ESP/ microcontroller power on control PCBs

ESP32 has a non linear ADC, so using it for encoders is unreliable

Need to modify control PCBs to include lora module instead of nrf module

Look into using 898/915mhz lora modules instead

No feedback to controller, the communication is only one way

Ackermann like steering with skid steering

Slipping when climbing slopes

Receiver PCB

Battery management system

Mode change in controller is too confusing

Arrangement of everything in the rover

Gripper fingers move forward when closing

Using the robotic arm to collect soil is very unstable. We need a dedicated driller

power wires going to traversal motors are not managed properly.

Wrist assembly is too heavy

**ESP/ microcontroller power on control PCBs**

The ESPs are being powered using micro B cables which is adding to the mess of different wires, would be better to power it using the PDB itself. A simple 2 pin JST should suffice, like the one we use to power the LoRa right now

**Need to modify control PCBs to include lora module instead of nrf module**

The current PCBs have nrf module pinout, need to modify it to have sx1278 pinouts.

While we’re at it,

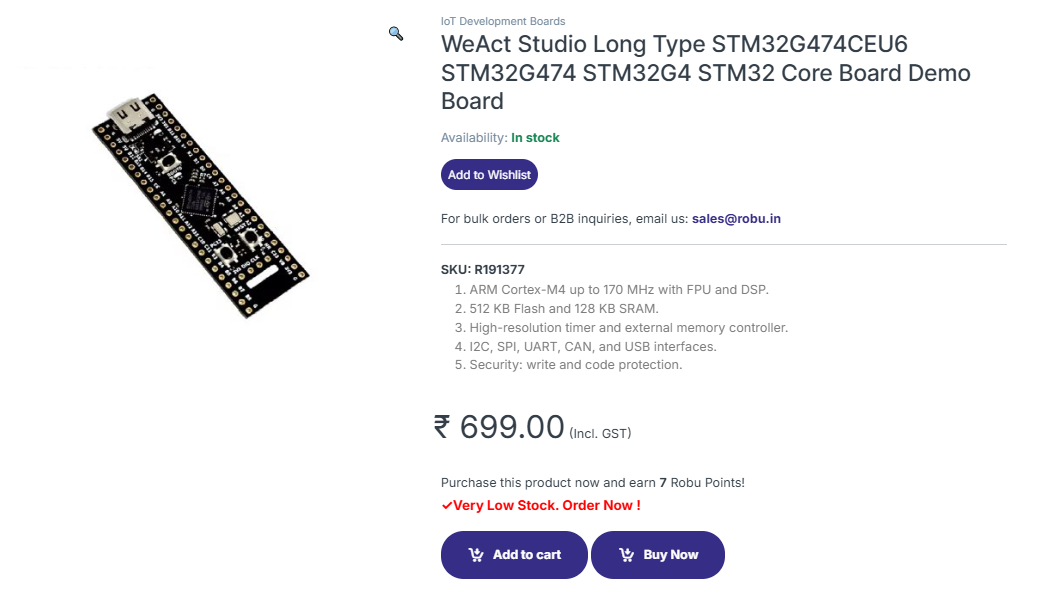
**Microcontroller and lora chips embedded onto the control PCBs**

Instead of using dev modules, we can solder the microcontroller and LoRa chips onto the control PCBs. With this you get to use chips with more GPIO pins, like STM32G4, it has like 50 GPIO pins while still being small, so you can, for example have a lot of encoders connected to the robotic arm PCB instead of relying on multiple microcontrollers and communicating between them

This would also be much cheaper than buying dev modules. For example, again, STM32G4 dev board is 1700rs, even the smaller dev board are upto 700rs. While the MCU chip is only 250-350rs. The cost of the entire control board would possibly come down to less than the price of a dev board because we’ll be mitigating a lot of unnecessary parts.

(I mean, honestly, the cost isn’t even a reason. It’s just cooler to embed the chip onto the traversal pcb. We would’ve essentially made a product, and not just a prototype. Now isn’t that cool.)





**898/915mhz lora modules instead 433mhz**

We’re using sx1278 modules right now, which operate at 433mhz rf. We could switch to RFM95W (915mhz) or RFM69W (898mhz). the main reason for this switch is that it’s much easier finding antennas for 900mhz. also RFM modules might have a better range, need to test.

**Receiver PCB**

Make the control PCBs more modular by making the receiver into it’s own PCB. All the other control PCBs connect to the receiver PCB in a star topology and receiver receives and sends the data to each.

