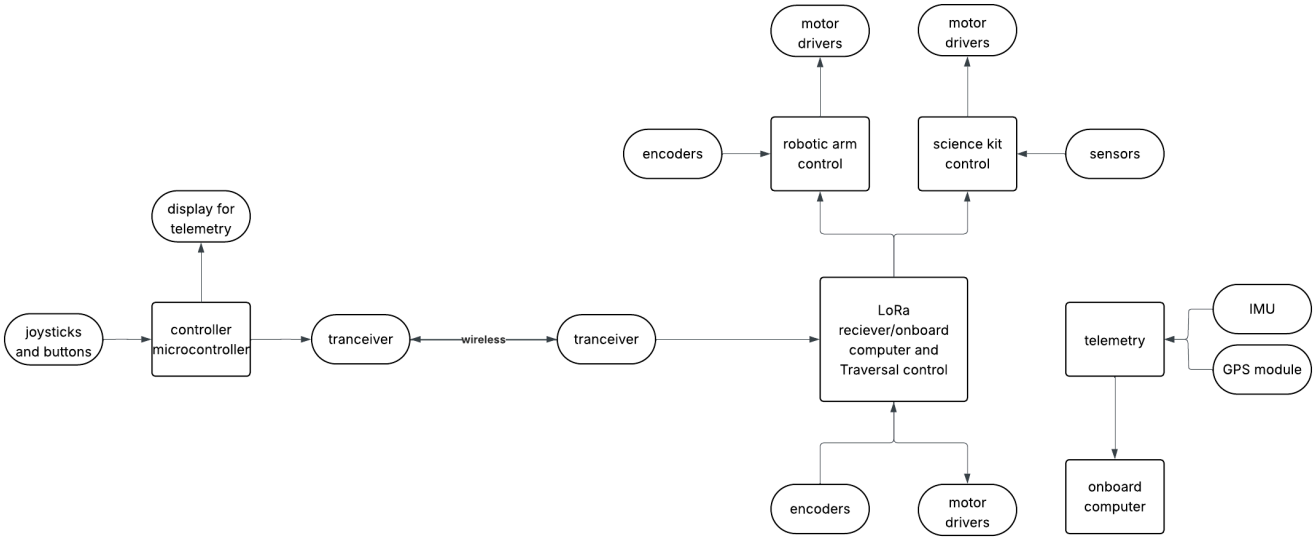
document for new electrical changes (rev 6)

problems with the current electrical system



architectural problems

- same board handles traversal and LoRa reception
- change in rob arm or science kit module requires change in traversal board for no reason
- IMU and other telemetry data are completely disconnected from the rest of the electrical systems (traction control would need IMU data)
- there's no centralized data hub, the onboard computer has to be manually connected to every board it needs data from
- long UART lines running across the rover over power components, UART is not noise resistant and we've already seen this being a problem when we try to transmit a lot of data over UART (between traversal and rob arm, we had to move a lot of the processing to rob arm esp which was already overloaded with processing. we had to look for other methods like dual core processing on the rob arm board)
- switching between autonomous mode and manual mode requires reprogramming the traversal board -which has caused confusion for software guys because they have to keep asking electrical for advice when programming their microcontroller code
- autonomous and GCS need to learn and write microcontroller code when all they need is data

power system issues

- the PDB lacks any sort of BMS -voltage, current, temperature monitoring
- buck converters are hard soldered onto the PDB
  - two bucks are currently shorted and replacing needs time consuming desoldering
- ESPs are powered through USB cables from the onboard computer
  - extra clutter inside the rover
  - difficult to manage with the number of ESPs growing inside the rover
  - micro B is fragile and we've had many cases of micro B ports ripping off of the esp dev boards
  - with a growing number of ESPs (there's already 4-5 ESPs) the laptop's (onboard computer) battery will drain quicker

excessive and disorganized wiring

- power wires:
  - small wire from PDB to motor driver x 6
  - long wire from each motor driver to each motor x 6
  - micro B cable for each ESP connected to a USB hub from the laptop -at least 4-5 ESPs
  - multiple PDB output wires for the robotic arm PDBs
- data wires:
  - 6 x 3 pin JSTs from traversal board to motor drivers
  - 6 encoder JSTs from motors to traversal ESP
  - 2 UART connections: traversal to robotic arm and traversal to science kit
  - more wire if BMS and telemetry are integrates
- more wires = more connectors = more points of failure
- these many connecting wires defeats the entire purpose of custom PCBs

changes

there are 4 types of changes here

1. modularization
2. standard power format
3. standard data format
4. connection topology

