

# Power management

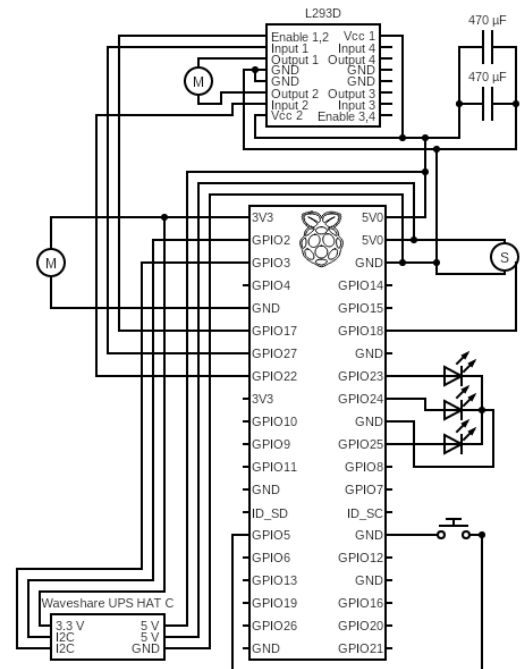
Robot electrical system consists of:

- Waveshare UPS HAT C
- Raspberry Pi Zero 2W
- Raspberry Pi camera module v1
- L293D motor driver chip
- N20 3-6 V 500 RPM motor (driving motor)
- Sg90 servo motor (steering motor)
- 2x2x1 cooling fan
- 5mm RGB LED
- Program start/stop button.

Waveshare **U**ninterruptable **P**ower **S**upply **H**ardware **A**ttached on **T**op takes care of battery management.

It provides power directly to Raspberry Pi.

Due to that, every other component gets power through Raspberry Pi pins.



Motor driver has 5 wires to connect: 5V vcc, ground, motor enable, input 1 and input 2.

Motor enable, input 1 and input 2 are connected to 3 continuously positioned GPIO pins to be bundled easier.

5 V vcc and ground are connected to specific GPIO pins, that UPS HAT uses to supply power to.

It removes the load from Raspberry Pi.

Servo has 3 cables: 5V, ground and PWM.

Like the motor, its 5V and ground are attached to special pins that UPS HAT uses. Also to remove load.

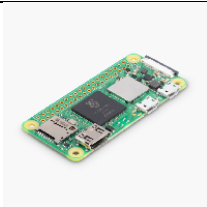

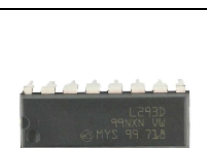



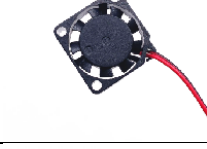

RGB LED is connected to 4 conveniently placed GPIO pins of the correct type in a correct order.

Allowing to connect it directly there.

5V Fan connects to 3.3 V to reduce power consumption, as it is more than powerful enough to cool hot CPU.

Camera is connected using a ribbon cable, that is made specifically for it.

Here is the table with images and descriptions of each element used in the electrical system of the robot:

Name	Task	Photo	Buy it online
Raspberry Pi Zero 2 W	Controls the robot		<a href="https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/">Official Raspberry Pi website</a>
Waveshare UPS HAT C	<i>"Uninterruptible power supply module specialized for Raspberry Pi Zero series"</i>		<a href="https://www.waveshare.com/ups-hat-c.htm">Official Waveshare website</a>
L293D chip	Motor driver		<a href="https://www.amazon.com/s?k=L293D">Amazon link</a>
N20 500RPM 6V DC motor	The driving motor		<a href="https://www.amazon.com/s?k=N20+500RPM+6V">Amazon link</a>
SG90 Servo motor	The steering servo motor		<a href="https://www.amazon.com/s?k=SG90">Amazon link</a>
2 x 2 x 1 cm cooling fan	Cools the CPU of Raspberry PI		<a href="https://www.amazon.com/s?k=2+x+2+x+1+cm+cooling+fan">Amazon link</a>
RGB LED light	Program visualization		<a href="https://www.amazon.com/s?k=RGB+LED">Amazon link</a>
12 x 12 mm Button Model ts-1112f	Start/stop the program		<a href="https://www.amazon.com/s?k=12+x+12+mm+Button">Amazon link</a>

470uF 25V Capacitor	Stabilize motor power draw		<a href="#">Amazon link</a>
------------------------	-------------------------------	--	-----------------------------

# UPS HAT development history

There were 3 electrical system designs.

Difference between them is the way they manage power.

Everything else, like the Raspberry Pi, camera, motors, etc. is identical between them.

First design.

It used a 3.3 V 1200 mAh lipo battery, and boosted it to 5 V needed for the system.

It used 2 circuits for this:

TP4056 is responsible for managing the battery's charge and discharge.

MT3608 is used as a voltage booster to increase the voltage from 3.3 V to 5 V.

This design had issues with MT3608 having a current limit of 2 A.

While initial calculations deemed this sufficient, the Raspberry Pi required a peak current of 2.5 A during startup.

