Mobility Management

The idea is to create a carriage that moves around while carrying and being controlled by the electronics assembly.

Design consists of two parts: driving assembly and steering assembly.

Main design considerations are:

1. The overall length needs to be less than 20 cm, for parking
2. Keep the carriage as low as possible, so robot is not a skyscraper box with wheels.

21 mm Lego City tires were used. Since they have good shape and size for this model.

Carriage ended up being 3 cm tall (including the 2 cm wheels), with a sg90 servo motor attached in the front.

It contains a motor driver, and a 4 gear differential.

Overall ground clearance is 2.5 mm.

Originally speed was intended to be 2 m/s, but was reduced to 1 m/s.

Program runs at 35 [ms/cycle], this measure has to be as low as possible, since robot doesn't detect/steer while program is calculating. 33.3 [ms/cycle] out of 35 [ms/cycle] are taken by camera (capturing video at 30 fps).

Parking is the most delicate part of the mission,

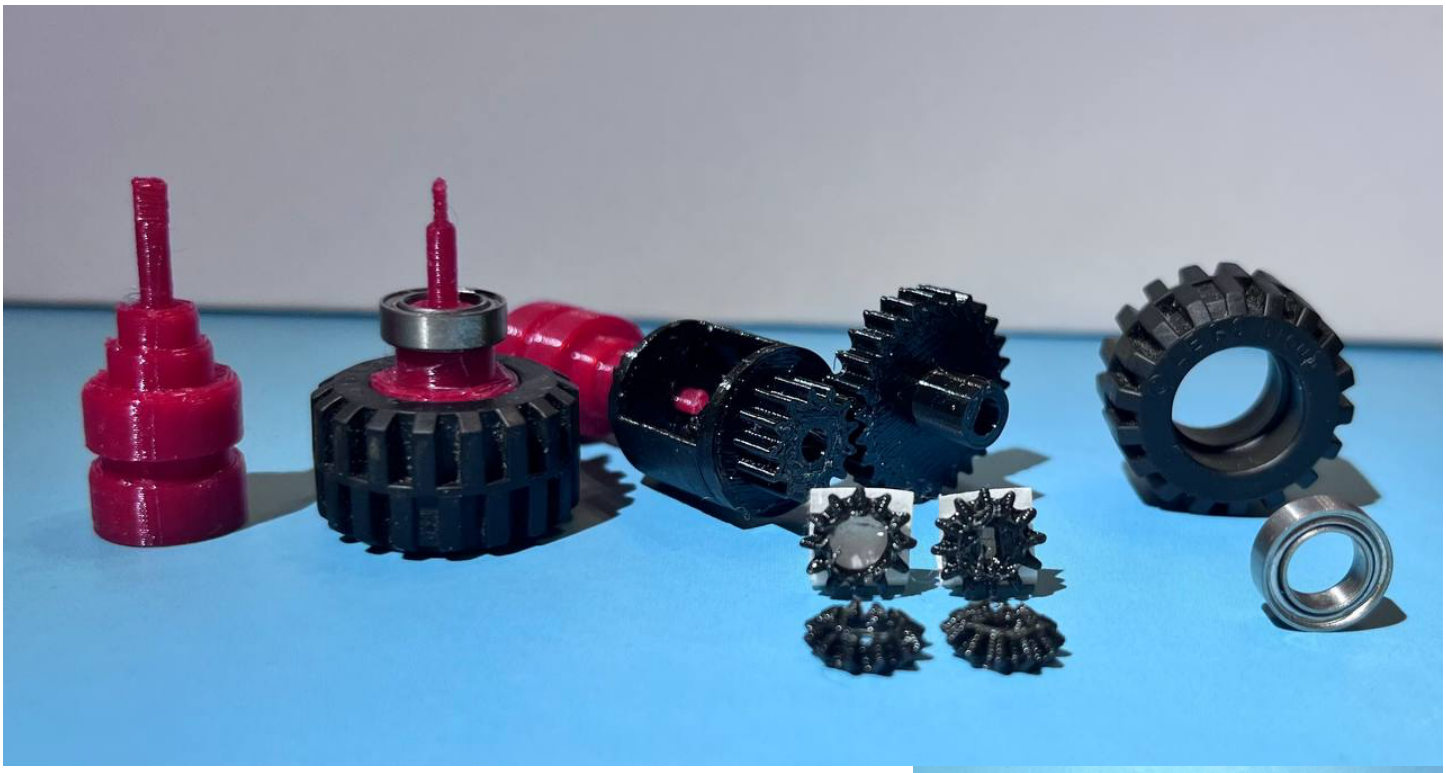
thus each simplification of it is directly simplifying the robot.

