Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Fall-Spring 2021-22

Positioning Partners
Positron

Daniel Marks
Deborah Jahaj
Kamal Karkidholi
Nathan Fusselman
Oscar Mariscal

Revision History

Revision	Date	Author(s)	Description
1	2.5.2022	DM,DJ,KK,NF,OM	document creation
1	5.10.2022	DM,DJ,KK,NF,OM	

Contents

List of Figures

List of Tables

1 Introduction

Positioning Partners is creating an autonomous rover that is capable of completing some simple tasks. One of these tasks would include spraying dots or lines of paint on the ground in a grid. The second task would include driving a stick of rebar into the ground at any of the marked locations. The rover would be able to maneuver autonomously to GPS coordinates, or a person could manually pilot the rover using a remote control allowing for more freedom in driving the rover. This would allow the rover to paint the dots and lines on the ground, and also allow it to drive rebar into the ground at any specified locations. Whilst driving, the rover would be able to scan its environment and surroundings using a lidar scanner and camera that is attached to it, allowing for safer traversals from point to point.

2 System Overview

To improve the deign and allow the team to easily modify parts of the design without the need to redesign other non dependent modules. This approach to design also improves the ability of team members to work on separate modules and make improvements to packages/layers without having to include others and contain bugs and crashes to the layer that contains it and those that directly depend on the data it provides. Some of these layers' subsystems will be software based and others will be hardware as well as most will have a portion of them that is both software and hardware. A Graphical representation of these layers has been provided with a in depth explanation of its responsibilities, dependencies, and capabilities.

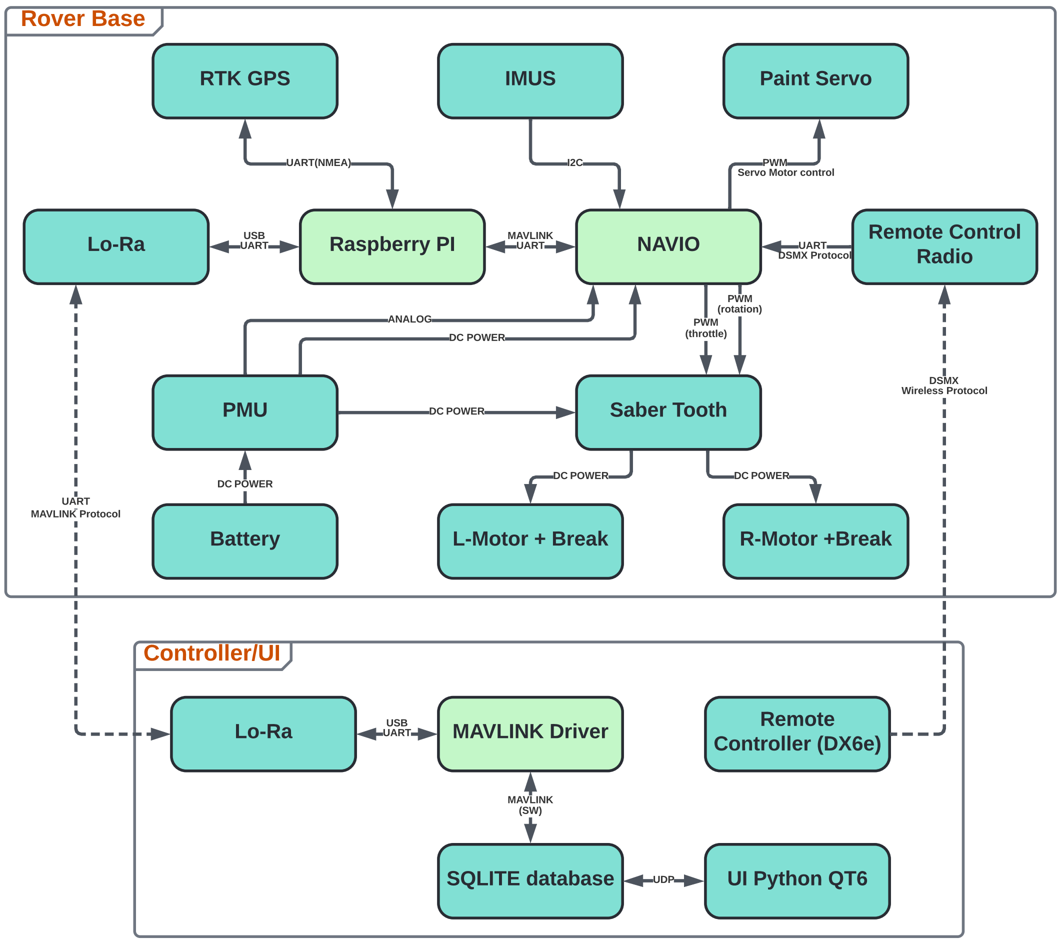

Figure 1: System architecture

3 Rover Base Layer Subsystem

The Rover Base is the central point of communication for all components and houses the primary computer that is responsible for all decisions that the rover makes. This is done with the use of a Raspberry Pi and then the device interfaces will all sensors, inputs, and control outputs to perform the desired tasks.