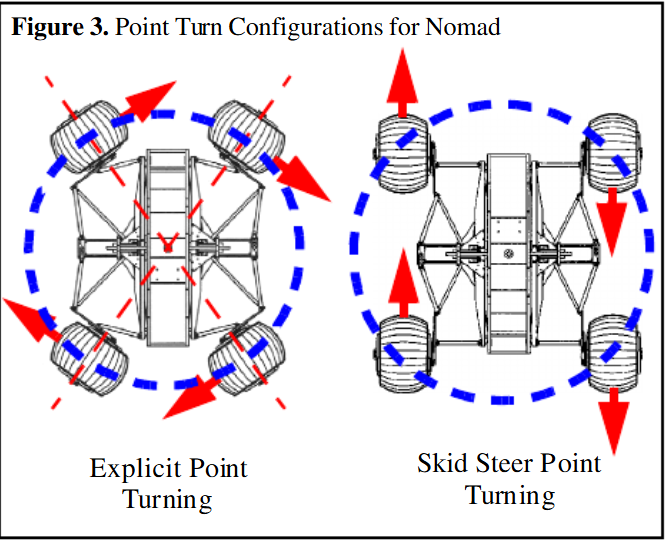
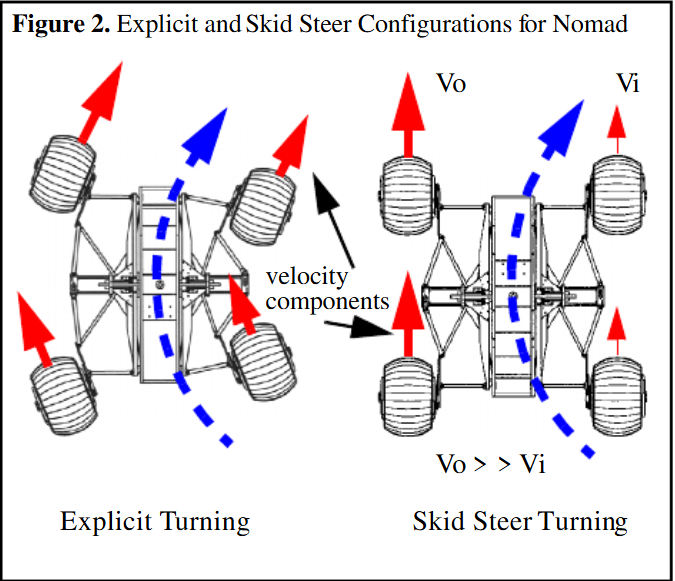
**power wires going to traversal motors are not managed properly.**

We had many times when the motor power wires got ripped apart in the middle or at the brushes. The motor wires are very loosely connected and the contacts to the brushes slightly exposed. The encoder wires have all been removed too. We need to route the power wires through a separate tubing or inside the carbon fiber rods to keep them in place. need to mitigate any amount of movement of those wires.

Instead of having to push the motor wires through the chassis, it would be better and safer for the motor drivers if we simply have some connecters hanging out of the chassis that the motors connect to. this would fix the problem where we have to constantly wiggle the xt connector soldered onto the motor driver to connect the motors (this has caused the xt connecters to loosen and fall off many times already)

We could implement the same idea for any connection between the inside and outside of the rover – robotic arm power, encoders

**Ackermann like (explicit) steering with skid steering**

Pics and paper: https://www.researchgate.net/publication/237768170\_Effect\_of\_Tire\_Design\_and\_Steering\_Mode\_on\_Robotic\_Mobility\_in\_Barren\_Terrain

The rover cannot traverse longer turns smoothly without stopping to correct it’s direction every now and then. this causes jerk and costs time with correcting the position. With irc25 this wasn’t a problem because of the tiny mission sites, but it could be a problem with larger sites where the destination is farther.

This could be as simple as having a mode where two joysticks control the left side and the right side independantly. Or it could be done in a more complex way with controls like an actual RC car.

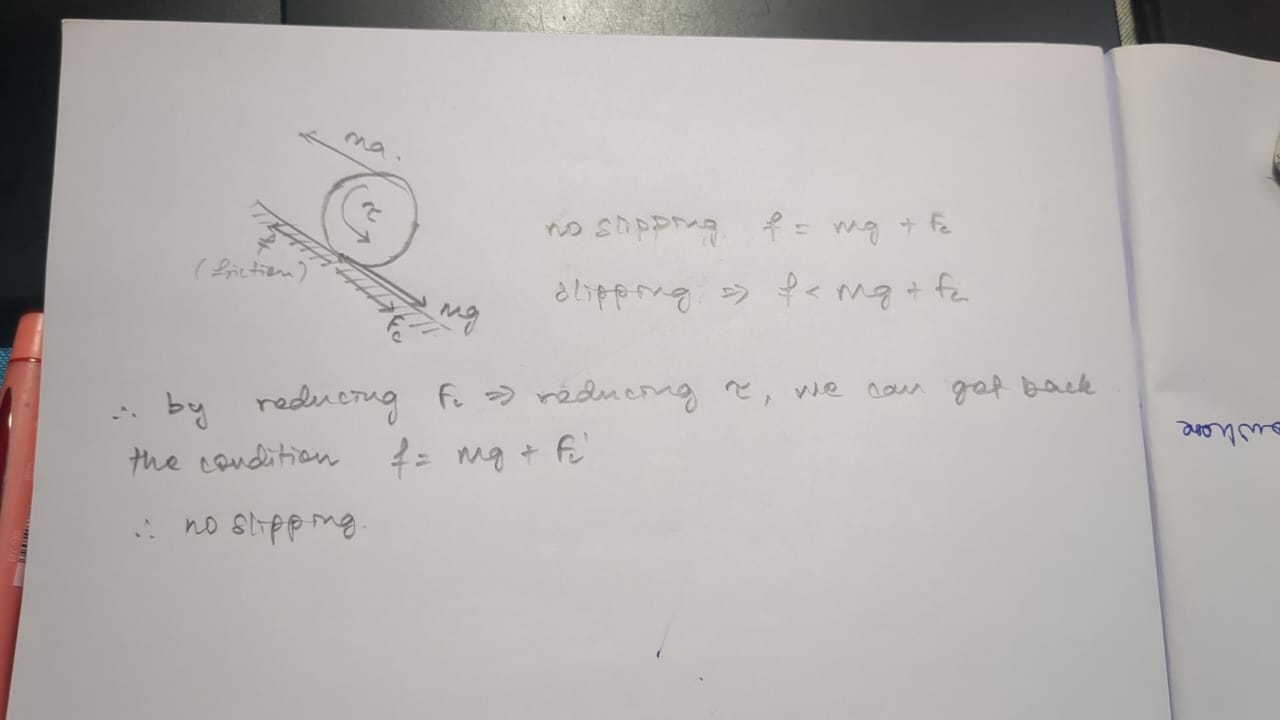
**Half the PWM, half the power, (around) half the torque**

