



*Dwight Look College of*

**ENGINEERING**  
TEXAS A&M UNIVERSITY

# Team 14: RF Triangulation Bi-Weekly Update 3

Josh Broyles, Brandon Stokes,  
Jack Parkinson, Kathleen Hutchinson

Sponsor: Max Lesser  
TA: Souryendu Das

# Project Summary

Biologists have struggled being able to collect data on wildlife's location, habitat use, and breeding patterns without getting directly involved with the animal they're tracking.

**Radio Frequency Triangulation** allows a user to track a **known frequency** (such as a previously tagged animal) within the **triangulated area of three antennas** by using a motor to successfully **pinpoint the strongest signal**.



Helpful to study both invasive and threatened wildlife, our RF Triangulation system will focus on being able to accurately **track a transmitter within a 150 meter radius with >10% error**.



# Integrated System Diagram

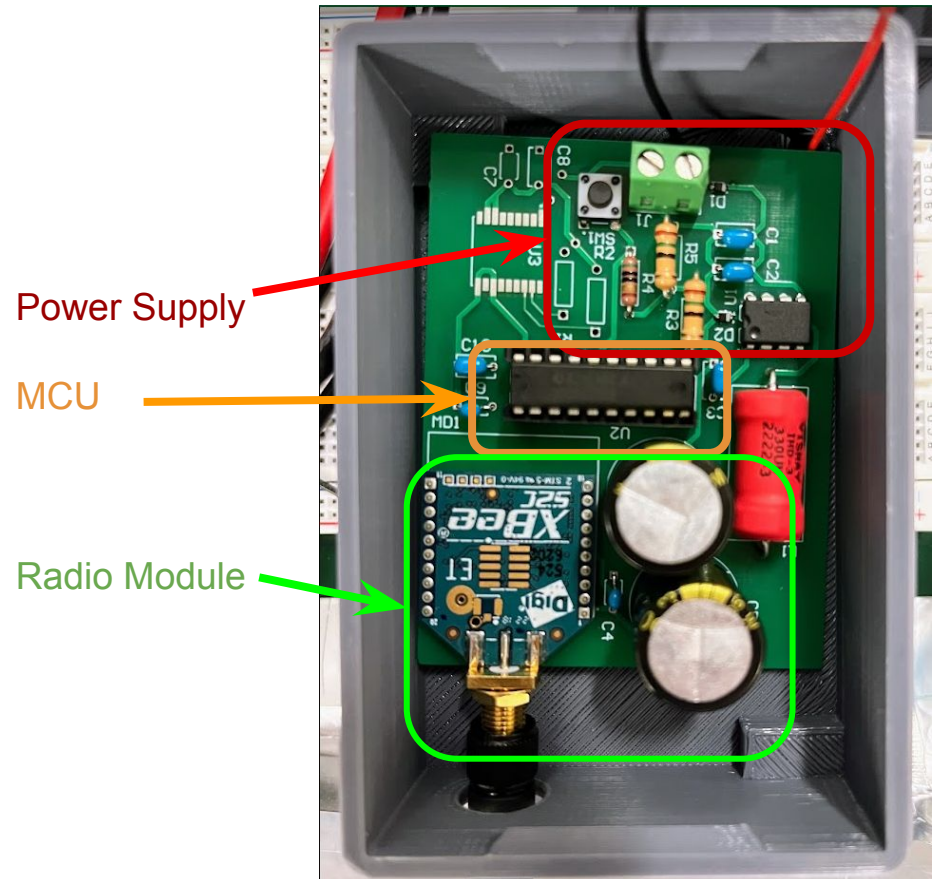
Ideally, our system will have one transmitter and three receivers.

The figure to the right shows a **completed transmitter**.

In the future, three receivers of *relatively the same form* will be created to connect to the transmitter.