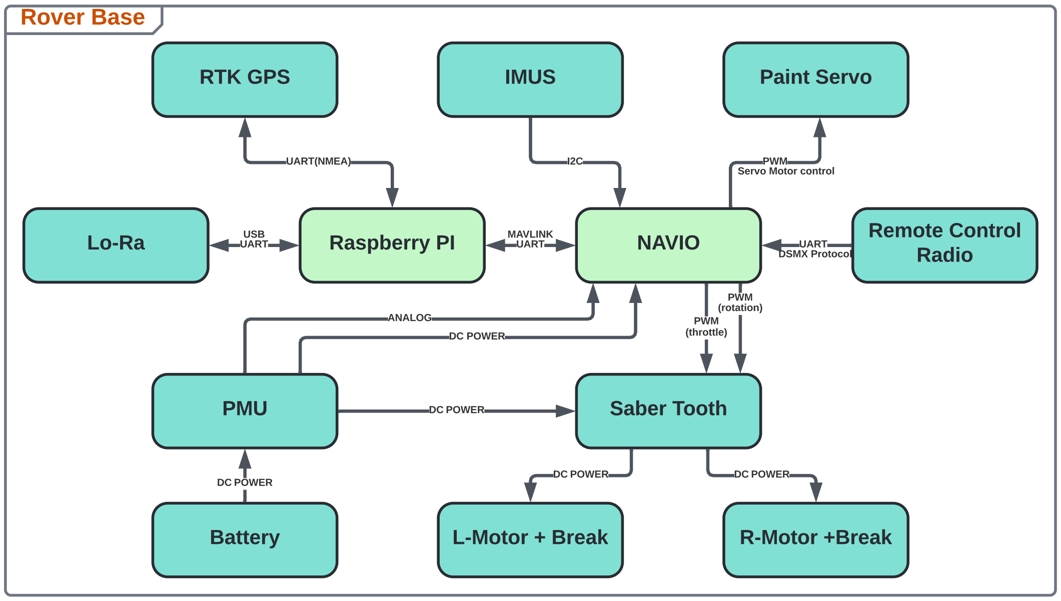

Figure 2: Rover Base Layer Diagram

3.1 Layer Hardware

A description of any involved hardware components for the layer. For example, if each subsystem is a software process running on an embedded computer, discuss the specifics of that device here. Do not list a hardware component that only exists at the subsystem level (include it in the following sections).

3.2 Layer Operating System

A description of any operating systems required by the layer.

3.3 Layer Software Dependencies

A description of any software dependencies (libraries, frameworks, etc) required by the layer.

3.4 NAVIO

The NAVIO serves as a rover ArduPilot and basic control system performing all operations related to controlling the motors and receiving instructions from the remote control subsystems. This subsystem also provides real time kinematic data from on board sensors, such as accelerometer, magnetometer and on board gps.

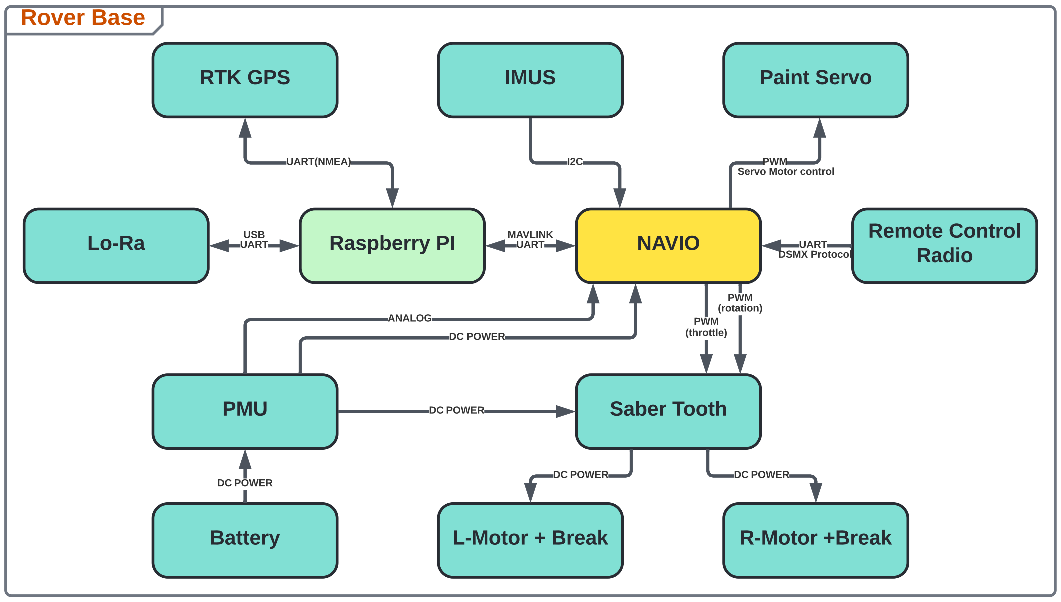

Figure 3: Example subsystem description diagram

3.4.1 Subsystem Hardware

The subsystem is comprised of a Navio2 which connects to the Raspberry Pi using the 40 pin GPIO header.

3.4.2 Subsystem Operating System

The OS running in this board is ArduPilot.

3.4.3 Subsystem Programming Languages

The Navio accepts instructions using the MAVLink2 protocol.

3.5 Raspberry Pi

The Raspberry Pi is a micro-controller running VLinux operating system which interfaces with the NAVIO and other high level sensor such us the RTK GPS. The overall function for this subsystem is to supply commands to the NAVIO over a MAVLINK connection in order to provide a cm level accurate location and to provide driving instructions to the NAVIO.

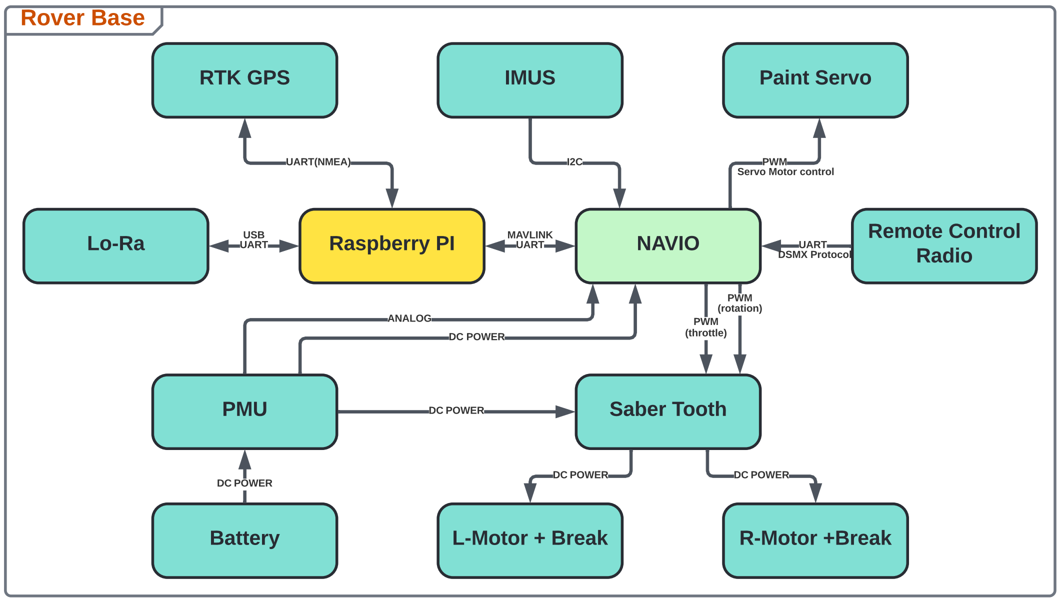

Figure 4: Raspberry Pi Subsystem

3.5.1 Subsystem Hardware

The subsystem is comprised of a Raspberry Pi 4 model B+ with 8GB with on-board RAM.

