Long Distance Underwater and Aerial Drone Communication (LUDAC)

Ethan Wilson, Nikhil Saroya, Antony Brittain, Xinkai Xu
EE Capstone Group 22

***Abstract* — The Long Distance Underwater Drone to Aerial Communication project (LUDAC) integrates LoRa and WiFi for long range and near range communication between a submersible and an aerial drone. It realizes long distance communication with a max transmission distance of 1.3km using the LoRa protocol. Near-range communication is realized with the ESPNOW WiFi protocol, which has the ability to transmit large file sizes. The board is configured to switch between WiFi and LoRa based on signal quality to conserve power. LUDAC boards have been benchmarked for performance criteria including power consumption, packet integrity, bitrate, transmission distance, packet delay, LoRa/WiFi switching delay, etc. Overall, LUDAC boards have exceeded or met most of the specification requirements. The proposed functionalities, along with the extended goals of large file transmission and power supply configurability have been achieved by the design.**

I. Introduction

Our client, Dr. Jason Myatt, is working on a larger project with the final goal of dispatching an autonomous aerial drone from a submersible device that can complete aerial missions with minimal human interference. The motivation behind the Long Distance Underwater Drone to Aerial Communication (LUDAC) project is to realize kilometer range half-duplex communication between the aerial drone and the resurfaced submersible, so that they may coordinate tasks such as mission updates, aerial drone retrieval, and payload delivery.

II. High Level Functionalities

The primary goal of LUDAC transceivers, as requested by our client, is to maintain stable long-distance duplex communication between a surfaced submersible and the aerial drone dispatched for survey, rescue, or object retrieval tasks. The main types of data to be transmitted include: GPS coordinates of the submersible and aerial drone, relative distance, mission status (e.g., mission complete/in progress), and landing requests. When the submersible and aerial drone are in close proximity (i.e., in the order of meters), the transceivers should engage WiFi mode to support high bitrate communication. Image transmission via WiFi is also suggested as a stretch goal. When the aerial drone exceeds the range of WiFi communication, the Long-Range (LoRa) protocol will be engaged to realize long distance, low bitrate communication. The board is expected to be flexible with different types of power supplies (e.g., USB 5V/3.3V input, direct lithium ion battery input ranging from 10 to 14V), meanwhile maintain a low power consumption to avoid shortening mission time.

Fig. 2.1 shows a high-level block diagram that captures the essential hardware components and the types of connections integrated in the board. The MCU (ESP32 WROOM 32 UE) interfaces with the submersible/drone on-board computer (Raspberry Pi) directly through serial communication. GPS and LoRa modules are set up with the MCU via software serial and SPI, respectively. If a DC power supply such as a lithium ion battery is used, the input voltage is stepped down by the buck converter to a steady and low noise 3.3V level for the MCU. Otherwise, the MCU is powered directly by its USB bridge. Detailed hardware and firmware design to realize such setup in Fig. 2.1 will be discussed in the next sections.

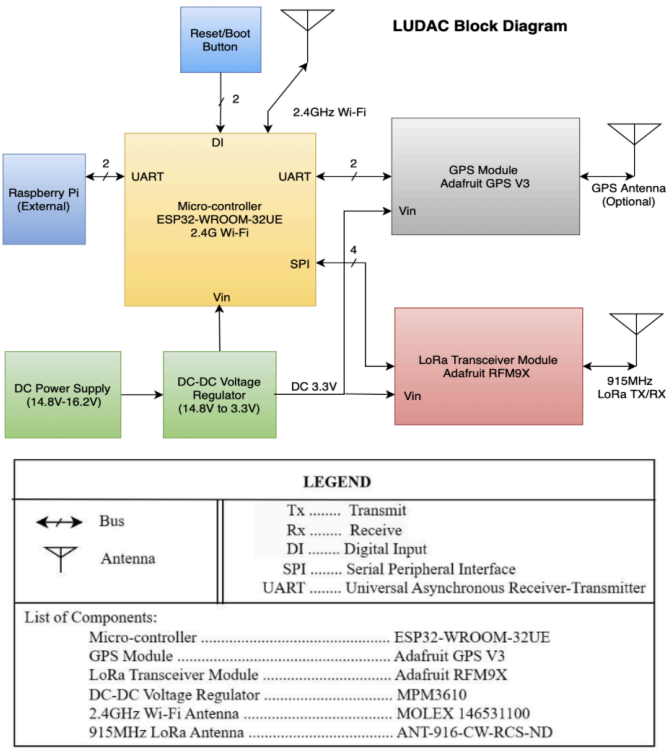

Fig. 2.1: Block diagram of the long distance underwater and aerial drone communication boards.

