**Rover Project Report**

**“Steve”**

ENEL 387

Mason Lane

200376573

11/04/2021

# **Table of Contents**

[**Table of Contents**](#_b2l4t2nj24i6) **2**

[**Introduction**](#_pd7e81599ulh) **3**

[Summary](#_uuo73ltii03n) 3

[Milestones](#_pdn47nl5tkvj) 3

[**Development**](#_l2u8qertw63) **4**

[Current Build](#_gnnczcxdfzxj) 4

[Design Changes](#_bwu3sax43gsy) 4

[**Testing**](#_xxuvfm4behqf) **7**

[Process](#_fe80ee5olnc) 7

[Persistent Issues](#_vik0qznones9) 13

[**Results & Conclusions**](#_rw4s86y3qxd3) **15**

[**Appendix**](#_3o1ib5xctdzm) **16**

# **Introduction**

## **Summary**

This device will navigate a particular course & achieve certain milestones as it proceeds. It must navigate corridors as narrow as 15cm, and have a width of 13.5cm or less. This robot will detect various checkpoints & interpret logic from their codes. Upon reaching a dead end, the device will find its way back around, down the path and return to its starting position. Our robot must not be pre-programmed or use dead-reckoning navigation. It must also be capable of navigating objects that may impede its path. Our motors will incorporate PWM speed, two distinct sensors, accept inputs and provide various output.

This document outlines the successes and amendments made since the initial Rover Functional Specification document.

## **Milestones**

The robot must navigate a course as depicted in the Deliverable and Deadlines document. All navigation must be sensor based; no pre-programmed or dead-reckoning navigation. The navigation may use any combination of infrared, ultrasonic, or line following technology (OR any other appropriate sensor based navigation).

The vehicle will be placed at the Begin/End location. On command, the rover will move through the course until it reaches the Dead End. At that point, the rover will either turn around or reverse, and move through the course until it reaches the Begin/End location. On reaching the Begin/End location, the rover will signal that it has completed the course and then cease moving.

It must be capable of detecting an object or wall in its path and taking an avoidance action

It must be capable of reading a barcode target placed at various locations along the course. The patterns may be mounted on a horizontal or vertical surface of the course at the students discretion.

When the rover senses a target, it will signal the unique information provided by the number of bars.

# **Development**

## **Current Build**

My robot will operate on a 4-AA battery pack, powering two DC motors. A 9v battery will be used to power the Nucleo-64 - STM32F103 board. Our board will receive input from 8 sensors. Two sensors include mechanical bumpers, which we will use for obstacle avoidance. Next, a line sensor array (3 sensor outputs) will be used to stay on a general path. The line follower will be used for turning around at the dead-end. Third, two rear-mounted IR sensors to detect our dead-end. Once all 5 IR sensors have detected the dead-end, the vehicle will stop. Finally, a single side-mounted IR sensor (pointing upwards) will be used for target detection. These targets will be roofs of white strips. When the robot drives under a target, it counts each bar & outputs an LED in binary, using 4 LEDs for display purposes. Note, the line sensor is an analog output, and is therefore our analog sensor. All other sensors are digital. This device includes 1 input to start/stop the device on command (it cannot stop interrupts). As mentioned, it includes 4 LEDs to indicate (in binary) the value read from targets. Lastly, our robot uses a PWM motor controller for setting the speed and direction of the device. Its two motors may spin clockwise, counter-clockwise & perform short-stops. Low pwm values will stop the motors.

This is different from my proposed design, which I will go into detail in the next section

## **Design Changes**

* + 1. **Operational Changes**

There have been several changes to the overall operation as originally described in the functional specification document. Firstly, there is now a 9v battery pack. It was originally expected that 4AA batteries would be enough. This was not the case. Luckily, it was not hard to make room without impairing the overall structure/integrity of the robot’s design.

Next, I opted out of using a 7 segment display and have instead introduced 4 LEDs. This is to display various robot states as well as the target values in binary. It was deemed too complicated and demanding of space to implement with the smaller breadboard used.

Instead of two buttons, I use a single red button for start/stop functionality. Users must press the restart button followed by the red button to begin normal running. To disable the vehicle, a subsequent press of the red button will exit the loop and turn off the motors. This red button cannot escape interrupts & must be done while the robot is path following.

Lastly, there has been extensive changes to the target detection. Originally the design planned for two possibilities. First, the rear ir sensors for dead-end detection would also measure targets. This was difficult to work out logically in code. Next, vertical placement of bars was also infeasible. Here, the quality of bar detection was too low; if the robot was too far to one side - or too close - it might struggle to read data. Therefore, it was redesigned to read barcodes that are placed horizontally above it. An upwards-pointing IR sensor detects said bars, and measures a score to display in binary on the LEDs. in this case, pointing the IR upwards will guarantee a more reliable distance from the target. This is far easier than varying distances or angles. Furthermore, this keeps our target values from being confused with dead-ends or paths. Originally the robot would drive forward, detect a bar, reverse, and move forward re-counting bars as it proceeded. If no bar was detected after a given amount of time, it would consider its final count of bars as the score read. So, three bars = binary 3 in LEDs. Currently, the robot does not reverse - testing revealed that this was an unnecessary step in the barcode reading.