To avoid parallel parking, robot length is limited to 20 cm.

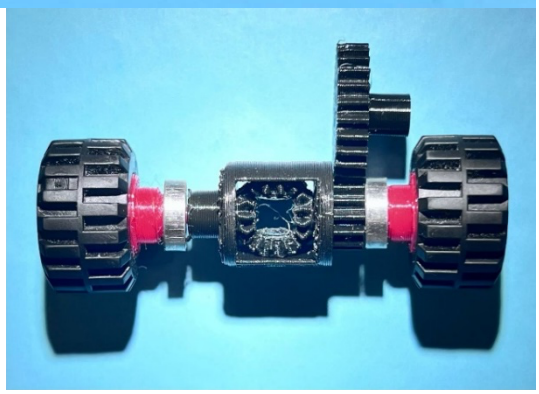
Since that is the length of the parking barriers.

Making it park linearly removes the need for motor encoder.

Driving assembly



Drive axle assembly consists of 12 parts.



- 2 LEGO City 21 mm tires
- 2 Wheel axles
- 2 Ball bearings
- 2 differential middle gears
- 2 differential axle gears
- Differential
- Motor gear

Motor used is a N20 5V 500 RPM Motor. It provides good torque and RPM, while being tiny in size. We had experience working with it, so designing for it was easy. It made it the overall best choice for this robot.

To reduce friction and ware, wheel axles are attached using ball bearings.

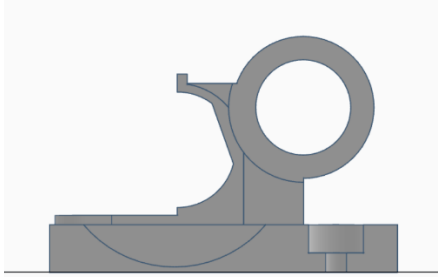
Original 2 m/s design required them to rotate at 1819 rpm. Now at 1 m/s, it is 909 rpm.

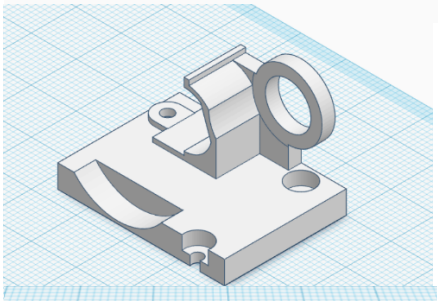
$$2\left[\frac{m}{s}\right] / (0.021\,[m] * \pi)\left[\frac{m}{rotation}\right] \approx 30.32\,\left[\frac{rotation}{second}\right] \approx 1819\left[\frac{rotation}{minute}\right]$$

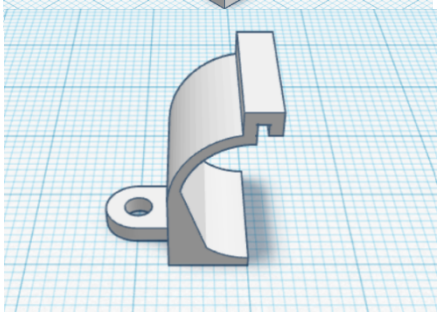
$$1\left[\frac{m}{s}\right] / (0.021\,[m] * \pi)\left[\frac{m}{rotation}\right] \approx 15.16\,\left[\frac{rotation}{second}\right] \approx\ 909\,\left[\frac{rotation}{minute}\right]$$

Ground clearance made differential 15 mm in diameter.

Due to that, differential axle gears have to be small, making axle connector tiny. (2.5 x 1 x 3 mm)







The wheel axles are thin at the end because they have to go through differential gear. Which originally was smaller.

500 rpm motor needed 1:4 gear ratio to get 2 m/s speed.

Since gear size is proportional to the ratio, it was done for optimization.