3.5.2 Subsystem Operating System

The OS running on the micro controller is Raspberry Pi OS commonly referred as Raspbian.

3.5.3 Subsystem Software Dependencies

The software running on this subsystem is written in Python3.10.2 using packages provided by Sparkfun Electronic in order to interface with the GPS, as well as MavGen to interface with the NAVIO.

3.5.4 Subsystem Programming Languages

All software will be written in Python3.10.2 to improve portability.

3.6 LoRa

The LoRa module on the rover base acts as the transmitter/receiver used by the controller to communicate with the rover. This module is capable of fast long range communication with the controller. The LoRa module is responsible for sending all telemetry data to the controller as well as receiving all commands that the controller sends. The LoRa module communicates with the Raspberry Pi via a UART connection.

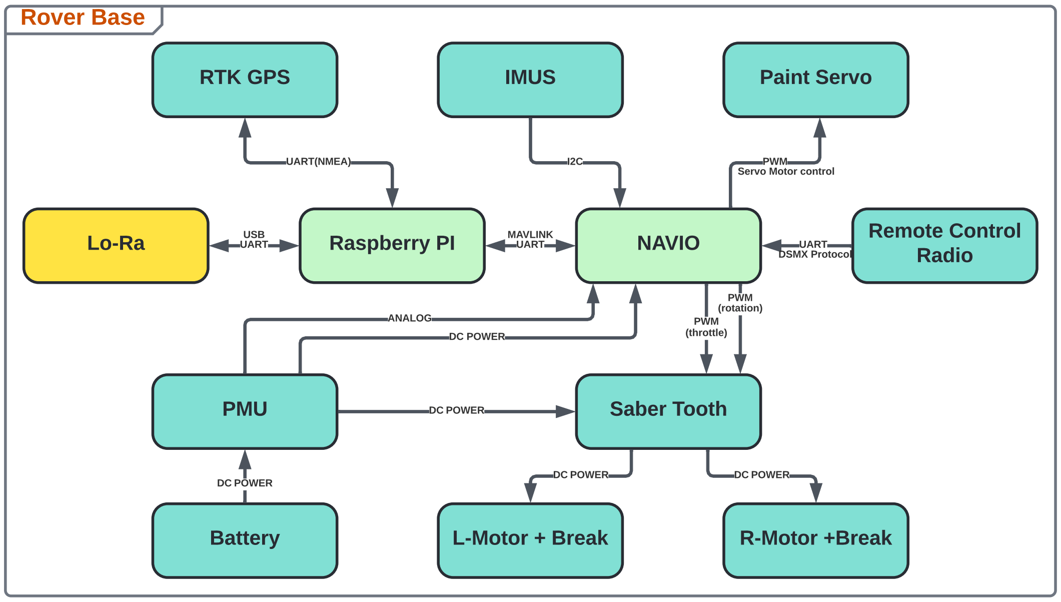

Figure 5: LoRa Subsystem

3.6.1 LoRa Hardware

The subsystem hardware consists of one half of a 433Mhz Holybro Telemetry Radio System with antennas.

3.6.2 LoRa Operating System

N/A

3.6.3 LoRa Software Dependencies

MAVLink protocol framing.

3.6.4 LoRa Programming Languages

The MAVLink communication protocol will be handled with Python using Pymavlink a low level and general purpose MAVLink message processing library.

3.6.5 LoRa Data Structures

The mavutil module will be used to set up and manage the communication channel. This module provides simple mechanisms for setting up links, sending and receiving messages, and querying some basic autopilot properties (e.g. flight mode). It provides access to the dialect module used for encoding/decoding/signing via an attribute (mav). Messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects build on top of this definition).

3.6.6 LoRa Data Processing

N/A

3.7 GPS

The GPS subsystem provides centimeter level accurate positioning data to the rover through the use of a ZED-F9R taking advantage of RTK GPS technology and providing this data to the Raspberry Pi, which is then passed to the Navio.

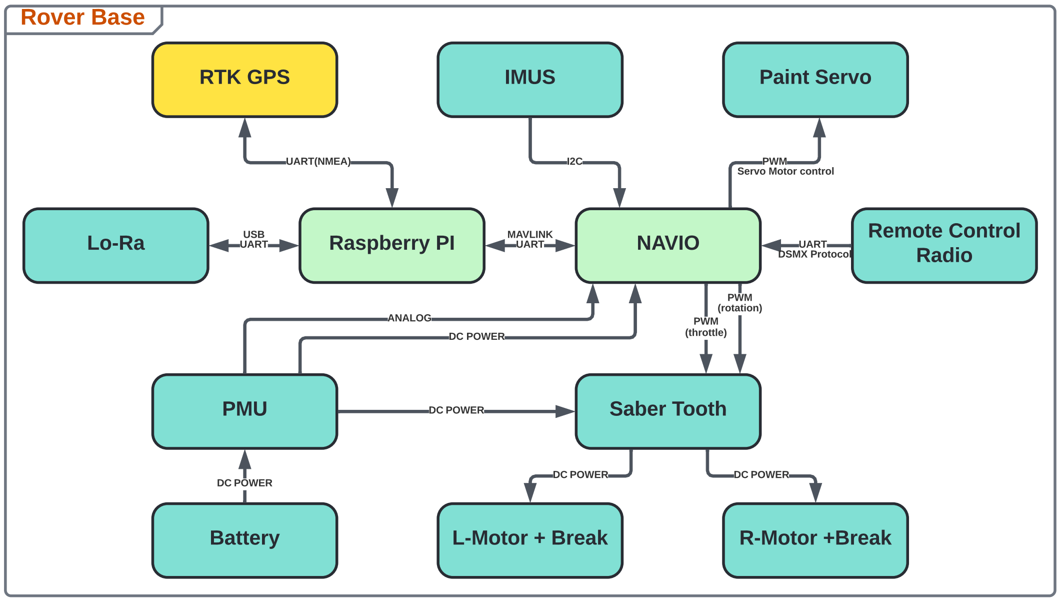

Figure 6: GPS Subsystem

3.7.1 Subsystem Hardware

The subsystem is comprised of a ZED-F9R board that interfaces with the Raspberry Pi over a UART Communication Link. Additionally, this subsystem comes with a GPS antenna provided by U-blox.

