as "motors" to measure their PWM signal from Mission Planner through the PixHawk.	• Integrate PixHawk to Sabertooth and perform test runs through Mission Planner • Further develop simulation protocol and get functionality across the system.

MCU Subsystem

Miguel Segura

```c
// RMT receive callback function
static bool rmt_rx_done_callback(rmt_channel_handle_t channel,
                                const rmt_rx_done_event_data_t *edata,
                                void *user_data) {
    // Calculate pulse width in microseconds from RMT ticks
    uint32_t pulse_width = edata->received_symbols[0].duration0 *
                          (1000000 / RMT_RESOLUTION_HZ);

    // Update appropriate global variable based on channel
    if (channel == rmt_chan1) {
        pwm1_width = pulse_width;  // Store Motor1's pulse width
    } else if (channel == rmt_chan2) {
        pwm2_width = pulse_width;  // Store Motor2's pulse width
    }

    // Notify waiting task that data is ready
    BaseType_t high_task_wakeup = pdFALSE;
    vTaskNotifyGiveFromISR((TaskHandle_t)user_data, &high_task_wakeup);
    return high_task_wakeup == pdTRUE;  // Return whether task switch is needed
}
```

```c
// LED update task
void update_leds(void *pvParameters) {  // Task function for LED control
    rmt_channel_handle_t channel = (rmt_channel_handle_t)pvParameters;  // Get channel from params
    rmt_receive_config_t rx_config = {
        .signal_range_min_ns = 1000,    // Minimum valid pulse width (1µs)
        .signal_range_max_ns = 2500000  // Maximum valid pulse width (2.5ms)
    };

    while (1) {  // Infinite task loop
        // Start receiving RMT data with error checking
        ESP_ERROR_CHECK(rmt_receive(channel, NULL, 0, &rx_config));

        // Wait for receive completion notification
        ulTaskNotifyTake(pdTRUE,        // Clear notification on entry
                        portMAX_DELAY); // Wait indefinitely

        // Update LED based on current PWM value
        if (channel == rmt_chan1) {
            // Set LED1 based on Motor1's PWM deviation from neutral
            gpio_set_level(LED1_PIN,
                          (abs((int)pwm1_width - PWM_NEUTRAL) > TOLERANCE) ? 1 : 0);
        } else {
            // Set LED2 based on Motor2's PWM deviation from neutral
            gpio_set_level(LED2_PIN,
                          (abs((int)pwm2_width - PWM_NEUTRAL) > TOLERANCE) ? 1 : 0);
        }
    }
}
```

Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Power Subsystem

Brady Lagrone

Accomplishments since last update 30 hrs of effort	Ongoing progress/problems and plans until the next presentation
• Established I2C communication with BQ7692003PWR battery monitor • Wrote and tested code for the ADC detecting levels 0-9, for integrating with the radio subsystem • Wrote code for to produce morse code from the generated sine wave, and tested and integrated with radio subsystem	• Continue and finish PCB integration between power and radio subsystem • Establish more modes of operation according to digits pressed

Power Subsystem

Brady Lagrone

```c
// Voltage thresholds (Need to change)
float voltageLevels[NUM_LEVELS] = {2.12, 0.72, 0.78, 1.57, 3.3, 1.77, 1.87, 2.21, 1.28, 2.05};

// Monitor ADC in a FreeRTOS task
void vTaskMonitorADC(void *pvParameters) {
    adc_init();

    while (1) {
        if(adc_action_flag == -1){
            float voltage = read_adc_voltage();
            printf("ADC Voltage: %.2fV\n", voltage);

            // Check which voltage level the ADC is closest to
            for (int i = 0; i < NUM_LEVELS; i++) {
                if (voltage >= (voltageLevels[i] - TOLERANCE) && voltage <= (voltageLevels[i] + TOLERANCE)) {
                    if(i != adc_prev_flag){  //Prevent code from flagging the same voltage level more than once
                        adc_action_flag = i; // Set flag to detected action number
                        printf("Action Detected: %d\n", adc_action_flag);
                        break; // Stop checking once a match is found
                    }

                }
            }

        }
        // Delay before next reading
```

```c
else if (adc_action_flag == 1){
    vTaskDelay(pdMS_TO_TICKS(500));
    //Set GPIO pin to set transistor to open allowing PTT
    gpio_set_level(Trans, 1);
    printf("Executing Action %d, reading Cell1 Voltage \n", adc_action_flag);
    // Read the voltage of Cell 1
    float voltage = read_cell_voltage(VC1_HI, VC1_LO);
    // Convert the voltage value to a string
    snprintf(message, sizeof(message), "%.2f", voltage);

    // Log the voltage value for debugging
    printf("Cell 1 Voltage: %s V\n", message);

    // Play Morse code message
    playMorseString(message, frequency);

    //Set dac to zero
    dac_output_voltage(DAC_PIN, 0);

    //Set GPIO pin to set transistor to close allowing PTT
    gpio_set_level(Trans, 0);
    adc_prev_flag = adc_action_flag;          //set to previous value to prevent looping
    adc_action_flag = -1;                     //reset the action flag to -1
}
```