modularization

- new PDB consisting of:
  - slotable buck converters and motor drivers

    - battery management system to monitor voltage, current and temperature of the batteries and buck convertors
- separate modules for:
    - LoRa transceiver
    - dedicated board for communication with the onboard computer -this board is analogous to the LoRa transceiver board, it will communicate with the rest of the system the same way the transceiver board will
    - traversal board
    - combined IMU, GPS and sensors board
    - robotic arm and science kit boards
    - a dedicated board for anything else we want to add (a servo control board for cameras maybe, if we are doing it)

standard power format

- for power electronics, we will make all the components slotable and modular using direct XT connections (instead of making short male-female XT wires)
- all the compute boards could also be slotted onto a motherboard -which would act as the data interchange hub between the different modules. (if we use STMs here, we could make use of DMA, but we could even just use regular transistors or muxes)
    - every compute board would have a standard connector to connect with each other
    - we could 3D print a mounting with a small adapter PCB for each motor driver and buck convertor that brings all the connectors to one side (even the JSTs)

standard data format

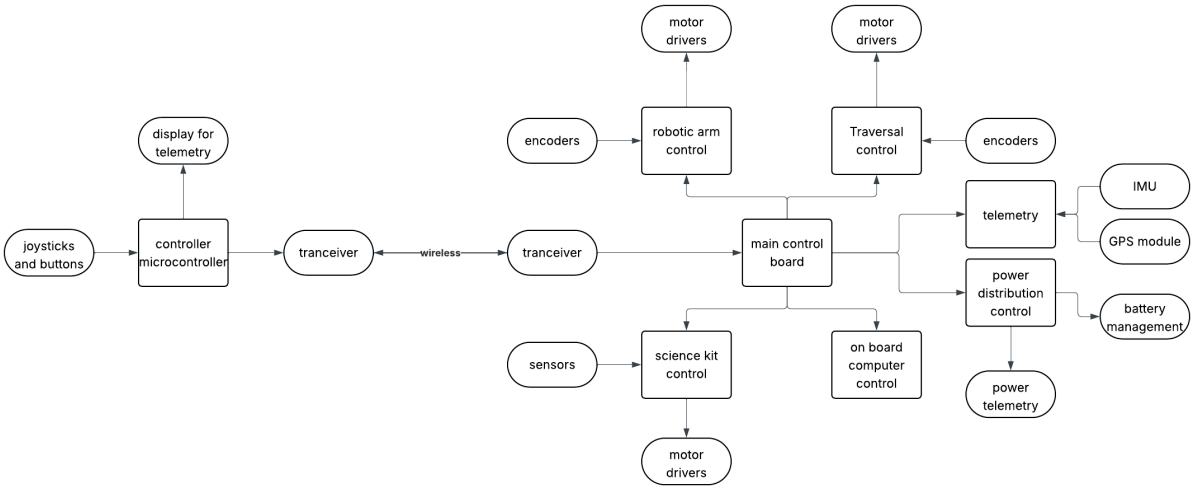
- all the control boards would use the same data format. example:
    - Traversal board only needs to receive `{velocity, angle}` and return `{wheel speed}`. it doesn't care if data comes from LoRa, onboard computer, or any other module — that's abstracted by the motherboard or bus.
    - now we can develop all three modules seperately.
        - we can change the transceiver from LoRa to zigbee, in the end we just have to package the data the same way.
        - we could implement traction control on the traversal board without changing anything on the transceiver or onboard computer comms board.
- there are several such examples.
    - GCS could decide to integrate BMS data into their GUI without touching the BMS board

- robotic arm might want to use data from both the controller and the onboard computer at the same time.
- the controller could include a small panel for telemetry data.
- we might implement traction control which would need IMU data (IMU data is directly connected to the onboard computer now, there's no connection with the traversal board)
- we could have a second simpler rover that's has the same dimensions as the current one that autonomous could develop using it -they'd just have to unplug their communication module, use it on the dummy rover and plug it right back into the main rover and it would work the same way regardless of what changes we make in the main rover