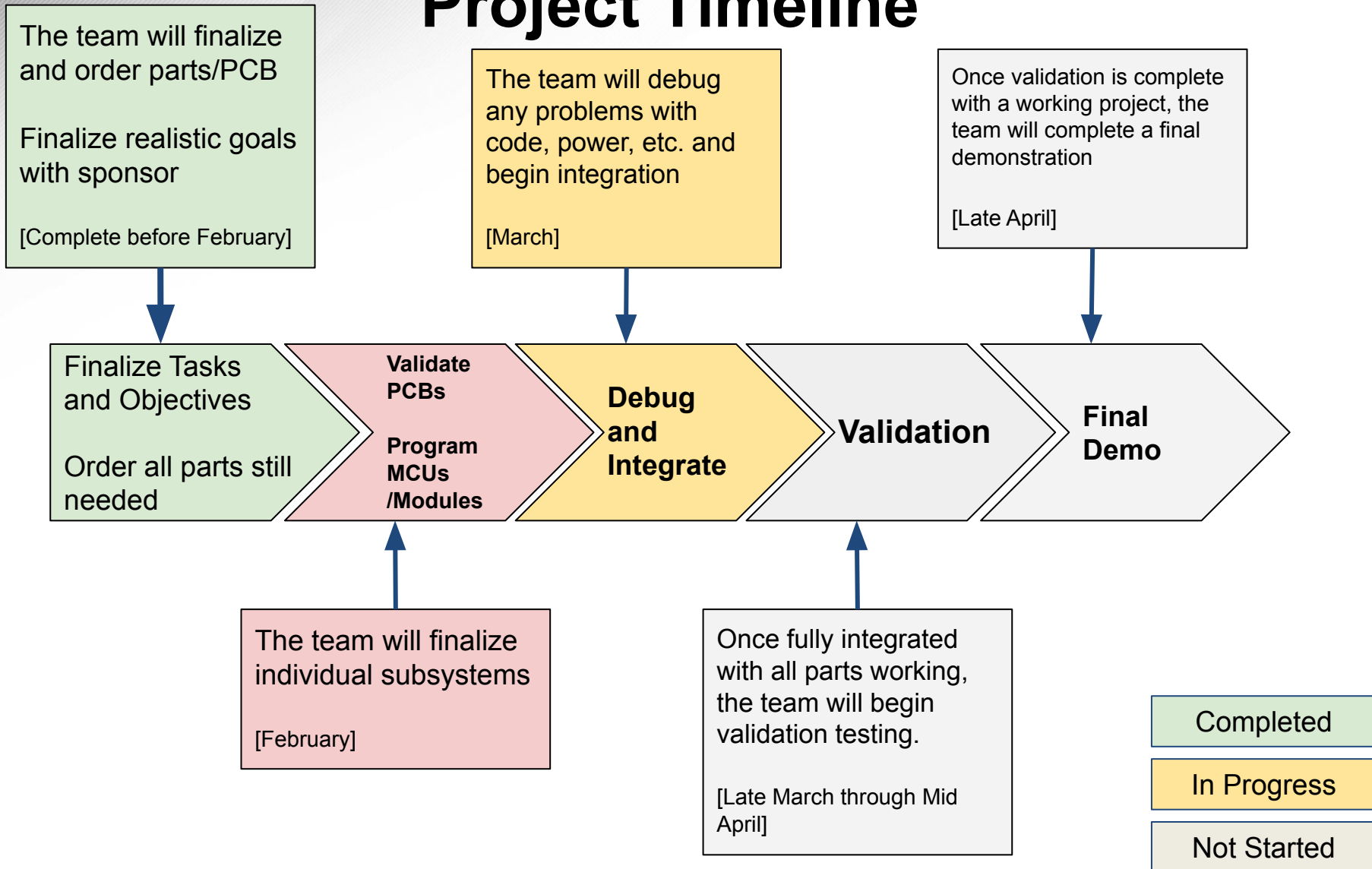
The only differences between will be *three added modules* that are essential to the *stepper motor* and data transmission through *LTE*.