But at 1 m/s, only a 1:2 gear ratio is needed.

That increases the size of the differential gear.

To house this assembly, ball bearings are press-fitted into base.

Motor is press-fitted at a specific angle, to be perpendicular to the differential.

This allows to mount it closer to the axle.

Base has cutouts for the differential and motor gear, screw holes for electrical assembly and motor cover.

The motor is housed at a specific height. That when in 1:4 gear ratio variant, the motor gear is at 2.5 mm from the ground. Which is the ground clearance.

Redesigning motor housing for 1:2 gear ratio could introduce new problems.

And thus would be an inefficient use of time, since 1:2 gear ratio variant can work with 1:4 gear ratio base. It therefore, wasn't updated.

Possible driving improvements

Driving assembly is unique, due to it being the only one without design iterations.

It doesn't have a lot of different design options though, but it still has its design problems.

While it is possible to make it go at original 2 m/s, that wouldn't work.

Motor, UPS HAT and likely motor driver would have to be replaced, since those can't handle the power needed.

Camera can't stay too, it can reliably shoot at 30 fps, that makes program cycle time too long to react at 2 m/s.

(30 fps camera is contributing 33.3 ms/cycle to the program speed)

$$2\left[\frac{m}{s}\right] * 35\left[\frac{ms}{cycle}\right] = 70\left[\frac{mm}{cycle}\right] = 7\left[\frac{cm}{cycle}\right]$$

That means that it makes a decision/discovery only *once* per 7 cm of travel.

It would likely be able to avoid traffic lights and walls, but is also likely to miss the map lines and sudden encounters.

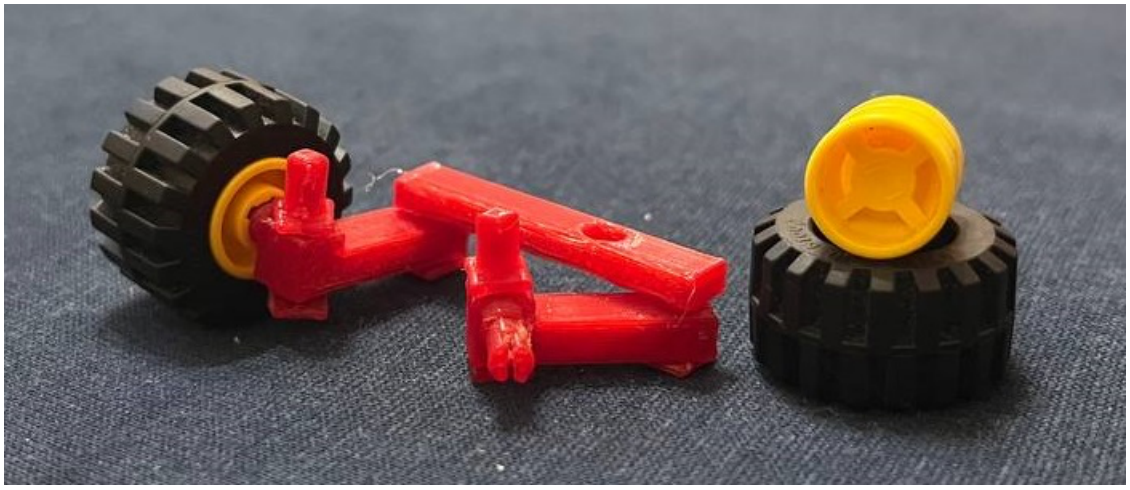
Parking wouldn't break, since it slows to 0.5 m/s for it.

Making it go at 2 m/s means redoing the whole robot, which is an overkill for a robot that already drives reasonably fast.

So the real possible improvements would be adapting it for 1 m/s.

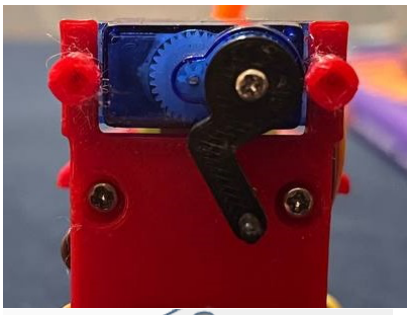
Wheel axles should be thicker, and motor repositioned. Everything else adapted for these changes. It will make the robot harder to physically break and will prolong its life.