III. Hardware Design

A. Overall system design (Schematics)

The transceiver board consists of four main modules: the microcontroller unit (ESP32-WROOM-32UE), LoRa module (Adafruit RFM95W), GPS module (Adafruit Ultimate GPS v3) and the power supply/UART switching circuit. The ESP32-WROOM was chosen as the MCU for its abundance of GPIO pins, hardware serial availability, internal WiFi capabilities, external antenna compatibility, adequate flash memory size (4MB), and well-documented IoT development cases [1]. The Adafruit RFM95W LoRa breakout board is based on the SX1276 LoRa module with an SPI interface. This module has a typical minimum sensitivity of -127dBm and is compatible with external antenna installation [2]. The Adafruit GPS module is based

on the MTK3339 chip, which supports serial communication and can achieve up to -165dBm sensitivity [3]. The hardware is designed to support two types of power supply and UART configuration: (1) the transceiver board is directly powered by the ESP32 USB bridge, through which the UART connection with the on-board computer (e.g. Raspberry Pi) is also established; (2) the board is powered by an external DC power source (e.g. a lithium ion battery) whose voltage is regulated by a buck converter, and the UART connection is established via a set of hardware serial pins on the MCU. The high level MCU pin allocation for each module is shown in Fig. 3.1.

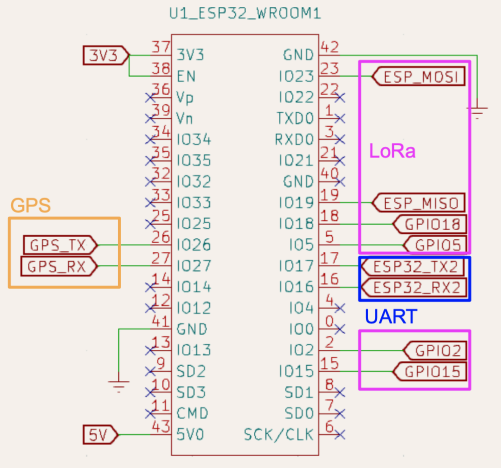

Fig. 3.1: ESP32 pin connection with GPS, LoRa, UART, and other power-related functions.

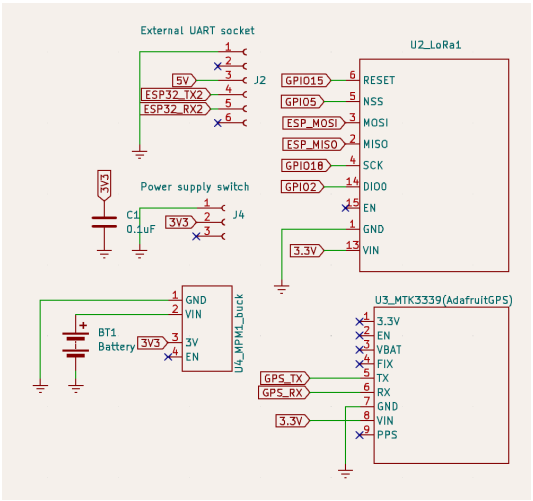

Fig. 3.2: LoRa module (U2_LoRa1), GPS module (U3_MTK3339), buck converter (U1_MPM1_buck), power supply switch (J4) and external UART socket (J2) schematics.

B. LoRa pin connection and external antenna

As shown in Fig. 3.2, the LoRa module is connected to ESP32 via SPI protocol. The module uses a 3.3V input voltage level. ESP_MOSI corresponds to pin GPIO23 and ESP_MISO GPIO19 on ESP32. The external LoRa antenna is a quarter-wavelength stick antenna with a nearly omni-directional radiation profile. It is connected to the LoRa module via a standard SMA connection.

C. GPS implementation

The GPS module is connected to ESP32 via software serial. This is because all hardware serial ports on ESP32 are occupied. GPIO26 and GPIO27 are allocated for software serial because they are not strapped pins and their positions on the MCU avoids trace clusters on the PCB. The GPS module has a default ceramic patch antenna. No external antenna is attached in the design of LUDAC.

D. Configurable power supply and UART connection

To realize the switching function as described in section A, a switching mechanism is implemented with header pins and a short jumper, figuratively shown as component J4 in Fig. 3.2. When the short jumper only connects pin 2 and 3 of J4, the battery, buck converter, and the external UART sockets are electrically disconnected from other parts of the board. In this case, MCU power supply and on-board computer UART communication are solely done through the USB bridge. This scenario is for user cases with available USB ports on the on-board computer. When the short jumper connects pin 1 and 2, it completes the circuit between the MCU and the external UART and power supply network. The USB bridge should never be connected in this case to avoid electrical damage to the MCU. In this case, the 3.3V pin on ESP32, connected to the buck converter output, acts as the voltage input to power the board. The external UART socket is designed so that it is compatible with typical commercial UART jumpers, with the pin layout , from left to right, being: 1-GND, 2-void or CTS, 3-VCC (5V), 4-TX, 5-RX, and 6-void or RTS.The buck converter has an input range of 4 to 21 volts and an output of 3.3V [4]. The typical lithium ion battery used by our client has a voltage range of 10 to 14 volts, which is fully compatible with our board.

E. PCB design (PCB layout)

The designed PCB layout is shown in Fig. 3.3, corresponding to the schematics shown in Fig. 3.1 and 3.2. This is a double-sided copper board with a 1.67mm thick FR-4 substrate and planar dimensions of 75mm by 75mm. The height of the board with all components installed is 1.8cm. Vias are incorporated to avoid trace clusters. The professional fabricated PCB with all parts mounted on is shown in Fig. 3.4. The modules are oriented so that the USB bridge, DC power leads, power switch, UART sockets, and the antenna SMA connector can be easily accessed by the user.