***Due to power supply failure, the receiver is still in progress to be finished.***





# Project Timeline





# Transmitter

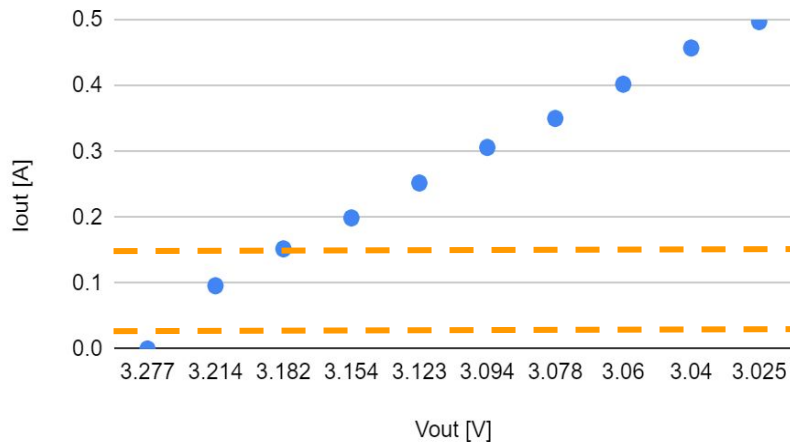
Josh Broyles

Accomplishments since last update 39 hrs of effort	Ongoing progress/problems and plans until the next presentation
Validated Transmitter Power  Completed Radio Housing  Working signal transmission to receiver side	N/A

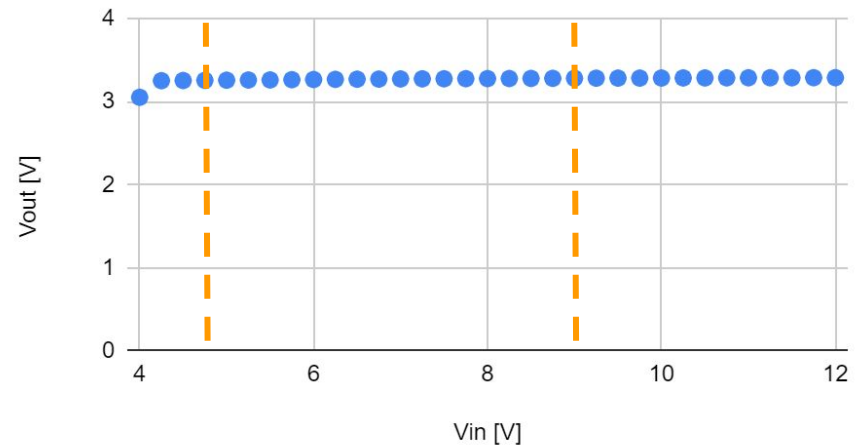
# Transmitter

Josh Broyles

Transmitter Load Regulation



Transmitter Line Regulation





# Receiver: Antenna & Motor

Jack Parkinson

Accomplishments since last update 26 hrs of effort	Ongoing progress/problems and plans until the next presentation
Finalized Antenna Design  Got Blinky code working on an offboard ESP32  Soldered on about half the PCB for testing	Write code for stepper motor based off of the blinky code  Get red badge  Buy parts for antenna  Help Kat with the PCB problems and integration between our systems



# Receiver: ESP32 & Modules

Kathleen Hutchinson

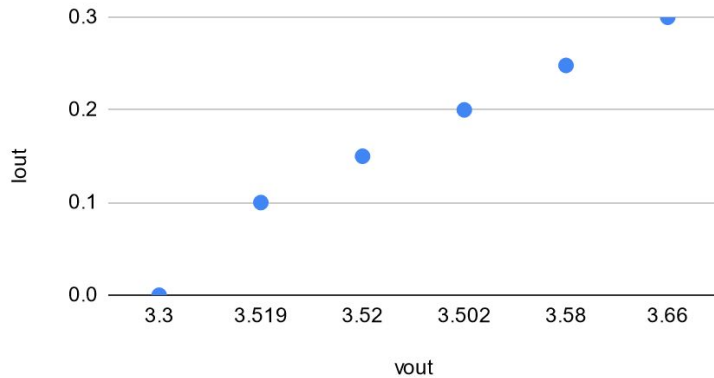
Accomplishments since last update 39 hrs of effort	Ongoing progress/problems and plans until the next presentation
Failed power validation  Connected Josh's XBEE, to my XBEE, through ESP32, onto Pycom to Brandon	Buying parts PCB design part 2  Problems: buck converter inductor XBee output