Steering assembly



The steering assembly is simplest by design, yet is the most complex to design.

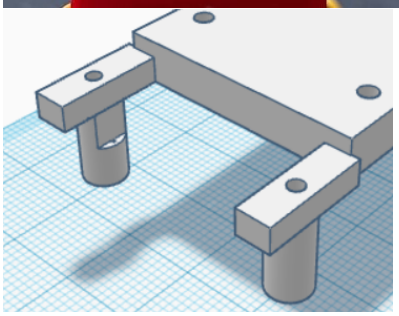
It can steer 60 degrees in both direction, although it is limited due to wheel drag to 45 degrees in program.



It consists of:

- Print-in-place lever-arm steering assembly
- Servo arm
- 2 LEGO City 21 mm tires
- 2 LEGO City wheels.

Motor chosen was sg90 (also known as ‘blue servo’), It is often used for projects of this size. It works well and is small. Also I had a lot of these to use.



The wheel levers have LEGO City wheel attachments, which reduces friction and ware during movement.

Servo arm is being held by the servo, and connects to lever-arm steering assembly using a simple rotating axle.

Lever-arm steering attaches to the base using a special press-fit diamond connector.

The base itself has two lift arms, with servo screw holes and diamond connector holes.

This creates a simple lever-arm steering mechanism.

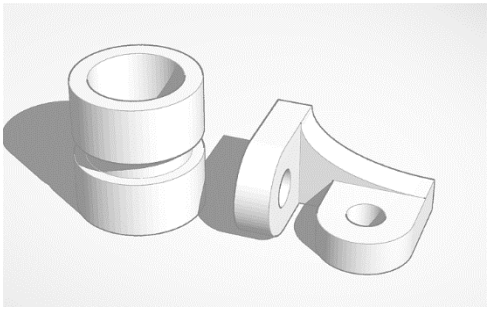
The development of this assembly wasn’t that simple though.

Steering development

During the development of the steering system, there were many different designs and even more of their iterations.

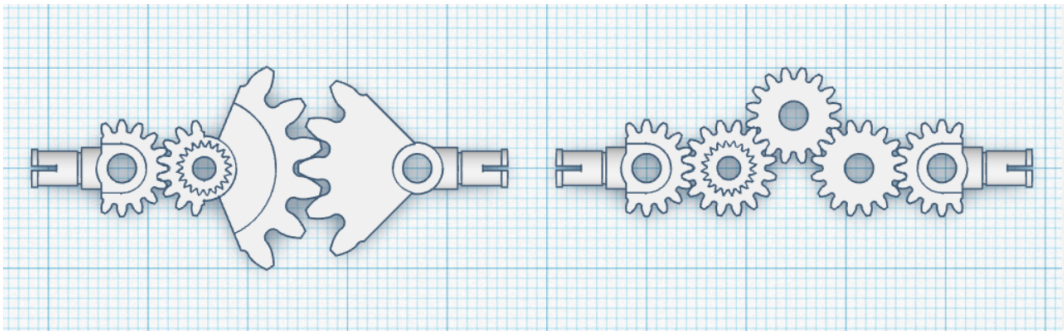
They are here to show the development process and mistakes made.

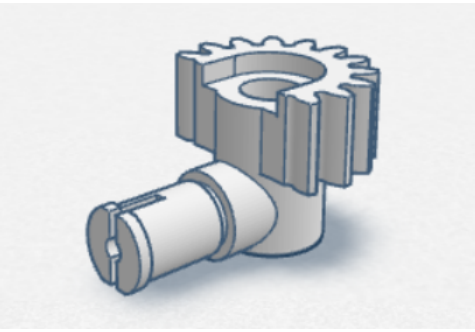
So if you want to modify this design or build your own, you'll know what not to do and issues that might arise in the development process.



Originally wheels were secured using screws. Those unscrewed themselves, despite many washers used in the axle. 4 in total, it didn't solve the issue.

Then, to solve this issue the design was changed to mimic the LEGO connector used for this wheel. It allowed to simplify the design, and proved reliable in use.





First developed design was the gear-driven steering system.