In this setup, the short jumper connects pin 2 and 3 of the switching header set, which sets the board in USB bridge mode. To configure the board to adapt external power and UART, the user is required to manually move the short jumper to pin 1 and 2 as described in part D.

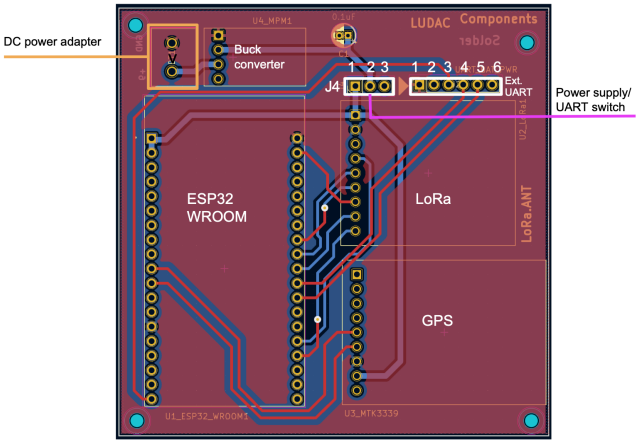

Fig. 3.3: Annotated transceiver PCB layout.

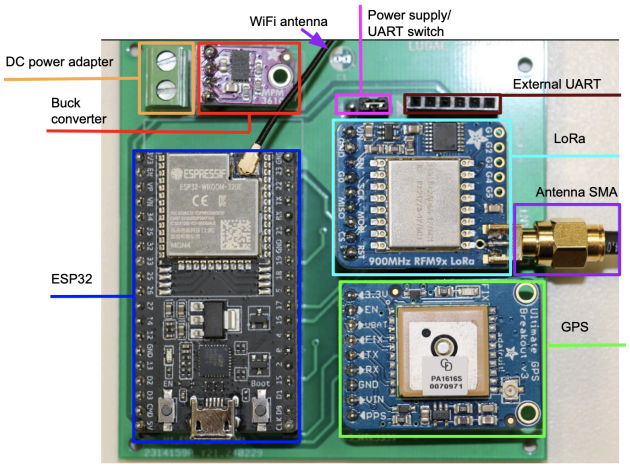

Fig. 3.4: The fabricated PCB with jumper sockets, header pins, antennas, and all breakout parts mounted. The short jumper is on pin 2 and 3 of the switch header.

IV. Firmware and Software Design

The high-level main-loop routine flow is shown in Fig. 4.1. The subsections of the main routine and functionalities are discussed subsequently.

A. Development environment

The firmware and software development is done with PlatformIO in the Espressif environment, an ESP32 proprietary development environment based on C/C++ framework. The Arduino framework is partially corporated with consent from our client, team members, and the ECE 491 instructors, as all LoRa/ESP-NOW libraries have Arduino dependencies. Also, by avoiding the poorly documented non-Arduino ESP-NOW libraries, this ensures smooth project handover to the client's submersible R&D team.

B. Interface with the On-Board Processor

The communication between the ESP32 and Raspberry Pi is achieved by continuously polling the UART interface to check for incoming instructions. This polling-based approach ensures consistent handling of data and events at regular intervals, making it suitable for applications where real-time responsiveness is not critical or where the system resources are limited. When an instruction is received to send data from the master (Raspberry Pi) to the slave (ESP32), the data is read and stored in a local buffer. If the received data exceeds the static buffer size, the code dynamically resizes the buffer to accommodate the incoming data. This prevents buffer overflow errors and ensures the complete reception of the data. Conversely, when an instruction is received to send data from the slave (ESP32) to the master (Raspberry Pi), the available data is transmitted to the master. The polling mechanism ensures that the system regularly checks for and processes these data transmission requests.