## Receiver: ESP32 & Modules

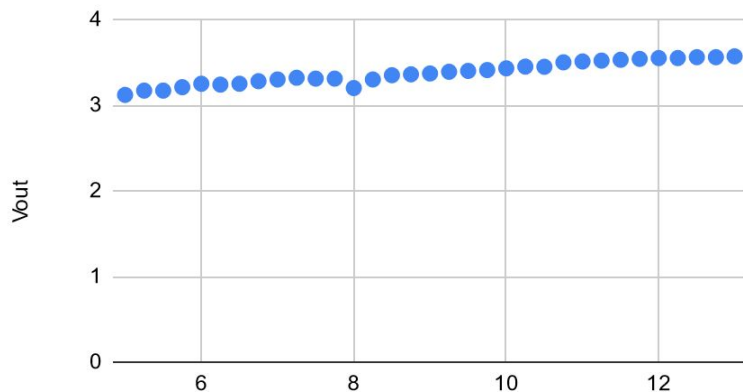
## Kathleen Hutchinson

## Receiver Load Regulation



vin	vout	lout
11.996	3.3	0
	3.519	0.1
	3.52	0.15
	3.502	0.2
	3.58	0.248
	3.66	0.3
	temp problem	0.35

## Receiver Line Regulation



The screenshot displays the Arduino IDE interface with the 'Serial Monitor' window open. The code in the background is for an ESP32 Dev Module, which is configured to use the 'USB Serial Device (COM6)' port. The code includes a setup function for UART pins and a loop function that prints data received from the UART. The Serial Monitor shows the output of the code, displaying the baud rate and the data received from the UART.

**Code (ESP32 Dev Module):**

```

15 Serial.begin(9600); // Serial monitor baud rate
16 // Starts UART peripheral
17 // MySerial.begin(baud rate, protocol format, RX pin, TX pin),
18 // SERIAL_BN1 means the data packet has 1 start bit, 8 data bits, 0 parity bits, and 1 stop bit
19 MySerial2.begin(9600, SERIAL_BN1, 16, 17);
20 MySerial1.begin(9600, SERIAL_BN1, RX1, TX1);
21
22
23 void loop() {
24 // Test message read
25 //MySerial2.print("Reading at 9600 baud! urnt2 \n"); // "sends" message out of TX pin
26 MySerial1.print("Reading at 9600 baud! urnt1 \n");
27
28 while((MySerial2.available() > 0) || (MySerial1.available() > 0)) { // while something is being communicate
29   pycomReceived = MySerial1.readStringUntil("\n"); // reads a string till it reads the character in the par
30   XBeeReceived = MySerial2.readStringUntil("\n"); // "
31
32   Serial.print("Printing UART1 RX: ");
33   Serial.println(pycomReceived); // prints the RXpin data to the serial monitor
34
35   Serial.print("Printing UART2 RX: ");
36   Serial.println(XBeeReceived); // prints the RXpin data to the serial monitor
37   MySerial1.print(XBeeReceived);
38 }
39 delay(1000); // delay 1 second
40 //Serial.print("Running..."); // print statement to show that its running
41
42

```

**Serial Monitor Output:**

```

Output Serial Monitor x
Message (Enter to send message to 'ESP32 Dev Module' on 'COM6') New Line 9600 baud

Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089

```

**Python Code (pycom.heartbeat):**

```

1 from machine import UART
2 import time
3 import pycom
4
5 pycom.heartbeat(False)
6
7 while True:
8   print("Running")
9   uart = UART(1, baudrate=9600)
10   strMsg = ""
11   uart.write("esp32 check \n")
12   time.sleep(0.5)
13   if uart.any() > 0:
14     strMsg = uart.read()
15     print(strMsg)
16     strMsg = str(strMsg, 'utf-8')
17     print(strMsg)
18   else:
19     strMsg = "noData"
20     print(strMsg)
21   time.sleep(2)

```

```

Output    Serial Monitor  X
Message (Enter to send message to 'ESP32 Dev Module')

Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 320995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089
Printing UART1 RX: esp32 check
Printing UART2 RX: 30.620995,-96.340089

```

```
30.620995,-96.340089
running
b'30.620995,-96.340089'
30.620995,-96.340089
running
b'320995,-96.340089'
320995,-96.340089
running
b'30.620995,-96.340089'
30.620995,-96.340089
running
b'30.620995,-96.340089'
30.620995,-96.340089
running
b'30.620995,-96.340089'
30.620995,-96.340089
running
b'30.620995,-96.340089'
30.620995,-96.340089
running
```