* + 1. **Schematic**

Due to various issues with ports, some inputs and outputs have changed in the schematic. This is the current iteration - including the added LEDs and reduction of buttons. See Appendix-6.2.

Several changes to the circuit have occurred, mainly to accommodate the inclusion of LEDs and exclusion of a button. Furthermore, the addition of the 9v battery bank had to also modify the circuit. Currently every device but the motors is powered by the 9v bank through the Nucleo64. Several devices do not go into the same ports; this was due to some confusion with permitted ports. These changes can be primarily noticed for the motor controller. This device is no longer using PB4, PB10 & is instead using PB1 & PB7 for PWMB and AIN1. AIN2 has been moved to PB5. LEDs are now using PA7, PB2, PB10 & PC7. Lastly, the start/stop button uses PC7. This should summarize the changes made to the proposed circuit.

* + 1. **Physical Build**

This device mainly follows the previously outlined build barring some changes to accommodate the IR sensor, breadboard orientation & upwards IR. In this iteration, the breadboard and nucleo64 do not sit side-by-side. Instead, the breadboard sits behind the nucleo. This was better use of space. Both the breadboard and nucleo have been attached to an acrylic platform, which has in turn been attached to the robot’s round upper-level. The rectangular acrylic is slightly larger than the robot’s round platforms. However, the 3 layers have still allowed the device to benefit from an incredibly small/compact form factor.

At the bottom of the chassis, standoffs have been adjusted to accommodate a 9v battery. This has not impaired the structure - though the 9v casing does disperse some weight. Despite this, the chassis is more than capable of supporting itself without the 9v battery. This chassis also incorporates the motors in it’s design as part of the standoffs for the second layer.

Next, the position of the 4AA power bank has not changed. However, rubber bumpers have been added in conjunction with adhesive tape to install a rail system for the AA bank. Operators may remove the power bank to replace batteries without having to remove the entire housing from its cradle. This has resulted in a far sturdier and effective solution for battery replacement. Similar was done for the 9v battery pack. While there is no rail system for the 9v pack, rubber bumpers hold it in place while still enabling it to be easily removed. Slack has been added to the wires of each battery pack to allow for the rail system & changing of batteries. There is a build video for a more visual description of this.

Finally, the side-mounted IR now points upward. A metal frame is used to point said sensor upward. This frame also serves as a rotatable cable shroud, which hides some of the cable management. It may be turned to reveal wires that run under the nucleo64 and into ports on the other side. Much has been done to hide the number of wires used. Barring wires used for the LED and power buttons (which would be soldered given a VLDiscovery Carrier board), few are easily visible & make attempts to leverage the free space in lower levels. Cable Ties are also used for this organization.

Motors, line sensor & rear IR sensors are all mounted in their proposed positions. Cables are run throughout the chassis to keep as much hidden as possible.

For a schematic of the robot’s updated build - see Appendix-6.3. Likewise, see Appendix-6.5 for images of the actual device.

# **Testing**

## **Process**

* + 1. **Chassis Assembly**

I started with the chassis & motor assembly. I ensured each component would fit adequately in their expected spots. Making room for the 4AA battery, I also ensured the chassis could still support its weight with standoffs moved to different positions; it can. I then added my sensors, and acrylic platform, breadboard and nucleo. I opted to test parts \*after\* assembly - instead of one at a time via ADALM, to avoid a mess of cables. This was a risk in the event of defective parts. Luckily, no parts were defective and it ultimately enabled me to create a more organized device.

* + 1. **Power**

As stated in the design changes, 4AA batteries were not enough to power the nucleo64 and had to be replaced with a 9v battery. The 4AA batteries still power the DC motors via PWM controller. This was discovered during initial testing with the nucleo64 & battery. While it did turn on, the device was incapable of outputting the voltage needed. This problem was then narrowed down using the ADALM and Nucelo64 documentation. Its power requirements were previously confused and it now appeared that 9v would be required. Luckily, it was not hard to attain a 9v battery pack.