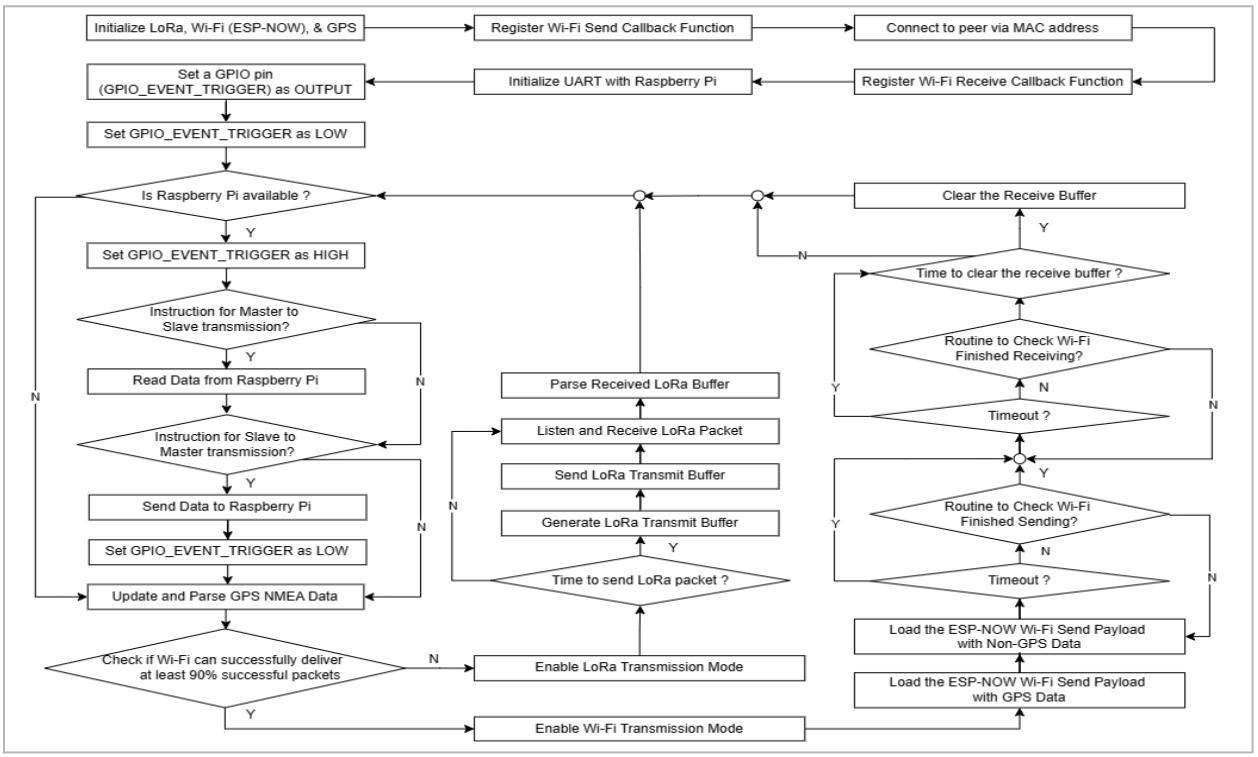

Fig. 4.1. The high-level software flowchart of the LUDAC transceiving system

The use of polling, rather than an interrupt-driven approach, is a deliberate design choice. Polling provides a more predictable and consistent data handling mechanism, which is important in applications where real-time responsiveness is not the primary concern. Interrupt-driven approaches can be more complex to implement and may introduce unpredictable delays or race conditions, especially in resource-constrained systems like the ESP32.

C. *LoRa half-duplexing*

LoRa half-duplexing is realized through random scheduling with an average transmission interval of 2 seconds, as shown in Fig. 4.2. This method does not involve acknowledgement of successful packet delivery/reception, but is confirmed to be virtually lossless from the benchmark results (to be discussed later) within the operation range. The firmware development is based on the Sandeep-Mistry-LoRa library with LUDAC application specific modifications.

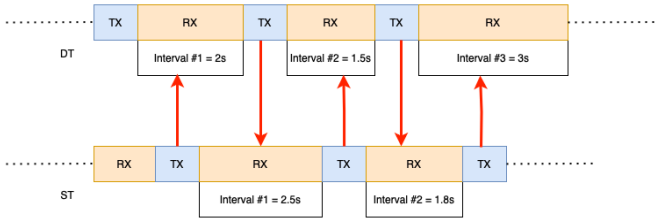

Fig. 4.2: LoRa random scheduling with base interval 2 second and a randomness of ± 0 to 1 second. DT: drone transceiver; ST: submersible transceiver.

D. *ESP-NOW*

ESP-NOW is a wireless communication protocol developed by Espressif that enables multiple devices equipped with ESP-series microcontrollers (like the ESP32 and ESP8266) to communicate with each other directly, without the need for a WiFi network [5]. This protocol operates on the 2.4 GHz WiFi band, offering a low-power, low-latency solution for various IoT applications. ESP-NOW is especially valuable in scenarios where devices need to exchange small amounts of data quickly and efficiently, such as sensor networks, home automation, and wearable electronics. Its ability to facilitate secure, peer-to-peer communication makes it an ideal for this project, which requires direct connection and coordination between multiple microcontrollers, minimal power consumption, and a simple network topology.

E. *LoRa/ESP-NOW switching*

Switching between LoRa and ESP-NOW protocols minimizes power consumption by ensuring that only one type of protocol is transmitting or receiving at a time. Switching takes advantage of the ESP32 packet delivery acknowledgement. This mechanism allows the sender to know whether the data was received successfully by the peer device. If the module callback function receives at least 90% success acknowledgements indicating successful delivery of packets, the system enables WiFi transmission mode, otherwise it enables LoRa transmission mode. This approach allows the system to dynamically adapt to changing environmental conditions and network connectivity, ensuring optimal power consumption while maintaining reliable data transmission. The switching function was initially designed to use an absolute distance calculation, but this resulted in measurement errors of up to 30% at the 1km mark (see Fig. 5.2).

F. *Transceiving Data Segmentation*

Fig. 4.3: Packet sending and receiving logistics.

Both ESP-NOW and LoRa communication protocols are limited by a ~250 byte payload size, so longer data is handled algorithmically via segmentation. In standard operation, payloads containing mission and GPS data are sent and received continuously, with space allocated in each payload for a packet of segmented data. This space is unused when the device is not transmitting long-form data. Upon receiving an instruction from the onboard computer to send long-form data, the firmware sets the 'sending' flag, which initiates the segmentation routine. The transmitting device first indicates to the intended recipient when it is about to begin segmented transmission by sending a payload indicating the length of the impending message and a flag called 'first_packet' set to true. This causes the other board to set its 'receiving' flag. The transmitter then begins loading each outgoing payload with a new packet of the data stored in 'sendBuffer'. When a packet is successfully received by the other board, the firmware increments the 'sendIndex' pointer and transmits the next packet of data from the buffer. It repeats this process until the message is

fully received by the other board, at which point the 'sending' flag is set to false. The final packet is indicated to the other board with a 'final_packet' flag. Upon receiving a packet from the other device, a callback function interrupts the main routine to load it into 'recvBuffer'. When the buffer is full, the data is sent to the computer and cleared. At present this process is only implemented for ESP-NOW, but it could be extended to LoRa in future versions. This routine is depicted in Fig. 4.3.