^ That group photo for 3 and 5 geared design is for presentation purpose, as no full group model has been recovered.

The individual gear models as seen in the 5-gear group photo are real designs, and a few of them were printed.

Geared design allowed to position the servo lower, than the lever-arm steering design would.

Reason being the problem which is back in the current design: Robot is challenging to turn on.

Servo blocks the turn-on switch on the UPS HAT. And it cannot be mirrored, as it would then block the charging port.

As seen above, there were two versions of this design: 3 gear design and 5 gear design.

The number of gears has to be odd, to make wheels' steer in the same direction.

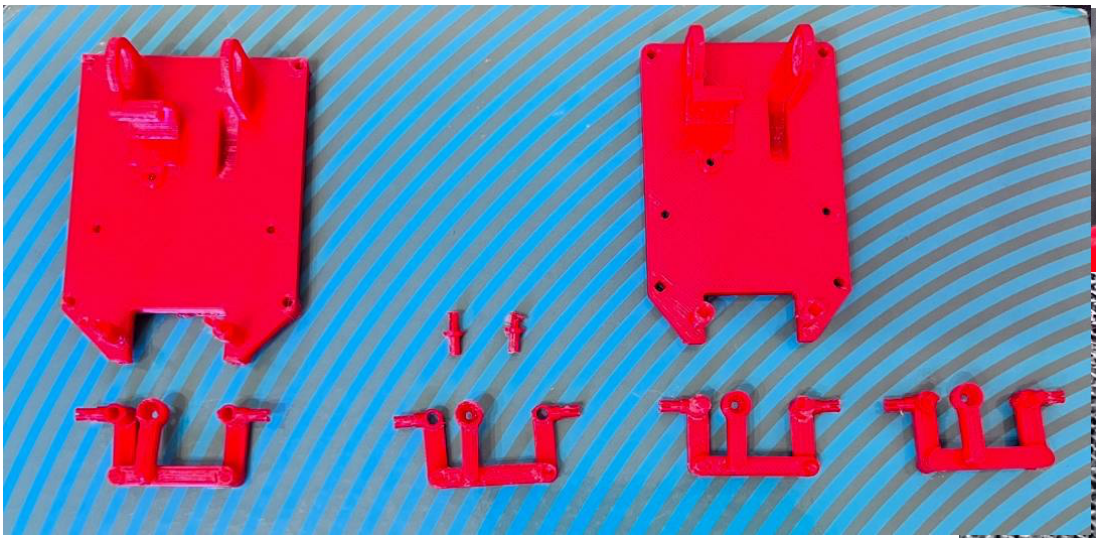
3 gear design is simpler, however it is less compact, since one of the gears having big radius.

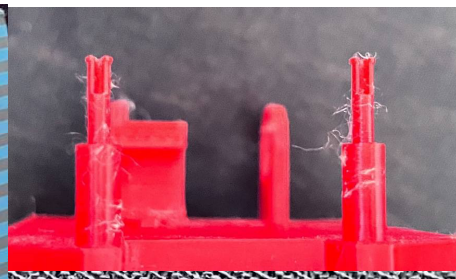
5 gear design is more compact, and allowed for higher steering angles - full 90 degrees.

The 5 gear design, despite superior to the 3 gear design by being more compact, was very complex.

With a lot of very precise small parts to make it function.

Then, the blocked turn-on switch problem disappeared due to a change of the UPS HAT.





Thus the development switched to a lever-arm steering design.

It is a bit less compact than the 5 gear design, but allowed for great simplicity of the 3 gear design.

There were 4 main designs for lever-arm steering.

Design 1 had problems with pins that hold the wheel levers.

They broke very easily due to their length. Their thickness at the end due to being split was less than a millimeter.

And since those were attached to the base. If one of them broke, the whole base had to be reprinted. Which is 2 hours!

Design 2 fixed it by creating the diamond connector.

The base only had to have rhomb shaped holes, and the steering assembly had to have a connector which friction fits to it.

Thus if a pin broke it took mere minutes to reprint, instead of 2 hours for the whole base.

The problem with this design was that the pins still broke.

Although it still could hold a bit instead of bending instantly like in design 1.

These issues are likely due to pins being not viable rotating joint method at this scale for the load they need to handle.