Using the 9v battery pack fulfilled the board’s power requirements, but enabled a lapse that went unnoticed for some time. It was initially believed that 3-5v powered sensors would, regardless, output 3v. This was not the case & all sensors had been initially powered from the 5v pin of the Nucleo64. While this board does possess some 5v tolerant ports, they were not the ones being used. This would likely damage the Nucleo had it not been resolved. At a 5V sensor output, the nucleo64 still functioned and handled said outputs. All seemed well, and there were no signs of damage during light testing. However, an unknown component gave off an unsettling smell if run for too long. During longer tests, this became evident. To remedy the concerning issue, attempts were made to pinpoint the source of the smell. Three possibilities existed: the line sensor, which ran hot, the motor controller or the board itself. While hot, the nucleo’s chip did not overheat, burn or exhaust smoke. Next, the ADALM was used to measure voltages and compared to datasheets for the sensors. Meanwhile, the same was done for the motor controller. Each component was rated for 5v and was correctly wired regarding their Vcc, GND and outputs. Next, sensor outputs were measured and compared to the Nucleo64’s port tolerance values in its documentation. No port was 5v tolerant, but was receiving such from the sensors. To fix this, the device now only uses the 3v3 pin to power its components. Luckily, all sensors & the motor controller are functional at 3v3. No unsettling smells or behavior has been observed since making this amendment. To comment on this, I would like to add how lucky I am to have discovered this problem when I did. I am unsure how long the device could have remained stable before causing catastrophic damage. Currently, there have been no lasting effects.

* + 1. **Motors & Controller**

In this case, two motors were already expected before the PWM lab. As a result, said lab was extended to prepare the necessary ports for the PWM motor controller. Doing so made it incredibly easy to fully implement the motor/controller solution. It was not necessary to do this in the lab, but made preparing the system much easier. There was even a setMotor function prepared for setting clockwise/counter-clockwise. Testing was also a trifle because of this. First, the pwm outputs were confirmed - just to be safe. Next, the circuit was compared against a tutorial - which was correct. Using the motor controller’s datasheets, the setMotor function was modified to send the correct combination of 1 and 0 values; 01 is counter-clockwise, 10 is clockwise. After doing this, the motor driver was then fully tested to ensure it moved/rotated as specified. For instance, setting both motors forward at pwm = 75 should move the robot forward at 75% of its max speed.

* + 1. **Bumpers**

Bumpers are digital sensors which send a signal when their circuit is completed by the metal bumper. It is essentially a very long switch. These bumpers would be used for obstacle avoidance. Testing began with first confirming via the ADALM what kind of signals were being sent. When was high, when was low. From there, a falling trigger interrupt was created. This interrupt would reverse for some time, and rotate opposite to the direction previously making contact with a bumper. How far a vehicle reversed or turned was tuned given subsequent tests. It was then confirmed that the robot could effectively avoid obstacles in a square enclosed area. The inner wall off the main course is removable for the purposes of these tests. In observation, the robot was capable of relying on its bumpers to navigate the 15cm passage if the line follower failed. In testing, the bumpers occasionally get stuck & cannot be pushed far enough to trigger. This is adjustable based on how close the bumper is from the contact point. If there is too much space, there is not enough force to push the bumper far enough.

Lastly, a check within the interrupt was used to guarantee it was being done only on falling trigger situations. For some reason, a falling trigger would call the interrupt without actually having IDR6 or IDR5.

A final bug has been discovered that causes serious issues due to the incorrect clearing of PR bits. A solution has not been resolved. In the correct circumstances, the bumpers may be triggered to break out of a loop and move forward infinitely (unless an interrupt occurs). As a result, this bug causes the robot to ignore all logic in main and therefore does not consider its line follower or rear IR state. A resolution is still being worked on at the time of writing this document. This is primarily why three trial videos have been posted - one with bumpers active and another without. This bug was originally undiscovered, as the target detection interrupt has its own line follower built in. As a result, it first appeared to be an issue with the line follower. Further examination realised that the line follower was still functional during interrupts - but was not in main - after the bumpers triggered their bug.

* + 1. **Line Follower**

Likewise in previous tests, the ADLM was used to measure analog values from the analog sensor. From here, functions were derived to calculate the ADC value. Using the ADC value in conjunction with equations derived from previous labs, it was then determined what threshold equated to a black line. It was then confirmed with black lines. Initially, the threshold at 2700 occasionally struggled to differentiate certain materials. After raising the value, very little confusion was noticed without harming the detection of the lines. Since black indicated high on the line sensor, it was determined that any value greater than 2925 was a black line.

After these tests, the array was mounted below the chassis. Unfortunately, its initial metal mount was too high for the sensors to function. After adjustment, it still occasionally missed lines. At first it was considered a threshold issue. However, upon further analysis it seemed the array was still not low enough. One final adjustment allowed the device to accurately detect lines.

Next, logic was coded for handling the robot’s response to a given line encounter. When the middle sensor detects black, it moves forward. If the middle is not triggered, and the left or right sensors are - the robot turns in the direction of the sensor until the middle is once again aligned with the black line.

Now, this did not immediately function flawlessly. First, the robot struggled to re-align itself if it ever lost its black line (or the bumper interrupt displaced it). To resolve this, several contingencies were written to enable attempts at re-acquiring the line. Originally, the robot would jiggle if both the left and right IR were triggered. This would occur if the robot drove into the line perpendicularly. To do this, speed biases were applied to the turn rates. In this case, the left and right triggers rotate the robot at different speeds. This was to help the robot shake itself out of its infinite loop. It was surmised that if each if statement moved the vehicle at different speeds, it might escape certain situations. Testing confirmed this, although it was not always an effective solution. This contingency is seen as a last resort.