# Database & GUI

Brandon Stokes

Accomplishments since last update 21 hrs of effort	Ongoing progress/problems and plans until the next presentation
<ul style="list-style-type: none"><li>Completed integration with Kathleen and the communication between our systems</li><li>Tested app error handling for receiver off angle and location</li></ul>	<ul style="list-style-type: none"><li>Continue to work with transitioning from WiFi to LTE</li></ul>



# Database & GUI

Brandon Stokes

403 GUI

Start

Receiver 1 Location	Angle 1
30.80872,-96.45003	54.0
Receiver 2 Location	Angle 2
30.80414,-96.44432	87.0
Receiver 3 Location	Angle 3
30.81474,-96.45804	234.0

Expected Transmitter Location  
30.81631-96.44355

Calculated Transmitter Location

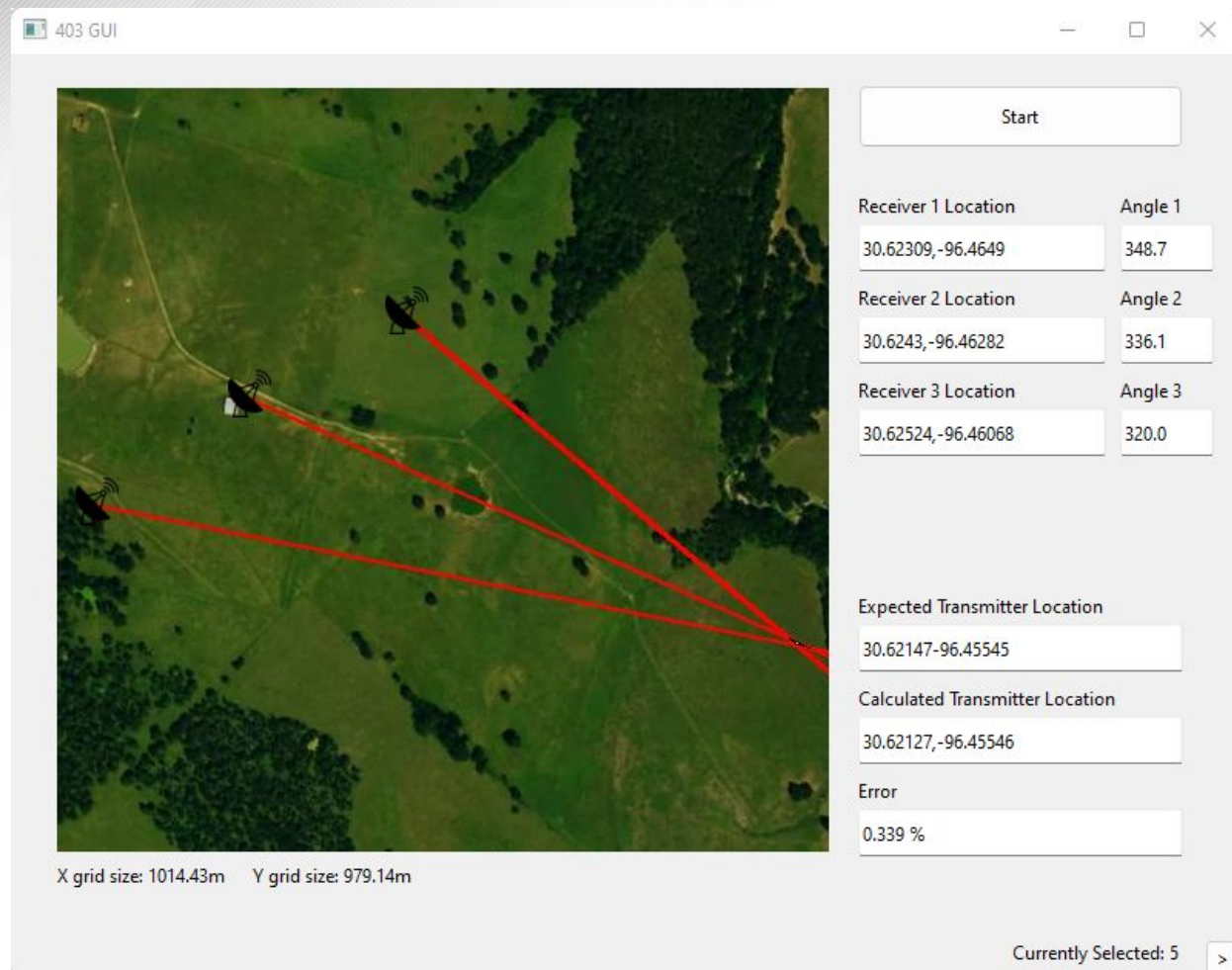
Error

X grid size: Y grid size:

Currently Selected: 3 >

Error in line 3

```
1 from machine import UART
2 import time
3 import pycom
4
5
6 pycom.heartbeat(False)
7
8 while True:
9     print("running")
10    uart = UART(1, baudrate=9600, pins=('P21','P20')) # voor esp
11    strMsg = ''
12    uart.write('esp32 check \n')
13    time.sleep(1)
14    if uart.any() > 0:
15
16        strMsg = uart.read()
17        print(strMsg)
18
19        #strMsg = str(strMsg, 'utf-8')
20        #print(strMsg)
21
22    else:
23        strMsg = "noData"
24        print(strMsg)
25    #time.sleep(2)
26
27
28
```



Time taken for dataset 5 is 2.518



# Execution Plan

[illegible]



# Validation Plan

Paragraph	Test Name	Success Criteria	Methodology	Status	Responsible Engineer
3.2.1.1	LTE Stability	The LTE does not drop more than 1 time per 5 minutes and shall reconnect within 20s.	System is put into default operational state (tracking transmitter) and left to run for 30 minutes while Pycom tracks LTE connection	UNTESTED	Brandon Stokes Kathleen Hutchinson
3.2.1.2	Antenna Characterization	Physical antenna has a gain of at least 7 in the direction of the antenna.	Physical antenna will be made and tested inside a characterization chamber with the help of Professor Nowka.	UNTESTED	Jack Parkinson