Design 3 used a new technique I learned – printing in place.

The pins were replaced with an axle built-in the wheel lever.

The problem with design 3 was that the levers were bending down due to having a lot backlash.

The wheel axles bended down, while servo arm remained at level.

It worked, but looked unreliable.

Design 4 had the steering link printed in place into the servo levers.

Since it was secured at level by the servo axle, it lifted wheel levers up.

That reduced the bending.

After that there were only changes in the servo arm, and small differences in some parts to make them easier to print.

Servo arm was changed to its standard servo horn. To make it attach more securely.

It was later changed back in a new design, to allow for steeper angles.

Possible steering improvements

The current design has only one problem - a bit of backlash.

It remains, since steering joints are printed sideways, which in 3d printing means not a clean surface, and overhangs. That overall means bigger gaps, so it doesn't stick together.

It is done, so a precious wheel pin is as accurate as possible. So it doesn't break or have too much friction with LEGO wheel.

I think a possible fix is a double diamond connector, so instead of a wheel pin it has a diamond connector.

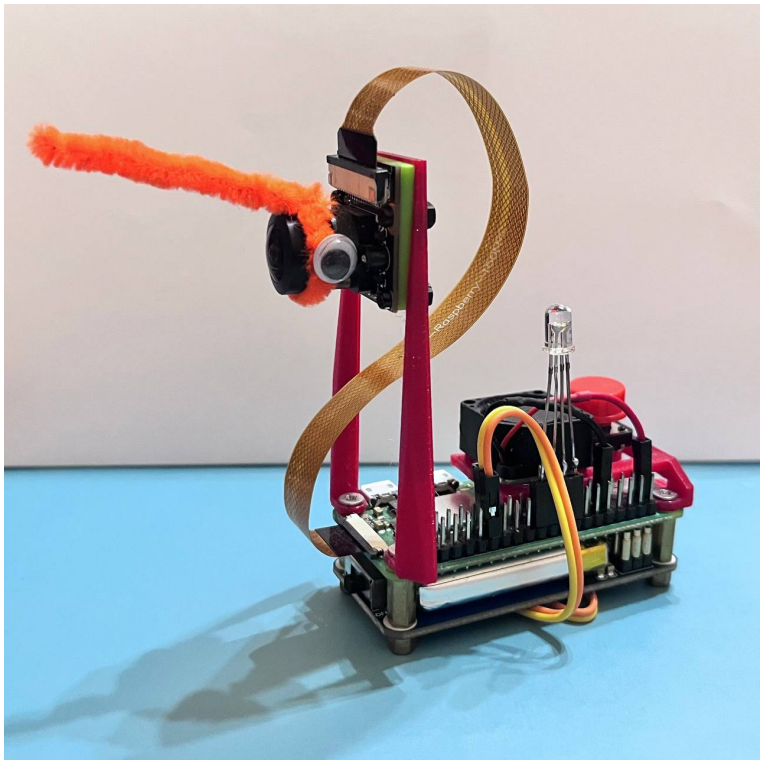
It theoretically allows to print all joints in good (vertical) orientation to be assembled together.

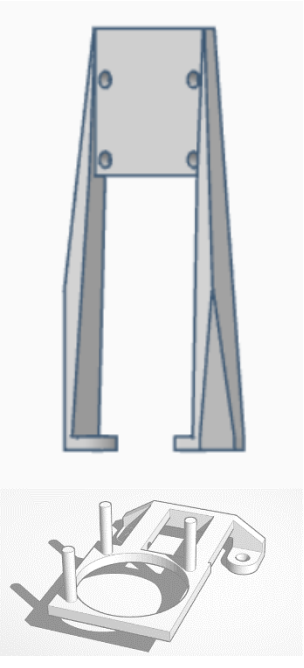
While that adds a bit of complexity, it removes backlash, that will likely result in a very good design.

So far I don't see any other issues from it.

The issue is small enough to be unnoticeable, but if a need for more precise control arises, this is where to go.

Electronics assembly





Electronics assembly is the assembly of main electronic components, which are closely grouped together.

This configuration simplifies the design, and reduces stress on their cables.