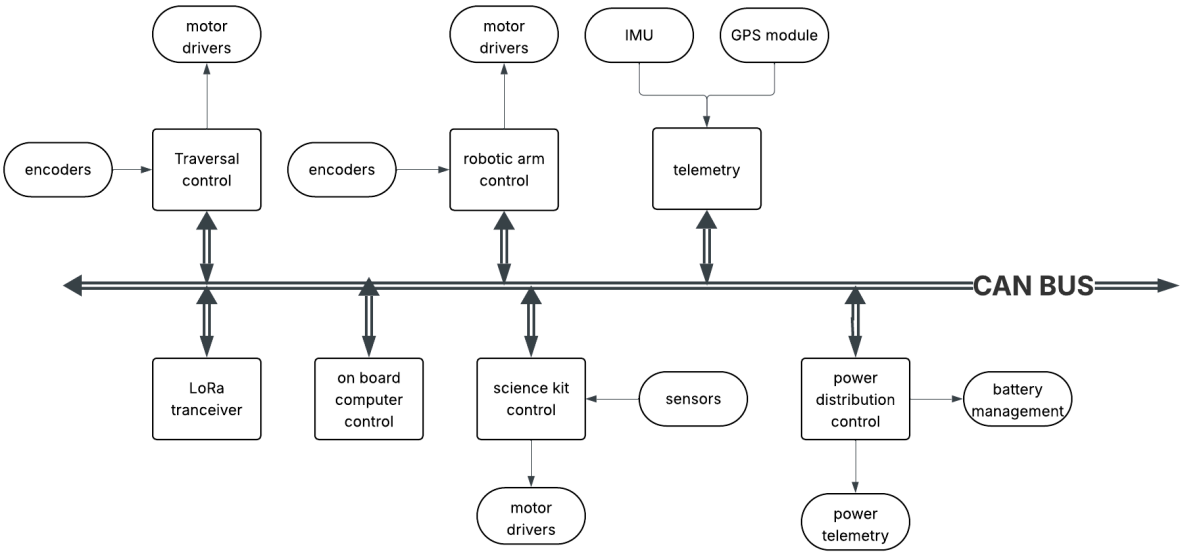
connection topology

all the different modules can be connected in two different ways we've thought of so far

- star topology:



- all the different modules would connect to a motherboard with a microcontroller dedicated to handle the data between these boards
- using Direct Memory Access would mitigate the issue of latency increase due to a middleman
- bus topology:



- there is no dedicated data control microcontroller, all the compute boards are connected to a bus and take turns communicating, have a priority order, etc.

- either of these topologies, we could use a differential signal (CAN or even USB) instead of UART for better signal integrity across the rover
  - both of these have their pros and cons -we could even use a mix of both these topologies

detailed changes

we described 4 types of changes earlier:

1. modularization
2. standard power format
3. standard data format
4. connection topology

now we define each subsystem and apply these 4 types of changes to them and their interlinks.

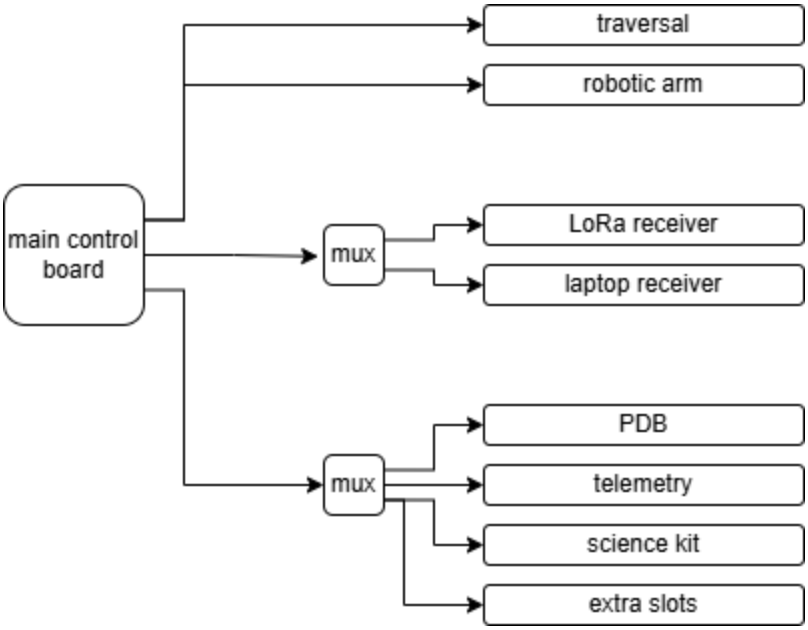
power and data

- we are not using CAN protocol. instead, we're using differential UART over twisted pair to improve noise immunity since wires will run near high-power lines.
  - CAN transceivers like TJA1050 were considered, but RS-485-style transceivers (like MAX3485) are a better fit for UART-based signaling.
- the connection topology is a star, with the main control board acting as the central data hub.
- no shared bus or arbitration logic needed — the main board directly routes data between modules.

- for permanent modules (like traversal, robotic arm), we will make dev boards that connect to the main board using shield-like headers.
- for distant modules (like telemetry, science kit, or PDB), we'll use 4-pin JSTs (VCC, GND, TX, RX).
- all slot-able power electronics will use XT60/XT90s soldered onto the boards.
- for removable or swappable connections, we'll use Anderson Powerpole connectors (easy to fix if damaged, no soldering needed).

main control board

- this is the central data hub. every board publishes data to it and subscribes to data from other boards — similar to ROS.
- this allows any new board to be added easily, as long as it uses the same data format.