3.2.1.3	Motor accuracy	The motor can turn with speed and precision while the carrying the weight of the reciever PCB and antenna.	After connecting the system to the motor, it will spin with vaiying speeds and steps and be stopped to check accuracy and time.	UNTESTED	Jack Parkinson
3.2.1.4	System Connection	The time it takes to connect and transmit data between the GUI, transmitter, and receiver shall be 30s.	System runs start signal from application to receivers and back to application with data and timed using timer function into code.	UNTESTED	Brandon Stokes Kathleen Hutchinson
3.2.1.5	Operation Time	System operates continuously on battery power for 30 minutes.	System transmitter and receivers are put into default operational state and left to run for 30 minutes.	UNTESTED	Full Team
3.2.1.6	Detection Range	The detection range shall be an 150m radius from transmitter to a single receiver.	Receiver antenna will be place 150 meters from transmitter and be able to detect and step towards the signal transmitted.	UNTESTED	Josh Broyles Jack Parkinson Kathleen Hutchinson
3.2.2.1	Mass	The weight of the system shall be at max 27lbs.	Measure receiver unit with digital scale.	UNTESTED	Jack Parkinson
3.2.3.1.1	Input Voltage	The input voltage level for the ESP32 and MSP430 shall be 3.3V	Use E-Load to validate line and load regulation	TESTED	Josh Broyles
3.2.3.1.2	Input Voltage (Motor)	The input voltage level for the DRV811PWPR (motor driver) shall be 12V.	Use multimeter to validate input voltage levels	UNTESTED	Kathleen Hutchinson
N/A	Full System Demo	A user of system is able to accurately track the transmitter in an open space with a positional error of less than <10%.	System runs start signal to receivers which tracktransmitter's strongest signal with the motor, both stationary and moving, which sends data to GUI that outputs an accurate map with an error of <10% calculated through GPS points.	UNTESTED	Jack Parkinson
					Full Team



*Dwight Look College of*

**ENGINEERING**  
TEXAS A&M UNIVERSITY

**Thank you for your attention!**

**Feel free to ask us questions**