It consists of:

- Raspberry Pi Zero 2W
- Waveshare UPS HAT C
- Raspberry Pi camera module v1
- 2x2x1 cm cooling fan
- RGB LED
- Program start button
- Camera holder
- Cooling fan scaffolding

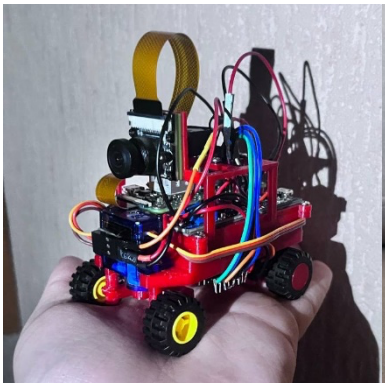
Raspberry Pi and UPS HAT are connected by design.

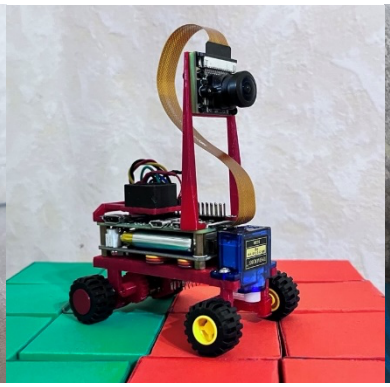
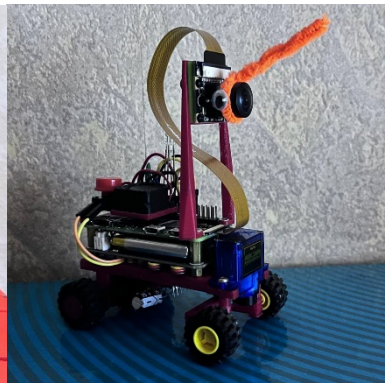
Cooling fan and camera need a 3d printed connector to attach. These connectors attach to the robot using Raspberry Pi screws.

Button is attached at the end of the robot, to the cooling fan connector. That is so robot doesn’t bump into fingers upon program starting.

RGB LED connects directly to 4 continuously positioned Raspberry Pi GPIO pins, that are of correct type and order.

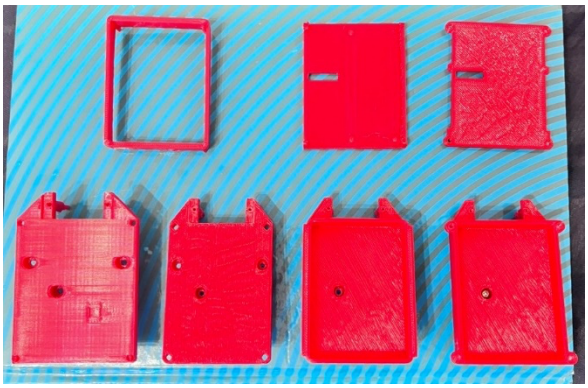
Robot development



Robot development had 4 major designs.

These are also closely followed in power management documentation.



DESIGN 1 had a custom UPS HAT. And it took longest to develop.

It is the most complex one of them all.

It consisted of two parts, the bottom plate (carriage and battery box).

And a lid (UPS HAT electronics, Raspberry Pi, camera and button)

The carriage was, as it is now, a plate that connects driving assembly and steering assembly.

The battery box, however, had 4 design iterations:

First Box and Second Box had a middle wall, for easier manufacture.

It was attached using a scaled diamond connector from Steering assembly. It didn't work because it didn't align well.

When middle wall was attached, there were gaps between it and the bottom plate.

The difference between the First and Second designs were steering designs (First had integrated pins).

The Third Box design used a lid connector, similar to food storage boxes.

This design merged middle wall into the bottom plate.

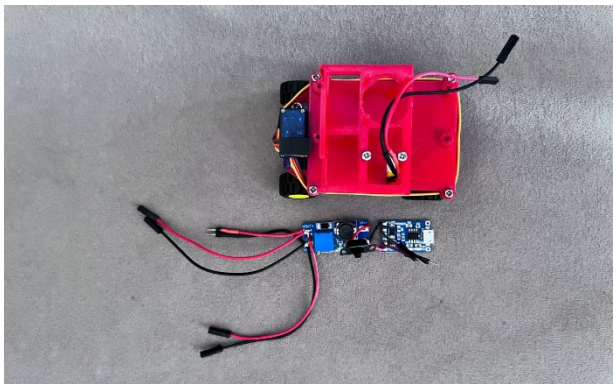
The design didn't work out; it wasn't connecting to the base.