When we did testing on the steep hill with lose soil, we had a harder time climbing with the slow and super slow modes. The reasoning is simple, the slow speed, for example is only a pwm of 128, so 50% duty cycle. For a 24v motor the motor driver gives out a smoothened out 12v output.

The Torque is directly proportional to the current; T = K \* I; where K is some constant. And the current I = V/R.

so halving the voltage halves the current, which doesn’t halve, but definitely decreases the torque.

so If we need to climb up a steep hill with lose soil, we have one option excluding making a transmission box: give full PWM, which only ends up making the wheels slip because there isn’t enough friction to counteract both the weight of the rover and the contact force of the wheels when it rotates.



(\*mgsinx or mgcosx not mg)

So to reduce slipping, we need to reduce torque. When run at a lower PWM, the torque is reduced.

This system is already very very popular -it’s literally just ABS in reverse. ABS works to prevent locking i.e., skidding while braking quickly -we just need to take that concept and apply it in reverse, for accelerating.

Whenever we detect that the wheels are slipping, we reduce the PWM and hence the torque such that the wheels aren’t slipping anymore – Reverse ABS! or Anti-slip Acceleration System (AAS)

This would cause unpredicted behavior if implemented for when steering obviously, since we’re using “skid” steering -but we’ll just disable AAS when the wheels aren’t matching in speed (if we implement explicit turning) and direction. AAS would be active only when the rover is moving front or back. We could also make it simpler by just adding a AAS toggle.

Note: gradual increase would also give a better result than just full pwm, but AAS would be even better than gradual increase, since it’s a smarter system.

~~this might not be viable, but look at dirt bikes. Single cylinder engines, a power stroke only once every four strokes. So only 180deg of it’s 720deg crankshaft rotation delivers power -at 1~~~~st~~ ~~gear, assuming 0.25 gear ratio (I think it’s usually 0.225 or something idk), that’s only 1/4~~~~th~~ ~~of each revolution of the wheel that delivers power. Only 90deg of power delivery to the ground, and 270deg of rest for the tire to regain traction. Im not cooking this up by myself. I learn this from this video:~~

[~~https://www.youtube.com/watch?v=aOTz0Ol8fLA~~](https://www.youtube.com/watch?v=aOTz0Ol8fLA)

~~the first example in the video, he explains about single cylinder engines. It should explain what im trying to better than I did.~~

~~We could implement this onto the wheels somehow. There’s the obvious problem of high current draw on the motor driver due to the constant turning on and off, but it could be mitigated to an extent with much lower frequency~~

**Dedicated driller/Auger**

I feel Using the robotic arm to collect soil is very unstable. We need a dedicated driller/auger for soil collection.

**Controller:**

**Like mentioned earlier, 900mhz LoRa modules instead of 433mhz LoRa**

**No feedback to controller, the communication is only one way**

We need to implement duplex comms in LoRa to get basic data feedback. right now the controller only sends data without any form of feedback and it’s causing confusion like controller mode change.

**Mode change in controller is too confusing**

Need to add a 4th joystick to the controller on robotic arm mode for minute traversal controls because switching it to traversal is hectic

Need to change mode switch to a rotary switch because it’s confusing to use buttons

**Controller PCB**

this one’s obvious, the controller is made on a proto-board. We need to make it into a real thing by designing a PCB and a casing.

**ESP32’s non linear and noisy ADC is giving bad analog readings**

Ive attached screenshots from some sources to prove this. there are multiple sources and threads saying the same. This is a small issue for the controller, but a serious issue for any sort of pot encoder we’d be using

https://microcontrollerslab.com/adc-esp32-measuring-voltage-example/

<https://arduinokitproject.com/esp32-basics-adc/>

most technical data: <https://esp32.com/viewtopic.php?t=38340>

ive also run out of gpio pins in the ESP32 for the controller, so that’s an issue

the simplest fix to this would be to move to a different microcontroller. One with a reliable ADC and more gpio pins.