V. Testing Methods and Results

A. *Electrical testing*

The purpose of electrical testing is to check voltage stability, current, and power draw under different conditions. During testing, the input DC voltage range was tuned from 6V to 16V with a 2V interval to imitate the lithium-ion battery used by the drone and the submersible, and the output voltage was measured. The stepped down voltage shows slight increase with input voltage, with an average of 3.281V ± 0.1% as shown in Fig. 5.1.

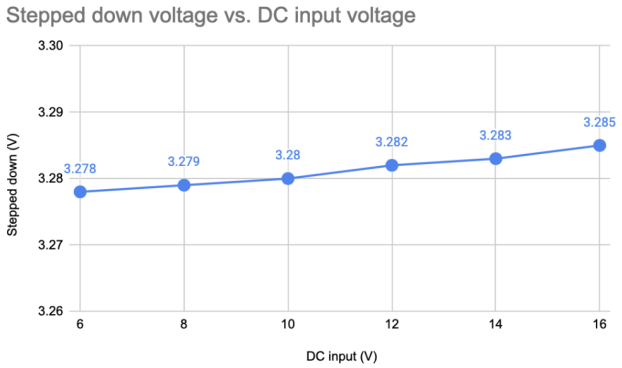

Fig. 5.1: The stepped down battery voltage with varying DC voltage input.

Table 5.1 presents the measured power consumption of the our board in three different scenarios when it is connected to a 10V DC power supply: : (1) ESP32 idling, no LoRa/GPS/WiFi functions (2) GPS and LoRa half-duplexing on, WiFi off, and (3) GPS and WiFi half-duplexing on, LoRa off. RX and TX current draws were measured separately.

Input to Buck Converter "DC Measurements"			
Scenario	Volts	Amps (mA)	Power Calculation [W]
Idling "Nothing"	10	45	0.45
LoRa "GPS ON" TX	10	56	0.56
LoRa "GPS ON" RX	10	45	0.45
WIFI "GPS ON" TX	10	65.3	0.653
WIFI "GPS ON" RX	10	63.6	0.636

Table 5.1: Current and power consumption under different scenarios

Table 5.2 presents estimated energy use and time averaged power draw for a mission time of 30 minutes. The total energy draw is 0.23 Wh, while the average power draw is 0.46 W.

		Time Running:	30 min
Scenario	Time percentage	Total Time (s)	Watt-hours
Idling "Nothing"	60%	1080	0.135
LoRa "GPS ON" TX	2%	36	0.0056
LoRa "GPS ON" RX	32.50%	585	0.073125
WIFI "GPS ON" TX	0.50%	9	0.0016325
WIFI "GPS ON" RX	5%	90	0.0159
		Total Watt-hours	**0.2312575**

Table 5.2: The time-averaged power consumption of each operation mode, assuming a 30 minute long mission time.

Known issue: When the board is battery powered, the 5V port of the ESP32 MCU only outputs 2.7V. This means that a UART to USB adapter cannot be used as it requires a 5V input voltage. However, this does not affect the direct UART communication where only TX and RX connections are required. This issue will be relayed to the client's team.

B. *GPS distances accuracy*

The relative distance between the LUDAC transceivers is calculated from the GPS readings of both boards using the Haversine formula [6]:

$$d = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\Delta Lat}{2}\right) + \cos(Lat_1)\cdot\cos(Lat_2)\cdot\sin^2\left(\frac{\Delta Long}{2}\right)}\right)$$

As shown in Fig. 5.2 and Fig. 5.6, the calculated relative distance shows little discrepancy with the expected value up to 500m, but grows up to 30% after that. Testing data shows that the Adafruit GPS readings may be off by as much as 20'. The module does not support hardware calibration, so further consideration must be given to replacing the GPS module.

C. *LoRa half-duplexing packet integrity and distance*

The LoRa half-duplexing communication testing was done by collecting the RSSI and signal-to-noise-ratio (SNR) at incremental distances. The testing LoRa packets were numbered. During testing, any lost packets (i.e., missing packet numbers) were documented. The packet loss rate is less than 5% over 1km of LoRa transceiving distance.

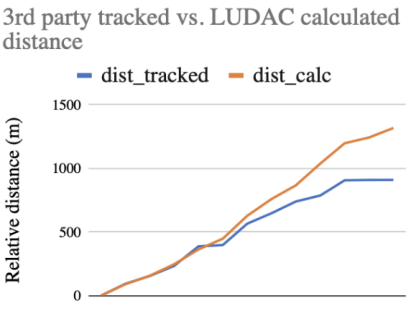

Fig. 5.2: Discrepancy between the LUDAC calculated relative distances and the 3rd party application tracked distances.