A final addition attempted to further address instances where the robot might find both its left/right IR sensors triggered. In these cases, the robot will turn left with a bias towards one wheel’s axis - this guarantees that it will eventually find itself re-aligned. This works by rotating the robot, but while also running either motor at different speeds. By halving the speed of our right motor, we can turn out of a situation where the left and right are triggered while simultaneously setting up the device to be triggered on the right IR sensor. While this solution was effective for escaping infinite loops, there is the risk of it mistakenly turning itself around. This is due to the fact that it rotates in a predetermined direction.

While these have not been implemented, further logic could be added to consider which IR sensor is detected first. For instance; if the robot approaches at a slight angle from being perpendicular, it would await the detection of the middle IR (note that it is slightly ahead of its side sensors). After this condition, it would remember which IR sensors are triggered next. When the middle IR leaves the black line, leaving only the left and right triggered, the device can determine which IR entered the line first and turn based on that. This would enable the robot to turn into the direction it was originally going instead of a predetermined rotation.

* + 1. **Deadend**

Initially it was planned for the bumpers to eventually guide the robot around. This was to showcase the bumpers. Unfortunately, there is not enough room for the robot to rotate this way and it often got stuck. Instead, several versions of line configuration were experimented with to discover an effective way of coaxing the robot around using the line follower. Initially, the line turned at a right angle into the wall, which then allowed the robot to rotate with its bumpers. In this case, the robot sometimes fell into infinite loops - hitting the wall, reversing & then re-aligning with the line only to hit the same position. Currently, the line is configured to simply turn the robot around a tight circle. Problems experienced here is that the robot sometimes has a hard time re-aligning with the line after turning around. This causes the aforementioned jiggle bug. Sometimes it works itself out, other times it cannot and either gets into an infinite loop or hits a wall; falling into a loop given the small room for maneuvers.

* + 1. **Start/Stop Square**

To handle the Star/Stop square, it was important that the device does not confuse the area with black lines. This has proven to be a complicated issue to solve. To handle this case, the rear IR sensors are used in conjunction with the line follower. On startup, the robot moves forward until the rear IR sensors do not detect black. At this point, the robot knows that it has cleared the square and may begin normal running. It finds the line and continues the course. To stop, it must re-detect black on every down-facing IR sensor. This includes the line follower array and read IR sensors. This has been done to ensure the robot does not stop on corners. By creating a triangle, the robot can guarantee very little chance of all IR sensors detecting black - that is, unless it's on the square. In testing, the robot occasionally fails to react with the stop point. This is caused by the bumpers after they break the loop in main.

* + 1. **Target Counter**

Testing began with a counter - LEDs would light up in binary based on how many times a white bar was detected. Following this, would be the logic for handling different states and the implementation as an interrupt. So, the logic for entering a target and knowing when it has exited a target. Finally, testing was done to determine the type of trigger; rising or falling edge. It was determined to be falling edge. However, the interrupt may still trigger despite it not being a falling edge. To avoid this, the interrupt uses a while loop that the robot cannot enter until the IRD has confirmed that IDR10 is indeed low. If it is not low, the robot exits the trigger. This means that the robot is likely triggering the interrupt many times - but does nothing as it includes the second check.

The target counter initially planned to mount on the side. This proved difficult, as varying distances from a wall impacted the reliability of the bar detection. These revelations concluded that the targets must be placed horizontally. However, to avoid confusion with the line sensor and rear IR sensors it was decided that the targets would be mounted overhead. This allowed the robot to maintain a consistent distance from the target without interfering with other components. Since no detection, empty space or “void” is considered black/non reflective, we may consider the entire overhead black. This way when the robot detects white bars, it knows it is reading a target. As the robot moves under a target, it counts how many white bars exist. If no white bars are found in ~1-2 seconds, it knows it has cleared the target. Its final count is the score & is then displayed in binary via 4 LEDs. Initially, the robot reversed upon finding its first bar - this was to ensure all bars were properly read. However in further tests, it was determined that this feature was redundant. Now, the robot simply drives under the target - it does not reverse & still counts the scores. Another hurdle came from the robot’s attempt to follow lines & count scores. Sometimes, the robot would count a target, then turn as a result of the left/right IR being triggered. When this happened, the robot occasionally rotated back into the detected white bar - which would then incorrectly count a larger score. So for instance, three would be measured as four. To remedy this, a millisecond timer was placed within the function. This created a short buffer between black and white lines. Doing so would prevent the robot from detecting the same bar twice. A misscount may still occur, but in far fewer trials.

## **Persistent Issues**

* + 1. **Target Counter**

On occasion, the robot moves too fast to detect bars. Latest iterations have not had this issue however. Furthermore, the robot sometimes detects the same bar twice despite the buffer. This may be resolved by increasing the size of the buffer. However, it's important to be mindful of the minimal time expected before reaching another bar. In this case, the robot must not fail to detect bars - but also be strict enough to deny duplicates. Lastly, the robot may be taken out of the interrupt if its bumpers are triggered. In this case, avoiding an obstacle is more important than reading the target; it cannot restart on a target and may read an incorrect target if this occurs half way between reading. If the robot loses its line & mistakenly turns itself around, it may also confuse the score of a given target.