  - 5 communication controllers:
    - traversal, robotic arm, and science kit will be always connected.
    - lora and onboard computer will be connected so that **only one is active at a time**.
    - remaining modules will connect through a mux for expandability and prototyping.
- uses an **STM32** to handle communication and routing using **DMA**.
  - there will be continuous DMA between traversal/rob arm and the receiver (LoRa or onboard computer).
  - remaining data transfers will be initiated on demand based on what data is subscribed to.
  - since there's a lot of data in and out, we could use a more powerful uC (like STM32H7) or dual uC

traversal control board

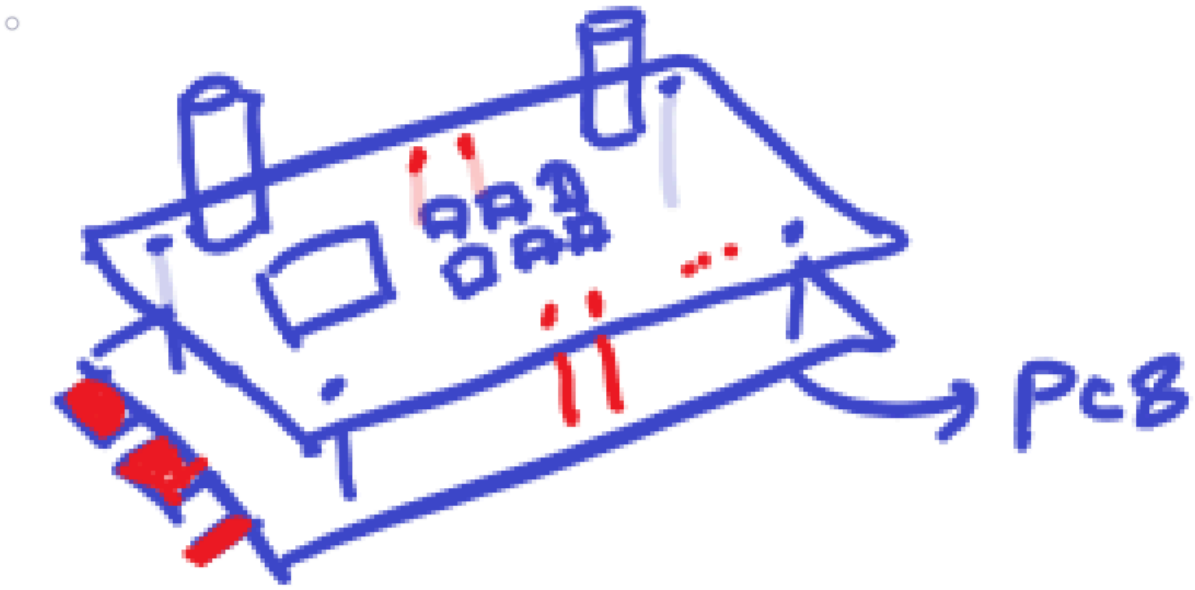
- shield-style module containing a microcontroller and differential UART transceiver.
- connects to main board using a **28-pin connector**:
  - 6 motors × (PWM, DIR, ENC A, ENC B) = 24 pins
  - +2 power pins, +2 UART
- all the motor drivers' control JSTs are connected to the PDB, and the connection goes: MD - PDB - Main - Traversal
- this way we can implement traction control or individual wheel control later on

LoRa Receiver

- simple module with LoRa module, microcontroller, and UART transceiver.
- connects to the main board using a 4-pin JST (VCC, GND, TX, RX).
- antenna mount will be part of the module (possibly 3D printed).
- communicates with the main controller.
- if LoRa fails, the onboard computer takes over as a backup communication link.

PDB-BMS

- new power distribution board with:
  - slotable buck converters:
    - 300W 12V buck for robotic arm
    - 5V buck for control boards
  - slotable motor drivers:
    - 6 XT60s for input power to motor drivers
    - 6 XT60s for motor driver output to motors
    - daughter PCBs to align motor driver ports to one side
  - 1 × 12V XT90 and 1 × 24V XT90 for robotic arm power



- PDB will include a **BMS** to:
  - monitor battery voltage, current, temperature
  - track power use on each port
  - report motor driver temperature
  - publish all this data to the main controller
- **still need to finalize exact BMS components (likely using current sensors and temp sensors)**
- **also need to look into motor driver protection and overcurrent mechanisms (maybe with polyfuses)**

onboard computer control board

- a small PCB that connects to the laptop via USB (like a USB drive) and to the main controller via 4-pin JST.
- contains only a microcontroller and UART transceiver.

telemetry

- remote module or shield, depending on where GCS wants to place sensors (IMU, GPS, etc).
- contains:
  - sensors selected by GCS
  - ESP microcontroller
  - UART transceiver
- connects to main controller via 4-pin JST.
- all sensor data will follow the same publish/subscribe format.

robotic arm

- this module depends on how rob arm want to connect with the main data hub

science kit

- this module depends on how the science kit will connect to the main data hub