That is 100 times more than when it is idle that was used in calculations, and thus exceeded the booster's capacity.

This rendered the design ineffective, as no suitable voltage booster was available that could operate at 3.3V and support currents over 2.5 A.

Second Design

The second approach took a different route, using a rechargeable 9 V PP3 type battery and stepping it down to 5 V

An upgrade from the last design was the use of an off-the-shelf battery, that allowed to simplify the design considerably by only needing a voltage regulator chip.

Since voltages were higher, it had a bigger current limit, so the power supply itself worked without issues.

However, this design had a critical drawback: it did not allow monitoring of the battery's charge level, making it challenging to manage the system effectively.

It wasn't cancelled immediately; the first programs were written on it.

Third current design

The final and currently implemented design leverages a pre-made battery management module. This off-the-shelf solution greatly simplified the design by integrating a custom PCB control board, which also provides the capability to monitor the battery's charge level.

# UPS HAT possible improvements

Development of a Custom UPS HAT was not a bad idea though.

The main problem of the current design is the small battery life, and a custom UPS HAT would increase it significantly.

Waveshare UPS HAT boosts a 3.7 V 1000 mAh lipo battery.

Assuming 95 % efficiency it has  $3.7 * 1 * 0.95 = 3,515$  Wh of energy

I think it is done to save weight and space, as I don't think cars were their original intention.

The second design used 9 V 1000 mAh battery.

Assuming 80 % efficiency it has  $9 * 1 * 0.8 = 7.2$  Wh of energy, twice as much.

However, the only realistic scenario where it would turn out better is with a custom PCB for it.

That would make it more compact and reasonable to use.

I would keep the use of a 9 V battery, but find a way to know its charge level.

Use of a custom UPS HAT would allow to have higher voltages to motor.

And even better – integrating the motor driver into PCB.

That would allow more consistent voltage on the Raspberry Pi since it would regulate it after motor power draw.

# Sense Management

The robot has two sensors installed. A camera, and a button.

Button is only for starting/stopping the program.

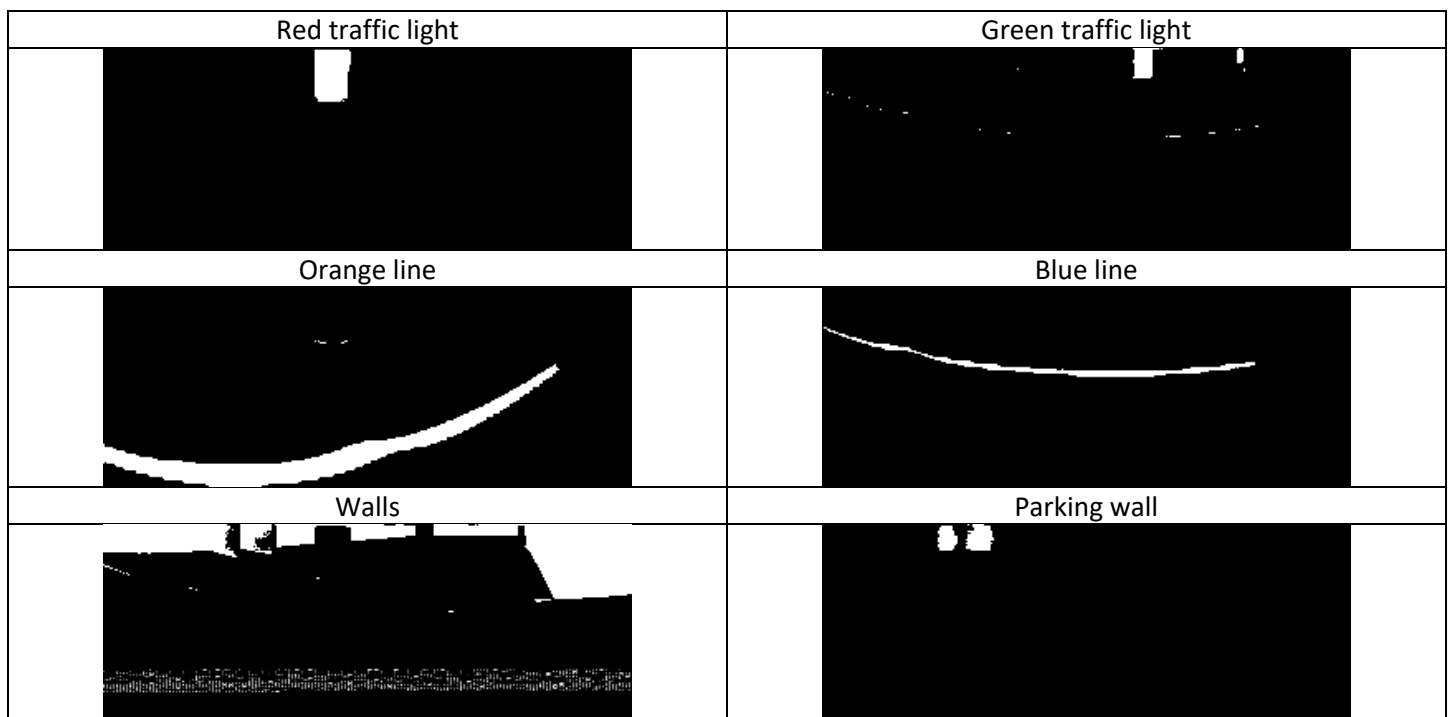
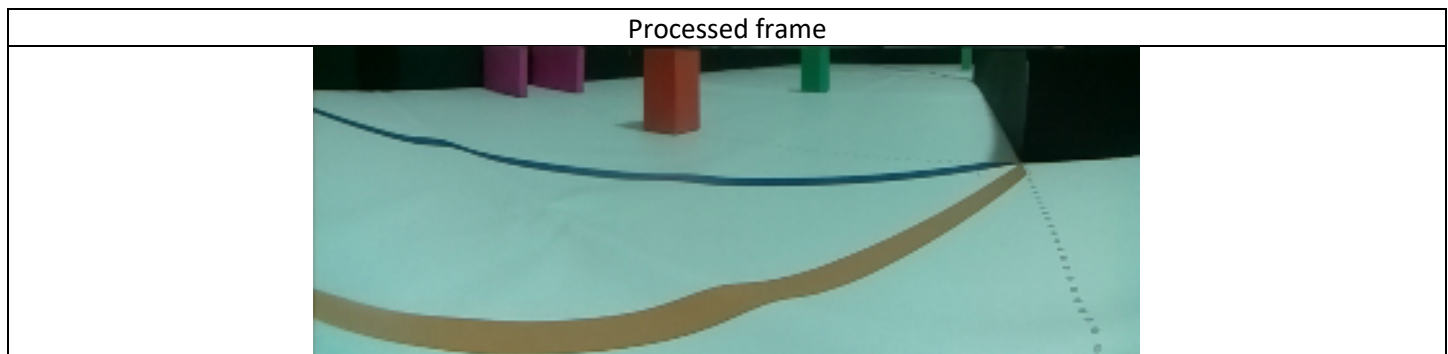
Using a single sensor simplifies program and strategy, while also making the design more compact.