As mentioned earlier, the robot’s interrupt requires an internal check to avoid false positives of falling triggers. The problem remains un resolved & extends to other interrupts like the bumpers as well.

* + 1. **Start/Stop Square**

At times, the line follower fails to stop over the end square; this is due to the bumper interrupt. Some contingencies have been added to all interrupts, aiming to encourage the robot to stop if all conditions are met regardless of being in main or an interrupt. Lastly, the device does not do more to indicate its conclusion other than stopping. This could be improved with a simple LED notification.

* + 1. **Deadend**

Currently the robot has a hard time turning back around. The aforementioned contingencies have been made to the IR line detector - yet it still sometimes gets stuck in its attempt to re-acquire the line. In this case, the line is in the shape of a hood - which aims to send the robot around the hook to release the robot towards the direction of the line. It should then re-align itself in the correct direction. As stated, it sometimes fails to do this.

* + 1. **Line Follower**

Recent versions of the line follower have increased its reliability regarding its ability to stay on a line and re-acquire lines. However, it still has jiggle issues that it cannot always escape from when trying to reacquire a line. Some ideas were envisioned but could not be implemented before the project’s completion date.

* + 1. **Bumpers**

As stated earlier, the bumpers require an internal check to ensure the interrupt is not being incorrectly triggered. This problem is unresolved but the solution has made it unnoticeable in tests. This is more a persisting issue in code - as it should be a redundancy if the interrupt trigger was working correctly. In other words, the solution is not the ideal solution and should not be permanent.

Bumpers also fail to trigger at times, due to the lack of force needed to move the metal bumpers. At times, the bumpers have too much normal force and the robot lacks enough force to push it far enough to trigger. In this case, the bumpers on rare occasions are stronger than the robot’s torque. Unfortunately this is a consequence of the slower speed chosen. A slow speed was preferred to better accommodate the line follower and target detector.

Lastly, bumpers have a high chance of breaking out of the main function & causing serious issues in the performance of the robot. This is clearly an issue with the clearing of PR bits. That, or the device has sustained damage from the initial 5v inputs which has caused unexpected behavior.

# **Results & Conclusions**

Ultimately the robot can navigate the course and accurately count targets on tests. Target detection and line following are not the predominant cause of failure. Currently, the greatest hurdle is the vehicle's bumper interrupt. Initial tests did not mirror the bugs encountered at the time of writing - and further investigation is required. As such, there are three recordings to document the devic’s bug. In testing, the robot handles the dead-end occasionally; unfortunately, the bumpers have a high chance of breaking the robot’s ability to complete the course. This is indicative of the need to improve that point, as it is the main cause of failure during start-to-finish trials that make it impossible to submit a fully functional video.

# **Appendix**

* 1. **Trial Videos**

[Robot Demo](https://drive.google.com/file/d/1FiWXX9EpEAgdSjajlTSXWPrzNJ3YERhg/view?usp=sharing)

[Stopping at Black Square](https://drive.google.com/file/d/1FuOrFnKXbfIPUBzg4u0UP4o1J4Xb0fmQ/view?usp=sharing)

[Replicating the Interrupt Bug](https://drive.google.com/file/d/1G1rM8EpQHXRfAGP2ID7Jpls6TfBXUFU8/view?usp=sharing)

* *This causes many failures including the inability to stop at the ending*
  1. **Build Videos**