Box design Four used screw connectors. It made a seamless fit.

Important thing to note, is that previous designs were made to be disassemble able by hand.

It is to allow fast battery-swapping. A feature of First robot and Second robot designs.

By using screws, it made the connection work, but now a screwdriver was needed to disassemble the robot.



The design of the lid had two iterations, that is due to voltage booster change.

It housed UPS electronics on the left, and Raspberry Pi on the right.

Lid has a scaffolding where camera, fan, and turn on switch are attached.

Unlike current design, here the motor and servo got power directly from the UPS HAT.

More on why this design failed is in Power management.

In a nutshell, due to a series of unlucky events, the Raspberry Pi didn't have enough current to boot up.

THE SECOND ROBOT DESIGN also had a custom UPS HAT.

It was more similar to the current designs though.

Consisting of a single base plate, where UPS HAT is attached.

The UPS HAT was greatly simplified, by using a battery with a built in recharge circuit.

That was the part where a lot of the cables came from, simplifying the design a lot.

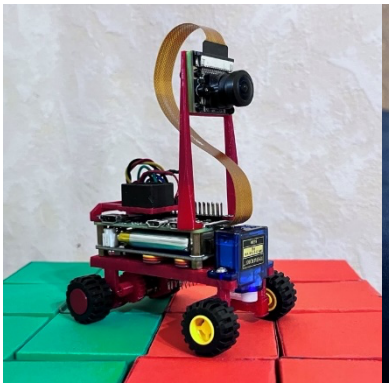
It was also more compact due to battery size.

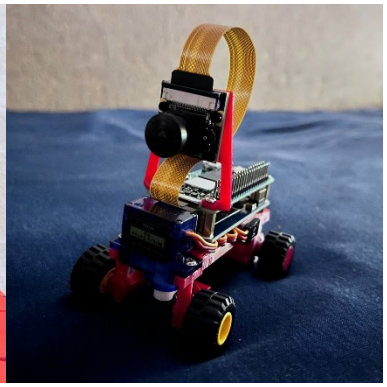
The Raspberry Pi had scaffolding on top to hold camera and fan.

This robot worked, and first programs were written on it.

The problems were that there was no battery charge level. And the overall built wasn't secure.

THIRD ROBOT DESIGN used a Waveshare UPS HAT C.





It made it even simpler, and more elegant.

Here the base plate is reduced to just connecting electronics, driving and steering assemblies together.

The Raspberry Pi CPU radiator was removed.

Camera was lifted from 7.5 to 10 cm.

More on that in the Sensor management.

This design didn't have issues, the difference from the Fourth (current) design is the electronics assembly.

Which is changed for better mission performance and user interaction, not design flaws.

It is also the original robot idea, before going into custom UPS HAT rabbit hole.

THE FOURTH (CURRENT) DESIGN is the more advanced model of the Third design.

The updates are:

- RGB LED light; program start/stop button.
- Motor to wheel ratio is lowered from 1:4 to 1:2.
- Motor driver receives two capacitors and a diode.
- Servo gets a new arm for bigger steering angles.

These changes are from two reasons:

User experience.

And brownouts during long driving, which Third design loved.

(A brownout is a condition of Raspberry Pi when, due to insufficient power supply it behaves weird/reboots.)

It also had cosmetics added.

They are for increasing robot length by 3 cm, while not interacting with the field. (due to their height)

Making parking easier.

But sacrificing the overall look, making the robot look like a toy.

Which wasn't the original design direction.

Robot base improvements

There isn’t much to do as this is a single parallelepiped connecting Electrical, Driving and Steering assemblies.

However, it can be improved.

In current design, if the robot runs into a wall, it can’t recover, a boundary can be made to fix this issue.

Robot base would need 2 screw holes. As seen in First and Second box designs from the First robot design.

Then an assembly that holds two ‘wall’ wheels would be attached.

It would allow it to drive on walls, and likely recover from bums.

I would use 11mm technic wheel, they are often used for this.