3.7.2 Subsystem Software Dependencies

This subsystem can interface with U-Center2 to configure the GPS module.

3.8 IMU

This subsystem provides redundant kinematic data to the rover through the use of two 9-degree of freedom IMU chips that are in the Navio board. These chips provide data to the Navio such as acceleration, magnetic heading and rotation on three axis.

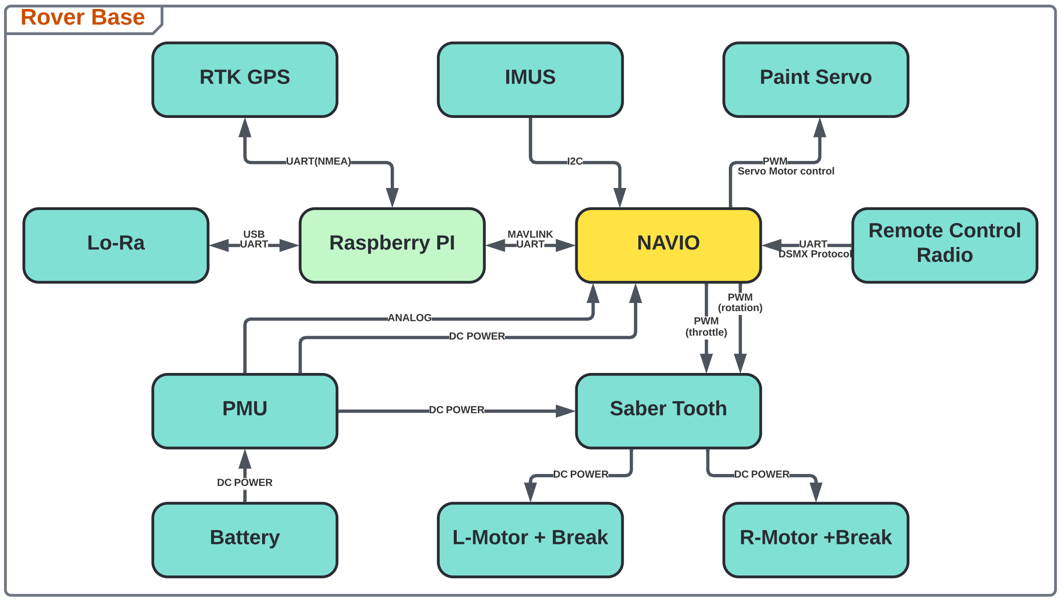

Figure 7: IMU Subsystem

3.8.1 Subsystem Hardware

This subsystems contains the MPU9250-9DOF IMU and thecLSM9DS1-9DOF IMU integrated in the Navio2 board.

3.9 Paint Servo

The Paint Servo controls the flow of paint to be dispensed on the ground by holding and releasing the spray nozzle down on a surveying paint can

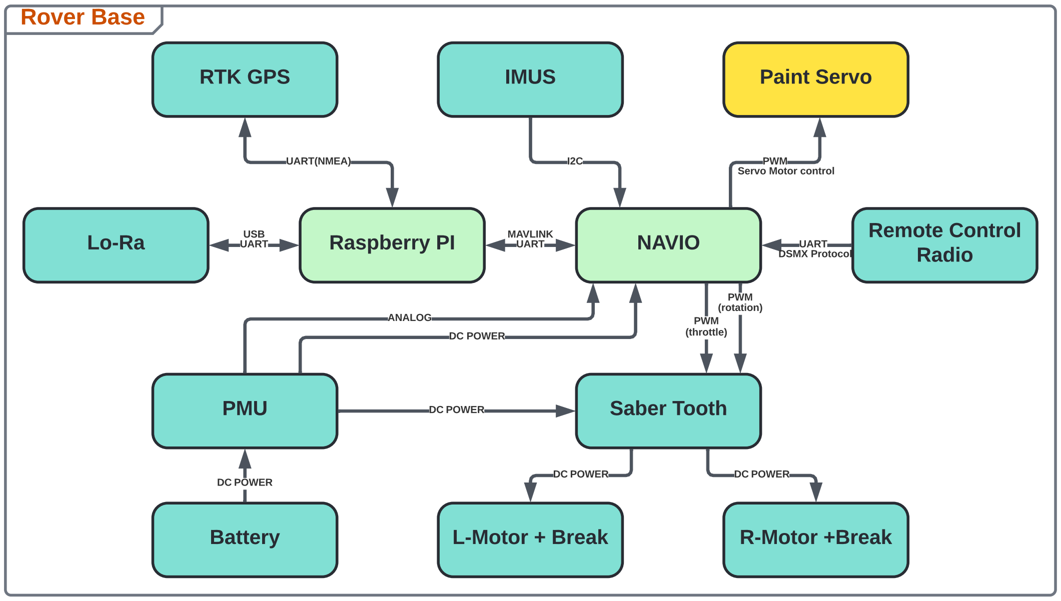

Figure 8: Example subsystem description diagram

3.9.1 Subsystem Hardware

This part is comprised of a servo controlled by the Navio2 board.

3.10 Remote Control Radio

The remote control radio is a device that is used to communicate between the remote controller and the NAVIO. It is able to send and receive signals from both subsystems with minimal latency loss.