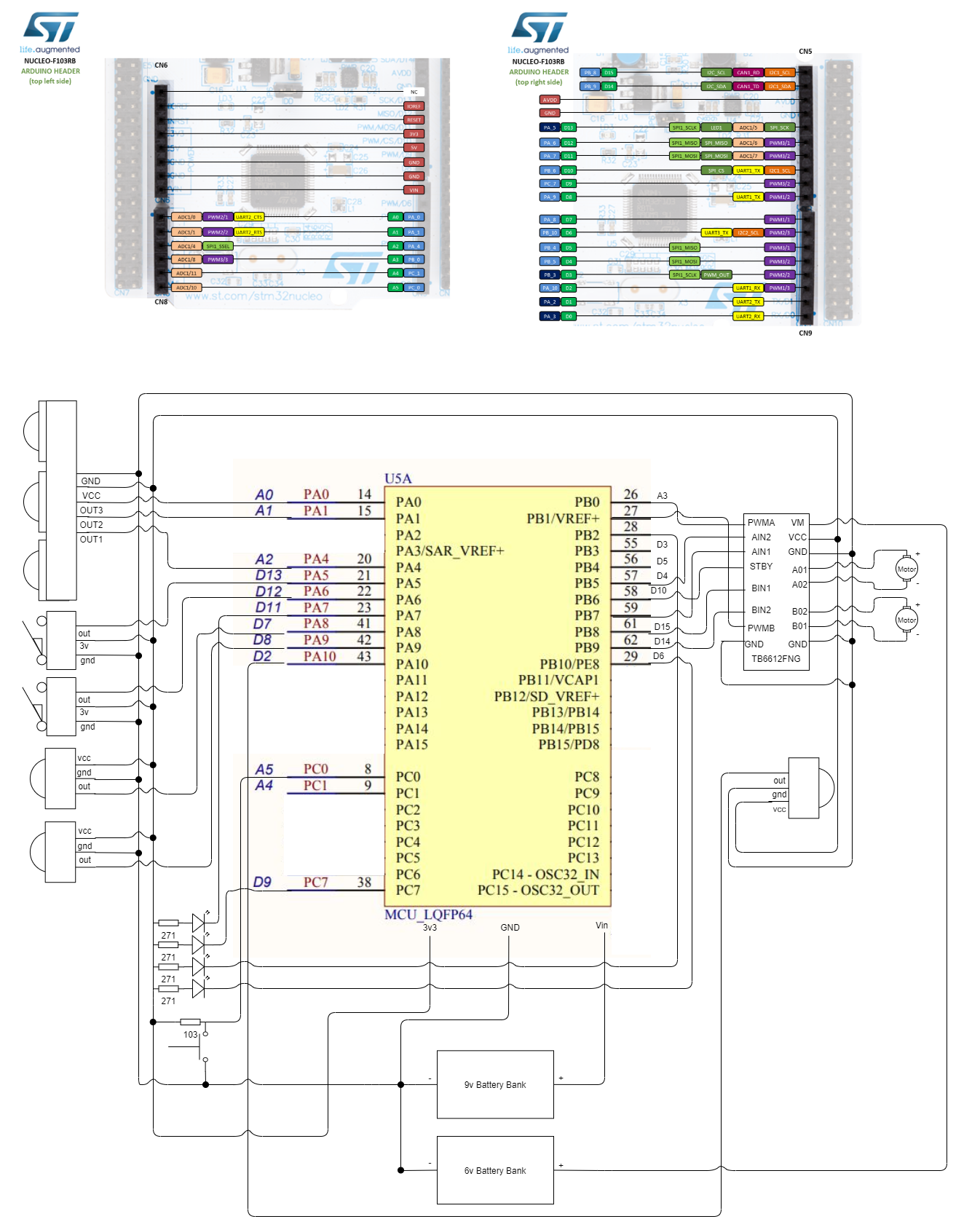
[Robot & Course Summary](https://drive.google.com/file/d/1GCuME3loFSES1GaVxIXGcZQCMY0pC-j1/view?usp=sharing)

[Robot LED Info](https://drive.google.com/file/d/1GM2V0iXogPVOkOU8fdnFzGP-_SqWbiuc/view?usp=sharing)

[Course Info](https://drive.google.com/file/d/1GCv3meURnIgSTgVPSo7GCq_2bEm3YYmo/view?usp=sharing)

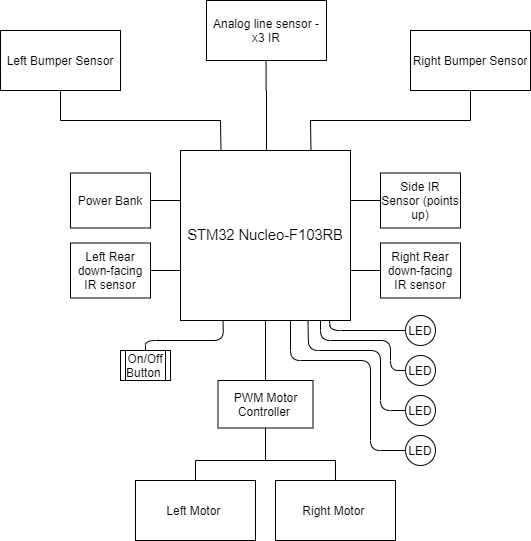
[Battery Info](https://drive.google.com/file/d/1GGNtO7iCc_q9j6Ktqifdj8JxEw-L3pt7/view?usp=sharing)