The effect of LoRa spreading factor, bandwidth, and TX power have also been investigated in controlled

experiments. The SNR and RSSI values were recorded at each spreading factor set point (6 to 12) with distance, TX power and bandwidth controlled. The results are shown in Fig. 5.3. The same data was collected with varying bandwidth (7.8kHz to 500kHz) with distance, TX power and spreading factor controlled. The results are shown in Fig. 5.4. Two TX power settings were also benchmarked. The RSSI and SNR values are -123 dB and -9.5 dB with TX power set to 17, and -120.4 dB and -8.5 dB with TX power set to 19. The LoRa transmitting time was measured at each spreading factor setting with a set 5 character long LoRa packet. The results are shown in Fig. 5.5. The time cost increases exponentially with increasing spreading factors. To optimize LoRa performance, the module should be configured (spreading factor, bandwidth, TX power) to satisfy the following criteria: 1) highest-possible RSSI, 2) lowest-possible SNR, 3) fastest-possible transmitting speed, 4) fewest-possible packet losses. However, it is difficult to satisfy both 1) and 2) as shown by the RSSI and SNR trends in Fig. 5.3 and 5.4. Therefore, the settings need to be configured based on application. For LUDAC, spreading factor is set to 6, bandwidth 125 kHz, and TX power 17 with the prioritized goal to minimize SNR and packet loss. Future users of this board are advised to configure these settings per application.

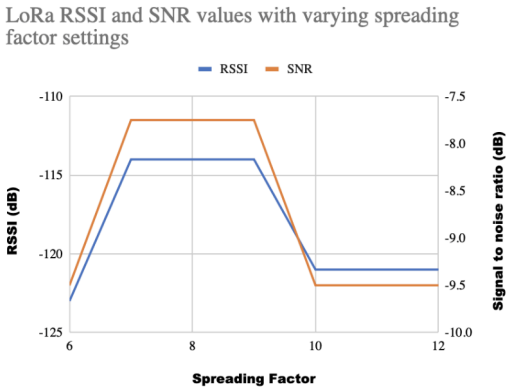

Fig. 5.3: LoRa RSSI and SNR values obtained with varying spreading factors from 6 to 12, with BW, distance (40 m), and TX power controlled.

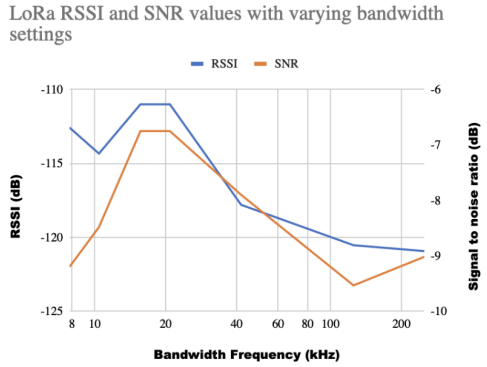

Fig. 5.4: LoRa RSSI and SNR values with varying bandwidths from 7.5 to 250 kHz, with spreading factor, distance (40 m), and TX power controlled.

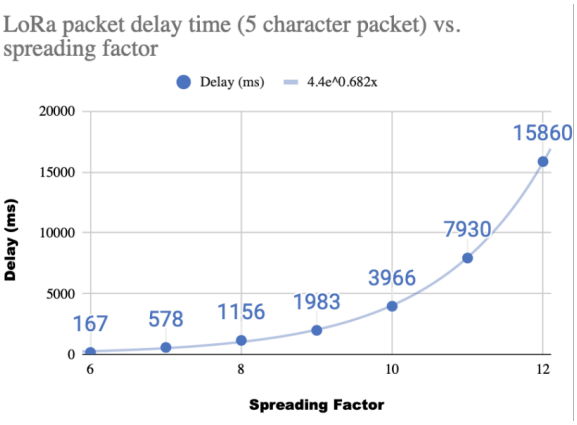

Fig. 5.5: LoRa packet delay time with a 5 character buffer with varying spreading factor settings. BW, distance (40 m), and TX power controlled.

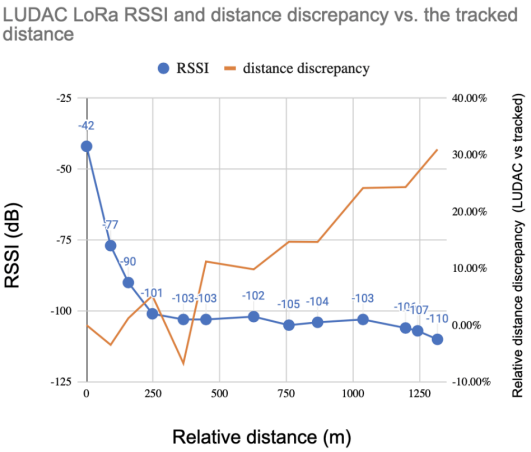

Fig. 5.6: LUDAC LoRa RSSI and distance tracking error over transceiving distance.