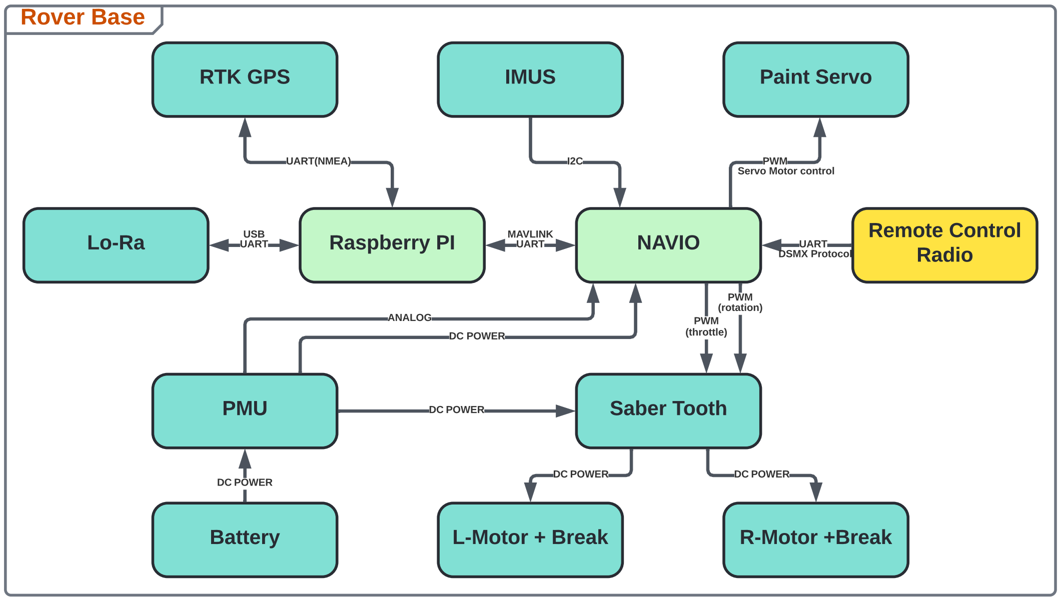

Figure 9: Remote Control Radio Subsystem

3.10.1 Subsystem Hardware

The remote control radio is classified as a full range sport receiver that has 6 channels and is able to transmit and receive the DSMX frequency.

3.10.2 Subsystem Software Dependencies

The remote control radio has ModelMatch and ServoSync technology, which enables it to be very compatible with other Spektrum technologies that use DSM2/DSMX frequencies.

3.11 Saber Tooth

The Saber Tooth motor controller converts PWM servo motor control signal to variable DC voltages that can be provided to the motors to control both speed and direction of the rover. The saber tooth also controls electromagnetic breaking in the motors to allow for emergency stop and other emergency features.

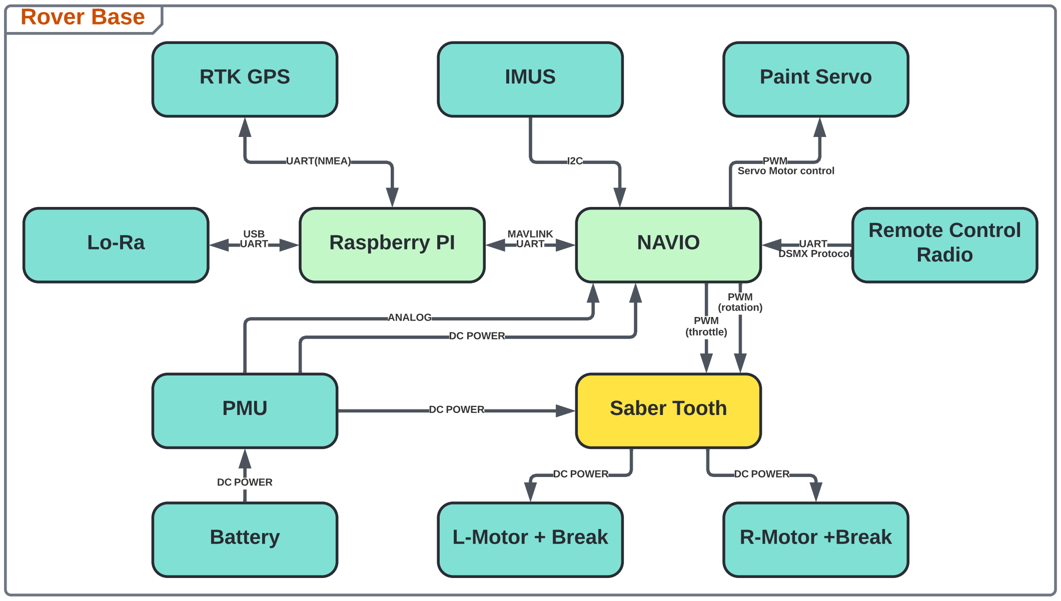

Figure 10: Saber Tooth Subsystem

3.11.1 Subsystem Hardware

This subsystem consists of a saber tooth module.

3.11.2 Subsystem Software Dependencies

This subsystem can communicate over serial to a device using DEscrive so that one can monitor motor power levels as well as current draws of this subsystems.

3.12 Left Motor

The Left motor is to drive the left wheel of the rover in the forward and reverse direction at the desired speed. The break helps to control its speed on the left side.

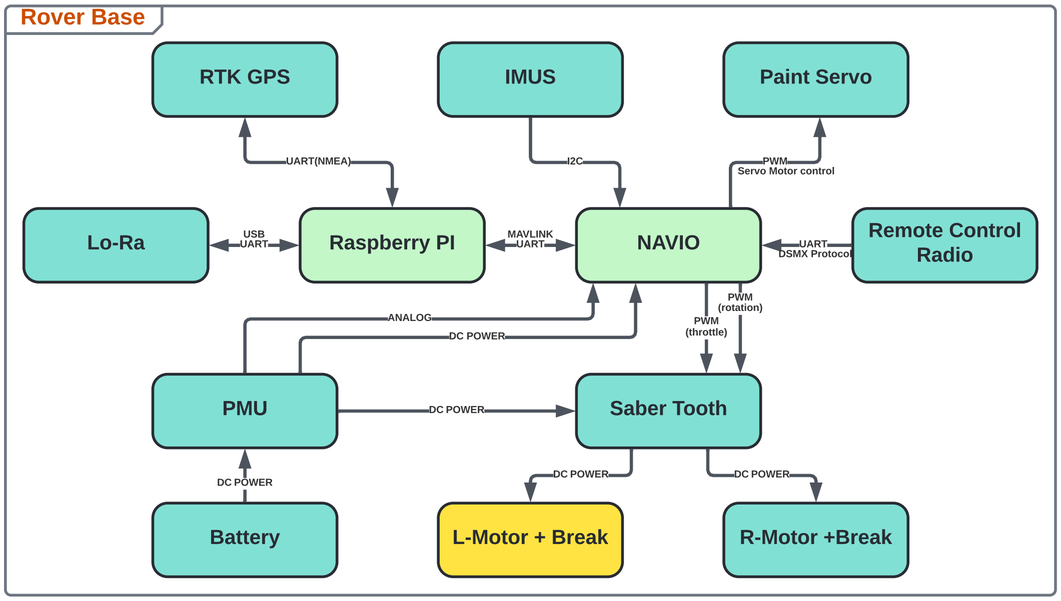

Figure 11: Left Motor Subsystem

3.12.1 Subsystem Hardware

It consists of left motor controller which communicates and gets its speed from Saber Tooth in the form of a DC Voltage.