ADC Monitor and changing what flag is triggered, for now 1 reads and prints (in morse) voltage of cell one from the battery monitor

```c
// Function to read the cell voltage
float read_cell_voltage(uint8_t vc_hi_addr, uint8_t vc_low_addr) {
    // Read the values of VC_HI and VC_LOW registers
    uint8_t vc_hi = read_register(vc_hi_addr);
    uint8_t vc_low = read_register(vc_low_addr);
    printf("This is the HI: 0x%02X, and this is the Lo: 0x%02X \n", vc_hi, vc_low);

    // Combine VC_HI and VC_LOW into a 16-bit value
    uint16_t combined_value = (vc_hi << 8) | vc_low;

    // Mask out bits <15:14> to use only bits <13:0>
    combined_value = combined_value & 0x3FFF; // 0x3FFF = 0b0011111111111111

    // Multiply the combined value by 375 microvolts (375e-6 volts)
    float voltage = combined_value * 375e-6;

    return voltage;
}
```

Read registers of cells and calculates the voltage representation

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Radio Subsystem

Sophia Panagiotopoulos

Accomplishments since last update 30 hours of effort	Ongoing progress/problems and plans until the next presentation
• Completed a bit weighted voltage divider to minimize pins needed from the ESP with the ADC being able to recognize different tones sent. • Coded a sine wave generator for the DAC that can transmit when a voltage is read from the ADC, allowing for radio transmission and MCU integration. • PTT is enabled by IO pin when ADC reads a value and DAC wants to transmit.	• Work on shutting down decoder board when PTT is enabled through something like a high-side driver circuit or op-amp. • Refine voltage divider since some values are too close for the ESP to differentiate between.

Radio Subsystem Figure

Sophia Panagiotopoulos

```
ADC Voltage: 0.00V
ADC Voltage: 0.00V
ADC Voltage: 0.00V
ADC Voltage: 0.00V
ADC Voltage: 0.79V
Action Detected: 2
Executing Action 2, reading Cell2 Voltage
This is the HI: 0x10, and this is the Lo: 0x44
Cell 2 Voltage: 1.56 V
Letter: 1, Morse Code: .----
Letter: ., Morse Code: .-.-.-
Letter: 5, Morse Code: .....
Letter: 6, Morse Code: -....
```

Image 1: ADC reading DTMF Tone 2

```c
// Generate sine wave lookup table
void generateSineTable() {
    for (int i = 0; i < SINE_RESOLUTION; i++) {
        sineTable[i] = (uint8_t)(127.5 + 127.5 * sin(2 * M_PI * i / SINE_RESOLUTION));
    }
}

// Initialize DAC
void dac_init() {
    dac_output_enable(DAC_PIN); // Enable DAC channel
}

// Output a value to the DAC
void dac_output(uint8_t value) {
    dac_output_voltage(DAC_PIN, value); // Set DAC output voltage (0-255)
}

int delayTime = 1000000 / (frequency * SINE_RESOLUTION); // Microseconds per step
while (1) {
    // Generate sine wave
    for (int i = 0; i < SINE_RESOLUTION; i++) {
        dacOutput(DAC_PIN, sineTable[i]); // Output to DAC (simulated)
        vTaskDelay(pdMS_TO_TICKS(delayTime)); // Delay for frequency control
    }
}
```