The maximum LoRa transceiving distance is 1300m over an open field with only vegetation in the line of sight. This distance is reduced to 340m when operating with buildings in the line of sight and RF interference. As shown in Fig. 5.6, the LoRa RSSI decreases sharply within 250m of distance, and reaches a plateau afterwards. At a distance of 1km, LUDAC boards were intentionally disconnected from power to test LUDAC connection recovery ability. Both LUDAC boards were able to reconnect via LoRa and resumed synchronous half-duplexing communication within 10 seconds. Same experiment has been done for ESPNOW connectivity, and it is proven robust to recover half-duplexing upon incidents such as loss of power or board reboot.

D. ESP-NOW half-duplexing distance and signal strength

The maximum distance for ESP-NOW half-duplexing is determined by the received RSSI values. Typically, an RSSI greater than -80dbm is reliable for WiFi or cellular communication. Two LUDAC boards with ESP-NOW initiated and operating were tested with increasing distance. ESPNOW does not report RSSI values directly. The testing script implemented WiFi in the promiscuous mode to obtain

RSSI readings via a callback function. Repeated tests have shown that more than 10% of the packets would fail at a distance greater than 20m. The reported RSSI values upon successful delivery of packets range between -40 to -85 dB.

E. LoRa/ESPNOW switching

The absolute distance at which LoRa and WiFi switching occurs was benchmarked. During the test, the protocol switching distance was measured between two LUDAC boards. The protocol initially operated in WiFi mode, and at 230m, both LUDAC boards switched to LoRa. The test environment included buildings and RF interference, so the switching distance is expected to be higher in a cleaner environment.

VI. Device Performance Evaluation and Future Steps

Table 6.1 summarizes the power and hardware specifications proposed in the design phase of this project, and what has been achieved via the LUDAC boards. The achieved power parameters are mostly lower than the proposed values, or comparable. The dimension of the PCB with all modules mounted on has a smaller 2D footprint but taller profile than the specification, but the overall housing is smaller than what's been proposed.

Power and hardware specs vs Measured	**Proposed Specs**	**Measured**
Min TX Power (W)	0.53	0.56
Max TX Power (W)	1.81	0.653
Average Power (W)	0.49	0.46
Min current (mA)	159.1	45
Max current (mA)	548	65.3
PCB dimension (cm)	9x5x1	7.5x7.5x1.8
Housing dimension (cm)	13x3.5x8	8.4x8.4x2.5

Table 6.1: Power and hardware specifications and measured values. [7]

Table 6.2 compares the transceiving parameters achieved by LUDAC boards and what's been proposed in the revised design report [7]. LUDAC has exceeded the proposed LoRa payload length and met the WiFi payload length, but is just shy from meeting the other requirements including bitrates, maximum LoRa transceiving distance, Max LoRa and WiFi sensitivity. The discrepancy between the distance and sensitivity is due to the non-ideal conditions due to RF interference and obstructions in the line-of-sight. The LUDAC LoRa bitrate is actually intentionally lowered because the proposed bitrate can only be achieved at a scattering factor of 12. As shown in Fig. 5.4, SF 12 implies a 15 second delay, which is impractical for the LUDAC transmission frequency. Improvements on LoRa performance will be discussed in the final section.

Transceiving specs vs achieved	**Proposed specs**	**Achieved**
LoRa Payload length (bytes)	64	250
WiFi payload length (bytes)	250	250
Max LoRa bitrate (kbps)	37.5	1
Max WiFi bitrate (kbps)	1000	240
Max LoRa distance (km)	2.5	1.3
Max LoRa sensitivity (dB)	-136	-125
Max WiFi sensitivity (dB)	-92	-85

Table 6.2: Transceiving specifications and achieved values. [7]

Future steps

The power conversion efficiency of the buck converter at low current draw (<100mA) is not ideal, and varies with the input voltage. As shown in Fig. 6.1, the practical efficiency is estimated to be 75% given the measured average current draw and the range of lithium ion battery voltage output. This buck converter also has a less than 3.3V output voltage, which may detriment the board functions. It is recommended that a better buck converter should be used in a future version of this project, to maintain a standard 3.3V output with little to no loss.

The external UART port design could be revisited as well, as in the current version, the 5V port only reaches 2.7V when the board is powered by an external battery. To resolve this problem, a buck converter that steps input voltage down to 5V could be used in place of the 3.3V converter. In this case, the 5V output of the buck converter will be connected to the ESP32 5V (input) port, and the external UART switch. The rest of the power supply network will remain the same.

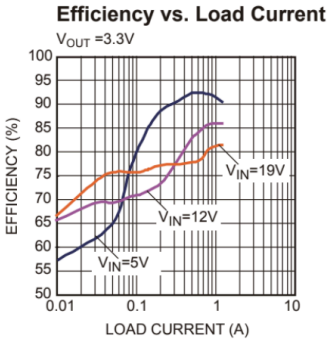

Fig. 6.1: Buck converter efficiency vs. the load current, at three voltage levels. [4]

The GPS accuracy is very much weather dependent, and measurements are more accurate in warm and sunny conditions than in cold and overcast conditions. Most outdoor testing was done in sub-zero temperatures due to weather constraints. Considering that the LUDAC board will be used with autonomous water vehicles, future testing should be done with controlled ambient conditions to mimic

the actual operating conditions, such as high relative humidity and above zero temperature.