3.13 Right Motor

The Right motor is to drive the right wheel of the rover in the forward and reverse direction at the desired speed. The break helps to control its speed on the right side.

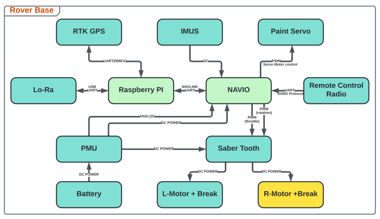

Figure 12: Right Motor Subsystem

3.13.1 Subsystem Hardware

It consists of right motor controller which communicates and gets its speed from Saber Tooth in the form of a DC Voltage.

3.14 PMU

The PMU (power management unit) provides power to all rover subsystems through the use of a power distribution panel. Additionally it monitors battery levels and provides real time current draw for the rover.

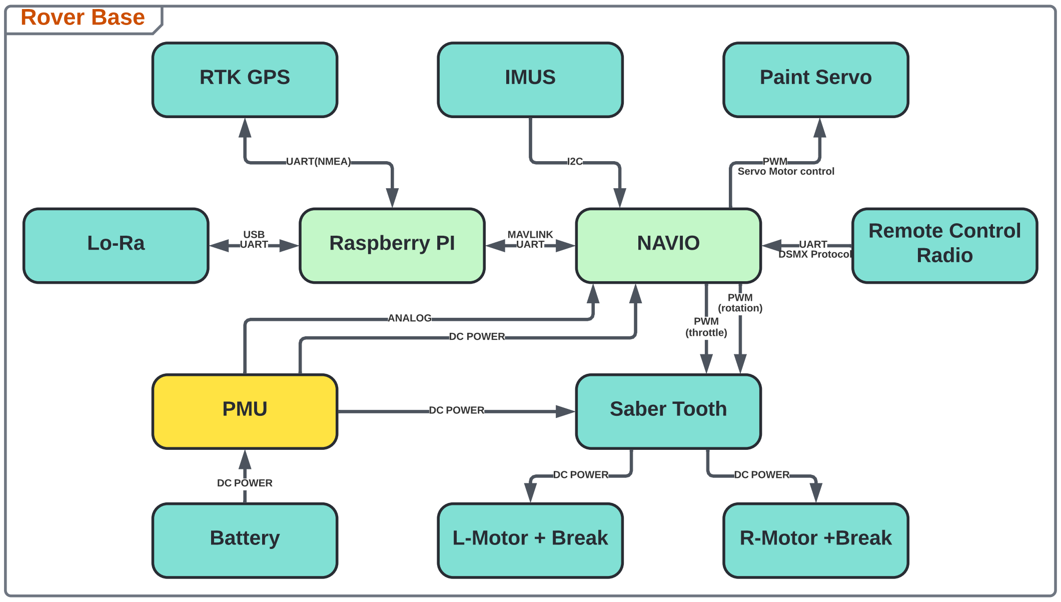

Figure 13: Example subsystem description diagram

3.14.1 Subsystem Hardware

The PMU is comprised of a power distribution panel and a BMS (battery managment system) that monitors voltage and power draw.

3.15 Battery

The Battery is used to provide power to all components and is comprised of two 12V Sealed Lead Acid Deep Cycle batteries that are wired in series to provide 24V to the rover. The Batteries are able to be recharged and this module includes the connection to the charger that allows this to be done externally. It provides DC Power to PMU which is responsible for power distribution.

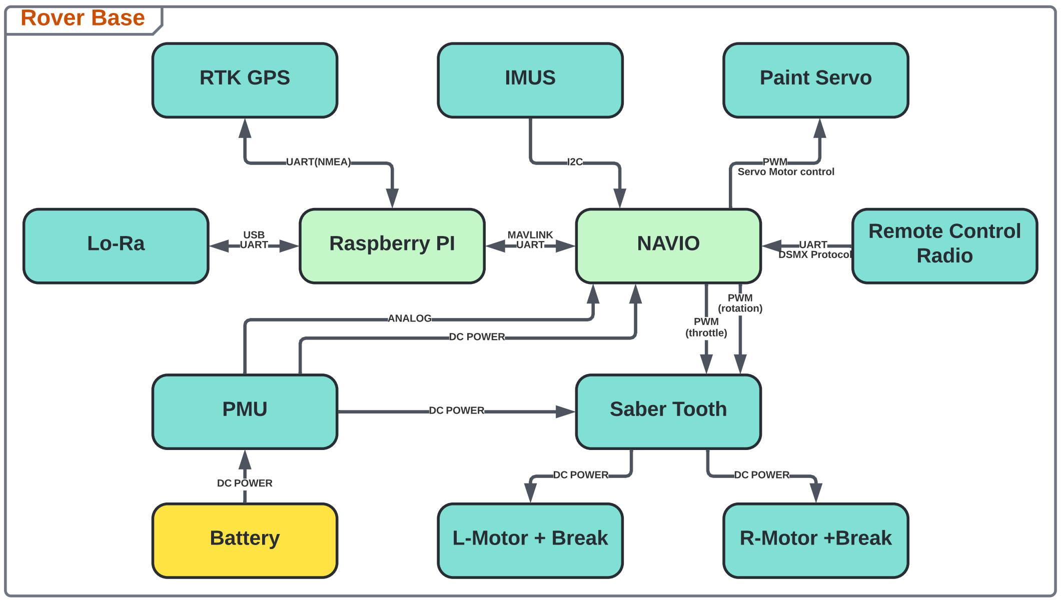

Figure 14: Example subsystem description diagram

3.15.1 Subsystem Hardware

Two 12V Sealed Lead Acid Deep Cycle Batteries

3.15.2 Subsystem Data Structures

DC Power supply

4 Controller/UI Layer Subsystems

The DX6e remote controller provides more control for the user as they are able to make manual commands for the rover.