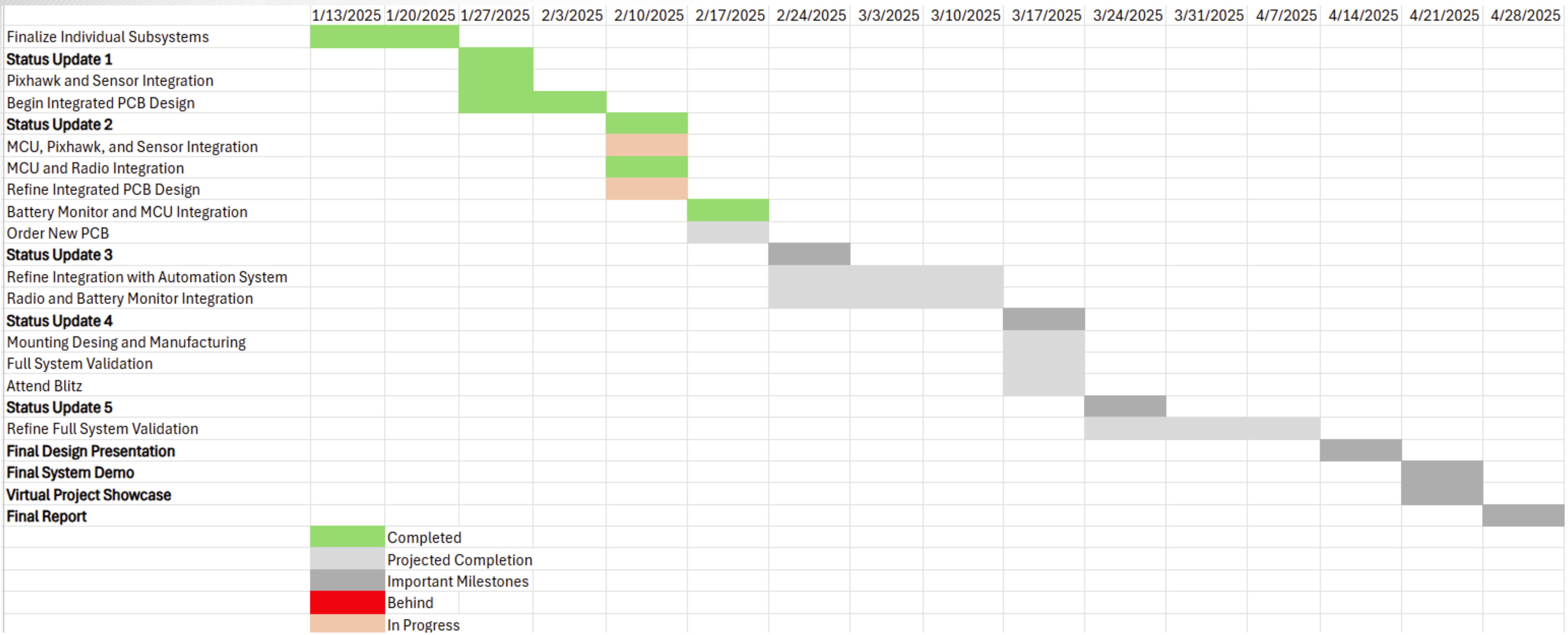
Image 2: Code to Generate Audio Frequency

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

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 1 hour	The batteries are able to run the system for 1 hour when left alone.	UNTESTED	Brady
3.2.1.3	DTMF Decoding Accur	A binary value, output of the decoder, will correspond to the keypad number sent by the	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 decioded 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	UNTESTED	Miguel
3.2.1.6	Morse Decoding Accur	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.	UNTESTED	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	The sensors shall be able to be mounted to the corners of the chassis and the radio will be	Attatch the sensors to the corners of the chassis and test for accuracy. The	UNTESTED	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 system if the voltage becomes too low.	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.	UNTESTED	Brady and Miguel
3.2.4.1	Enviornmental Resistar	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

Gantt Chart


1/13/2025
1/20/2025
1/27/2025
2/3/2025
2/10/2025
2/17/2025
2/24/2025
3/3/2025
3/10/2025
3/17/2025
3/24/2025
3/31/2025
4/7/2025
4/14/2025
4/21/2025
4/28/2025
Finalize Individual Subsystems
Status Update 1
Pixhawk and Sensor Integration
Begin Integrated PCB Design
Status Update 2
MCU, Pixhawk, and Sensor Integration
MCU and Radio Integration
Refine Integrated PCB Design
Battery Monitor and MCU Integration
Order New PCB
Status Update 3
Refine Integration with Automation System
Radio and Battery Monitor Integration
Status Update 4
Mounting Desing and Manufacturing
Full System Validation
Attend Blitz
Status Update 5
Refine Full System Validation
Final Design Presentation
Final System Demo
Virtual Project Showcase
Final Report
Completed
Projected Completion
Important Milestones
Behind
In Progress