The Adafruit Ultimate GPS module should be replaced with a better option that supports calibration. LUDAC GPS readings can have up to 20' offset from the actual coordinates, resulting in inconsistency of the relative distance calculation. The maximum LoRa range can be extended by selecting a more robust LoRa module, superior antennas, and better SMA connectors. It is also worth considering a design that uses standalone chips (i.e. without breakout boards) and embedded RF feed circuitry to avoid parasitic impedances caused by the sockets, solder points, and jumper pins.

LUDAC should also be benchmarked for its mechanical properties which would impact the drone and submersible balancing and maneuvering. For example, analysis on the center of gravity should be done to decide how the board should be mounted relative to other parts of the drone and the submersible. The extruded WiFi and LoRa antennas are particularly likely to be points for mechanical failure. Antenna supports or alternate antenna structures could be used to reinforce the mechanical stability.

```
Input  : State of ESP32 and Raspberry Pi
Output: Signaling between ESP32 and Raspberry Pi
Initialization;
while True do
    Perform other tasks on ESP32;
    if ESP32 becomes ready then
        Signal readiness to Raspberry Pi;
    end
    if ESP32 becomes busy then
        Signal busy state to Raspberry Pi;
    end
end
```

Fig. 6.2: Asynchronous communication routine between LUDAC and Raspberry Pi.

As shown in Fig. 6.2, the asynchronous communication between the ESP32 and Raspberry Pi involves initializing GPIO pins on both devices to signal readiness, with the ESP32 also defining a volatile boolean variable "busy" to track its availability. However, this hardware function is not implemented in the current version of LUDAC board. Users are advised to connect a jumper cable to one of the GPIO pins (non-strapping) of ESP32, such as GPIO25, to realize this pin-assisted asynchronous communication. On the ESP32 side, the initialization process sets up the GPIO pins and other necessary configurations, after which the main loop performs other tasks or computations and signals readiness by setting the GPIO pin HIGH, and continues performing tasks. The ESP32 also defines a function "notifyRaspberryPi" to indicate its busy state by setting the GPIO pin LOW. On the Raspberry Pi side, the GPIO pin is configured to receive the readiness signal from the ESP32, and a function "wait_for_ready" is implemented to wait for the ESP32 to become ready or timeout. The Raspberry Pi continuously monitors the GPIO pin, sending requests or performing actions when the pin is HIGH (indicating the ESP32 is ready), and waiting for the pin to become HIGH again when the pin is LOW (indicating the ESP32 is busy), with a timeout mechanism to handle cases where the ESP32 does not become ready within a specified time.

VII. Conclusions and Future Directions

The designed and fabricated LUDAC transceiver boards successfully accomplish half-duplexing communication via switchable WiFi and LoRa protocols at a range of 1km with low power consumption and a small footprint, allowing facile integration with the drone and submersible housings. The board also realizes the stretch goals of configurable power supply/UART connections and large file transmission (data segmentation). The total cost of two LUDAC transceivers is $317 CAD, equivalent to 80% of the proposed budget. Testing shows a time-averaged power consumption is 0.46W, and a maximum LoRa transceiving distance of at least 1.3km, which exceeds the expected 1km range. The GPS-based relative distance calculation is accurate up to 500m, but sub-par above this limit. For the client's interest, the dependency of LoRa RSSI and SNR on the scattering factor, bandwidth, and TX power settings have been benchmarked and presented in the report. The LoRa transceiving bitrate is consistently 1kbps, and the WiFi (ESP-NOW) bitrate could be tuned up to a maximum of 240kbps. However, there are still areas of improvement that the team should consider for future development.

References

[1] Espressif Systems, "ESP32-WROOM-32E / ESP32-WROOM-32UE Datasheet", Version 1.6, 2023. Accessed: Apr 9, 2024

[2] Adafruit, "Adafruit RFM69HCW and RFM9X LoRa Packet Radio Breakouts", 2023-10-24. Accessed: Apr 9, 2024

[3] Adafruit, "Adafruit Ultimate GPS", 2023-08-29. Accessed: Apr 9, 2024

[4] Monolithic Power. "MPM3610" Rev. 1.01 (1/7/2015). Accessed: Apr 9, 2024 [Online] Available:https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MPM3610GQV-Z/document_id/2090/

[5] "ESPNOW," Espressif Systems, https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html. Accessed Apr. 9, 2024

[6] "Haversine formula to find distance between two points on a sphere," GeeksforGeeks, https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/ (accessed Apr. 9, 2024).

[7] A. Brittain, N. Saroya, E. Wilson, X. Xu. "Revised Design Report". ECE 490. Accessed: Apr 9, 2024

Acknowledgements

The authors of this report would like to acknowledge the advice, contributions, and feedback given to us by the ECE 490/491 instructional team, including Loren Wyard-Scott, Zoltan Kenwell, Jinyuan Wei, and Mohammadreza Khoshhal. The authors would also like to acknowledge the support given to us by Dr. Abdulhakem Elezzabi, whose ideas and feedback were valuable throughout the design process. We would like to acknowledge Yeh-In Kang, Hariharan Krishnan, and Nico Schiavone, who worked collaboratively with us as the drone development team, as well as Aiden Hussey, who provided valuable insight into the operation of the existing project. Finally we would like to acknowledge the support given to us by our client, Dr. Jason Myatt, whose guidance was greatly appreciated by the team.