Camera captures more than enough information due to the use of computer vision.

It uses masking technique to extract object data.

For objects, program estimates the distance using object hitbox area. its position on the screen and area are enough data to use further.

For lines and field wall, it simply counts the amount of pixels, and then uses that data to calculate further.



# Positioning the camera

When positioning the camera, you can't let it see outside the game field.

Otherwise it will confuse the program, by finding something that looks like an object program searches but actually isn't

For example, red/green t-shirts will look like traffic objects. And there is no way around it.

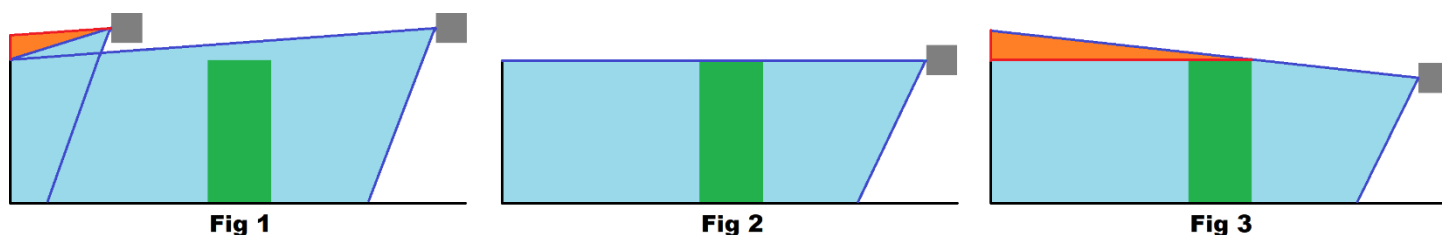
Also object needs to be fully visible to estimate the distance from it.

Since robot is moving, the only unique position parameters relative to the field are height and angle of the camera.

The angle is set to be 0 relative to the horizon, to simplify the program that cuts the input frame.

Height (from map to the center of camera lens) has to be as close as possible to the 10 cm.

The height of the objects and walls.



If higher (Fig. 1) it would need see the outside when going close the wall.

If its cut to be not see out when close to the wall, it would miss the objects when far away.

If it is lower (Fig. 3), then to see the object it also has to see outside of the wall.

If it is at exactly 10 cm (Fig. 2) then the object is fully visible, and it cannot see the outside.

# Camera possible improvements

Current camera problem is that it is slow, only 30 fps.

33.3 ms – it is 95% of the program time.

If the camera is replaced it would allow to run the program a lot faster.

And thus improve accuracy.



# Possible improvements

## Return to custom UPS HAT

As it was discussed in **UPS HAT development history**, custom UPS HAT will have a lot more energy.

However, to make it more reliable a **custom PCB** is needed, to save space and to remove dangling wires.

## Update camera to Raspberry PI camera module V2

- Higher FPS, 120 instead of 90. Camera FPS is the slowest part of cycle time.
- Better light adaptation, making object detection more reliable.