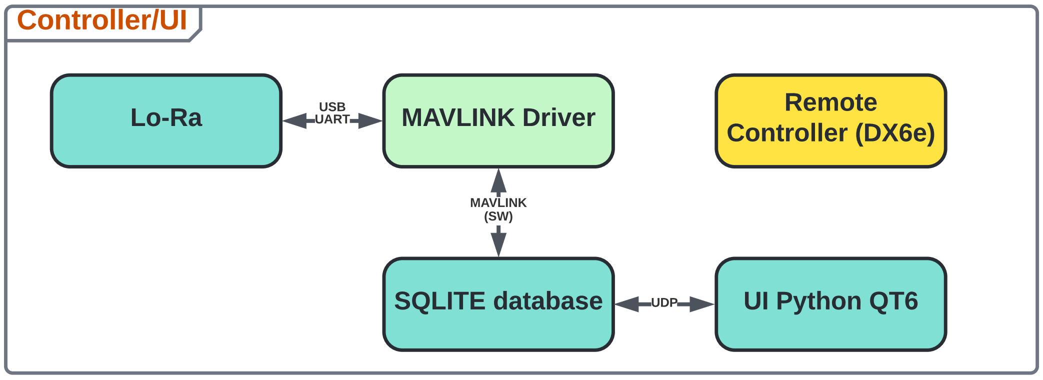

Figure 15: Remote Controller subsystem

4.1 Layer Hardware

The controller has a built in telemetry that is able to give information about the rover including: battery voltage, signal quality, etc. The many different switches and buttons on the controller are able to be used in order to drive manually, switch from manual to automatic driving, etc.

4.2 Layer Software Dependencies

The controller runs on the Spektrum AirWave software that allows for it to be programmed in many different ways. These setups are used for aviation, but the most of the information is still valuable to gather.

4.3 MAVLink Driver

MAVLink is a very lightweight messaging protocol for communicating with drones (and between onboard drone components). MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with retransmission.

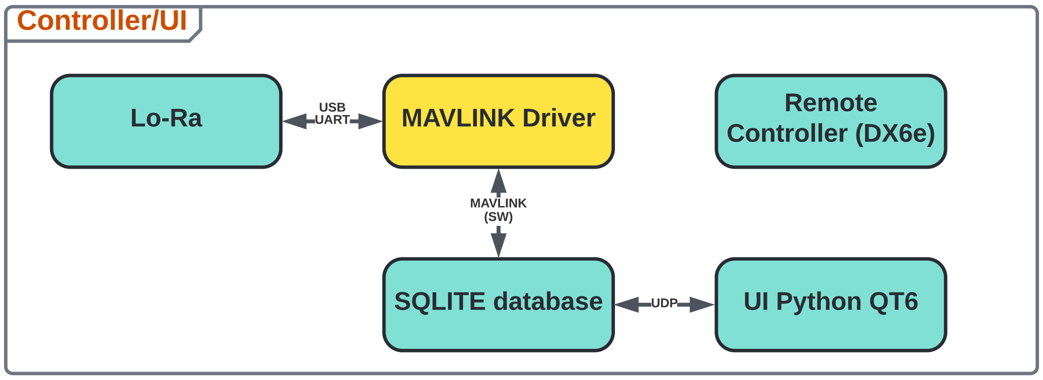

Figure 16: MavLinkDriver

4.3.1 MAVLink Driver Hardware

N/A

4.3.2 MAVLink Driver Operating System

MAVLink Driver will be compatible with the Windows 10 Operating System or Mac OS.

4.3.3 MAVLink Driver Software Dependencies

MAVLink protocol framing.

4.3.4 MAVLink Driver Programming Languages

The MAVLink communication protocol will be handled with Python using Pymavlink a low level and general purpose MAVLink message processing library.

4.3.5 MAVLink Driver Data Structures

The mavutil module will be used to set up and manage the communication channel. This module provides simple mechanisms for setting up links, sending and receiving messages, and querying some basic autopilot properties (e.g. flight mode). It provides access to the dialect module used for encoding/decoding/signing via an attribute (mav). Messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects build on top of this definition).

4.3.6 MAVLink Driver Data Processing

N/A

4.4 LoRa

The LoRa module on the controller acts as the transmitter/receiver used by the rover to communicate with the controller. This module is capable of fast long range communication with the rover. The LoRa module is responsible for sending all telemetry data to the rover as well as receiving all data that the rover sends. The LoRa module communicates with the controller via a UART connection.

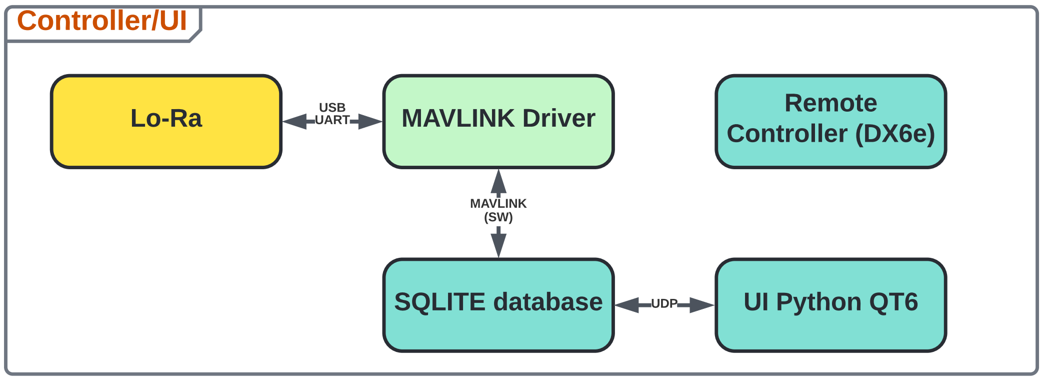

Figure 17: LoRa Subsystem

4.4.1 LoRa Hardware

The subsystem hardware consists of one half of a 433Mhz Holybro Telemetry Radio System with antennas.

4.4.2 LoRa Operating System

N/A

4.4.3 LoRa Software Dependencies

MAVLink protocol framing.