* 1. **Updated Schematic**

****

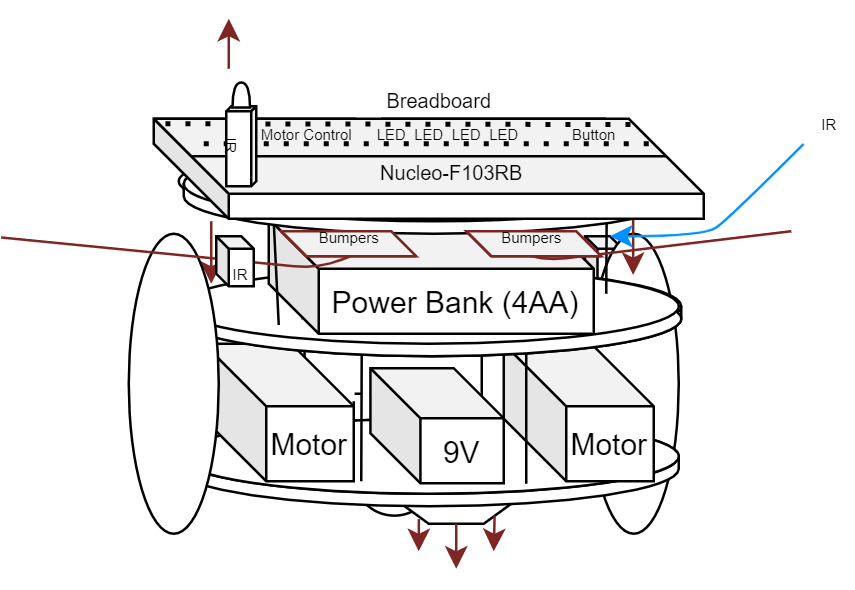
A revised version of the circuit used

* 1. **Updated Diagram**

****

A revised version of the proposed schematic in summary

* 1. **Updated Robot Design**

****

A revised diagram of the envisioned build

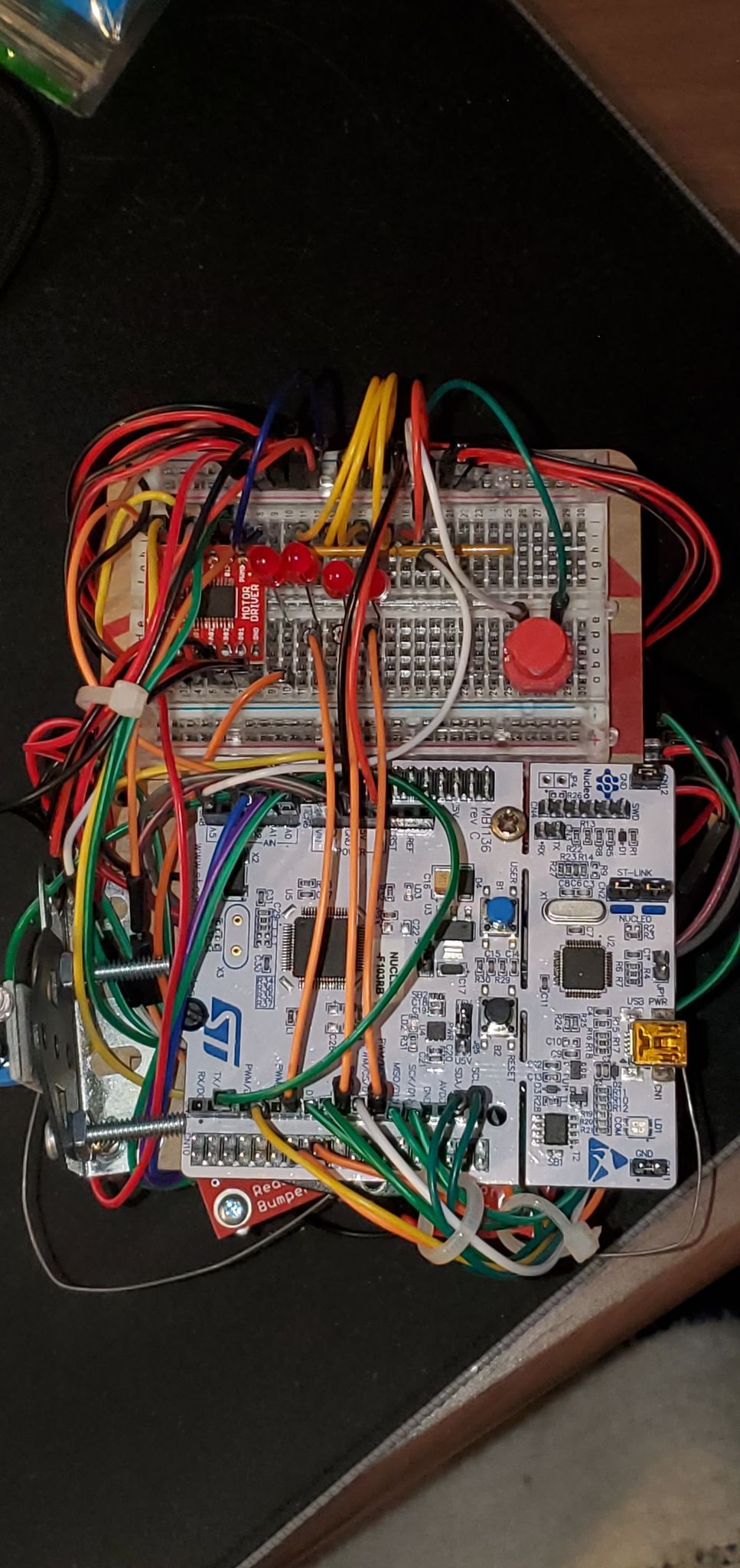
* 1. **Parts & Datasheets**

List of parts used & their datasheets

| **Oiyagai 5pcs IR Infrared Sensor** | Detection range of 2-30cm.  Directly connect to microcontroller  3-5v DC power modules. Red power indicator  3 pins. Vcc, gnd, out | No formal datasheet given.  See [link](https://www.amazon.ca/gp/product/B0776RCLH6/ref=ox_sc_act_title_1?smid=AVKVKSQRPUCTR&psc=1)  [Generic Datasheet](https://components101.com/admin/sites/default/files/component_datasheet/Datasheet%20of%20IR%20%20Sensor.pdf) |
| --- | --- | --- |
| **VIPMOON 40pin Male, 40pin Female, 40pin Male to Female** | 8 inch solderless breadboard jumper wires  Males insert into standard 0.1"(2.54mm) female sockets  Females for insertion onto standard 0.1"(2.54mm) male headers | No formal datasheet given.  See [link](https://www.amazon.ca/gp/product/B01M1IEUAF/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1) |
| **Plastic mounting plate for breadboard and Arduino** | 3mm thick plastic plate with engravings  2 x 4-40 3/8" machine screws  4 x 4-40 hex nuts  4 x rubber bumpers  93.9mm x 114.2mm x 3mm | No formal datasheet given.  See [link](https://www.adafruit.com/product/275) |
| **Mini 3-Layer Round Robot Chassis Kit - 2WD with DC Motors** | Length of Metal Plates: 100mm / 3.9"  Width of Car: 109mm / 4.3"  Height: 66mm / 2.6"  2x Drive Motors (drive with 3-6VDC, 200-400 mA run, 1.5A hard stall)  2x Wheels  1x Plastic Caster Ball  Anodized aluminum frames and all mounting hardware for assembly | No formal datasheet given.  See [link](https://www.adafruit.com/product/3244) |
| **2x DC Gearbox Motor** | Rated Voltage: 3~6V  Continuous No-Load Current: 150mA  Min. Operating Speed (3V): 90+/- RPM  Min. Operating Speed (6V): 200+/- RPM  Stall Torque (3V): 0.4kg.cm  Stall Torque (6V): 0.8kg.cm  Gear Ratio: 1:48  Body Dimensions: 70 x 22 x 18mm  Wires Length: 200mm & 28 AWG  Weight: 30.6g | [Diagram](https://cdn-shop.adafruit.com/product-files/3777/3777_diagram.jpg)  [Datasheet](https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf) |