4.4.4 LoRa Programming Languages

The MAVLink communication protocol will be handled with Python using Pymavlink a low level and general purpose MAVLink message processing library.

4.4.5 LoRa Data Structures

The mavutil module will be used to set up and manage the communication channel. This module provides simple mechanisms for setting up links, sending and receiving messages, and querying some basic autopilot properties (e.g. flight mode). It provides access to the dialect module used for encoding/decoding/signing via an attribute (mav). Messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects build on top of this definition).

4.4.6 LoRa Data Processing

N/A

4.5 SQLITE Database Subsystem

The SQLITE Database is designed to hold all of the data collected by the rover and be able to send/receive information as needed. The database will hold information concerning the rover, important GPS data, and the battery's current status.

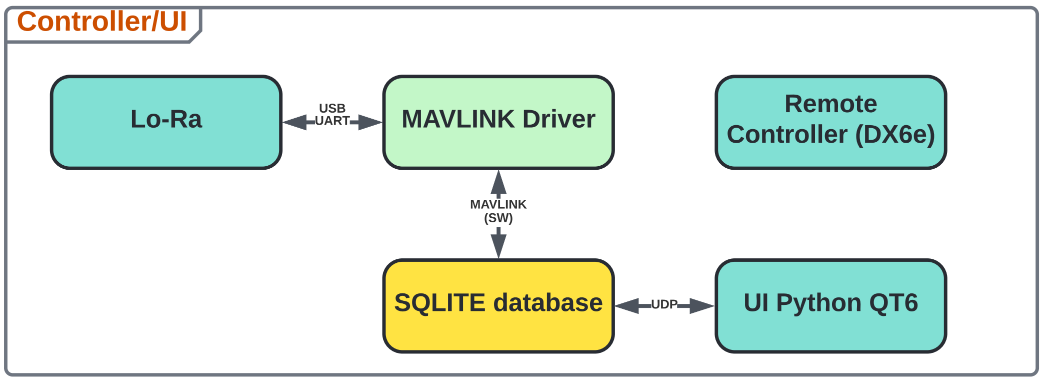

Figure 18: SQLITE subsystem

4.5.1 Subsystem Programming Languages

The database was coded using the SQL language, specifically using sqlite3.

4.5.2 Subsystem Data Structures

The database will connect with the Python UI using it's SQL supported libraries in order to communicate effectively.

4.6 PySimple Subsystem

PySimpleUI is a python package that enables to create the GUI. This subsystem is responsible for all the graphical user interface to the user. It consists of speed, acceleration, timer, battery status, inclination, compass, map etc.

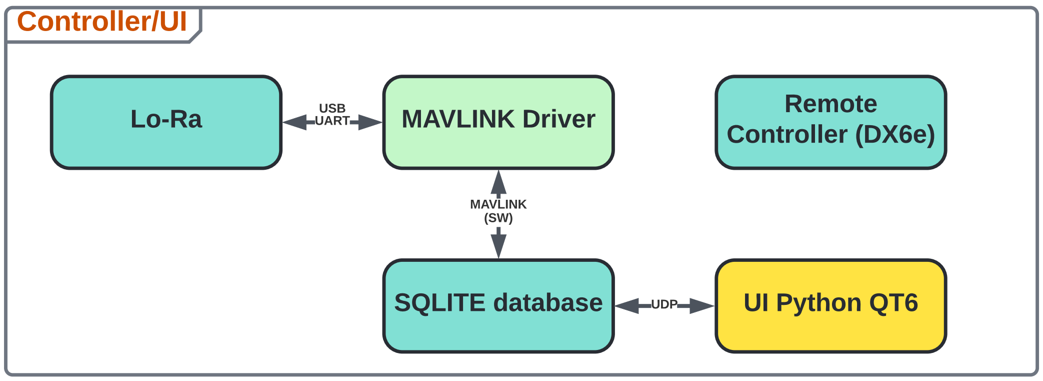

Figure 19: PySimpleUI subsystem

4.6.1 Subsystem Hardware

It will be able to display UI information on 800*480 resolution touchscreen.

4.6.2 Subsystem Operating System

It is compatible to Windows 10 operating system or ubuntu linux.

4.6.3 Subsystem Programming Languages

Python.

4.6.4 Subsystem Data Structures

It continuously receives data from SQLite Database Subsystem through UDP.

4.7 Remote Controller Subsystem

The DX6e remote controller provides more control for the user as they are able to make manual commands for the rover.

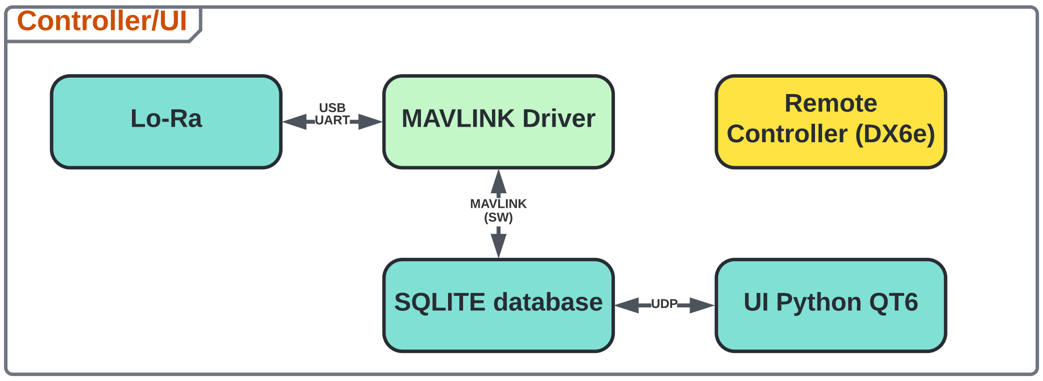

Figure 20: Remote Controller subsystem

4.8 Layer Hardware

The controller has a built in telemetry that is able to give information about the rover including: battery voltage, signal quality, etc. The many different switches and buttons on the controller are able to be used in order to drive manually, switch from manual to automatic driving, etc.

4.9 Layer Software Dependencies

The controller runs on the Spektrum AirWave software that allows for it to be programmed in many different ways. These setups are used for aviation, but the most of the information is still valuable to gather.

5 Appendix A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References