| **4xAA Battery Pack** | Box Dimensions: 2.5 x 2.75 x 0.75  Cable: 6" long  Weight: 34.9g  nominal output of 6V DC for alkaline (6.4V when fresh, 4V when dead)  4.8V DC for rechargeables (5.2V when fully charged, 4.4V when discharged). | [Datasheet](https://cdn-shop.adafruit.com/datasheets/EPD-200659.pdf) |
| **DFRobotShop Rover Line Follower Sensor** | 3 miniature reflectance sensors  Interface type : 0-5V Analog x3  Operating voltage : 5V (Nominal)  Supply current : 75 mA  Sensing range : 3mm to ~6mm  Optimal sensing distance: 3mm (0.125" )  Optimal line thickness: black electric tape is ideal (≈17mm) | [Datasheet](https://www.robotshop.com/media/files/pdf2/linefollowerdatasheet.pdf) |
| **2x RedBot Sensor Mechanical Bumper** | 1x Mechanical Bumper Board  1x Whisker  1x 3/4" 4-40 Nylon Standoff  1x 4-40 Hex Nut  3x 3/8" 4-40 Phillips Machine Screw  1.03 x 0.69" (26.27 x 17.67 mm) | [Datasheet](https://media.digikey.com/pdf/Data%20Sheets/Sparkfun%20PDFs/SEN-11999_Web.pdf)  [Schematic](https://cdn.sparkfun.com/datasheets/Sensors/Proximity/RedBot%20Whisker%20Bumper.pdf)  [Github](https://github.com/sparkfun/RedBot_Whisker_Bumper) |
| **Motor Driver - Dual TB6612FNG** | Power supply voltage: VM = 15V max  VCC = 2.7--5.5V  Output current: Iout = 1.2A (average) / 3.2A (peak)  CW/CCW/short-brake/stop modes | [Datasheet](http://www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf)  [Schematic](https://cdn.sparkfun.com/assets/b/0/f/7/4/SparkFun_Motor_Driver-TB6612FNG_v11c.pdf)  [GitHub](https://github.com/sparkfun/Motor_Driver-Dual_TB6612FNG/tree/master) |
| **Breadboard - Translucent Self-Adhesive** | 2 power buses, 30 columns, 10 rows  400 tie in points.  Pins spaced by a standard 0.1".  Two sets of five rows are separated by about 0.3"  Wire sizes in the range of 29-20AWG.  3.29 x 2.15 x 0.33" (83.5 x 54.5 x 8.5mm) | No formal datasheet given.  See [link](https://www.sparkfun.com/products/9567) |

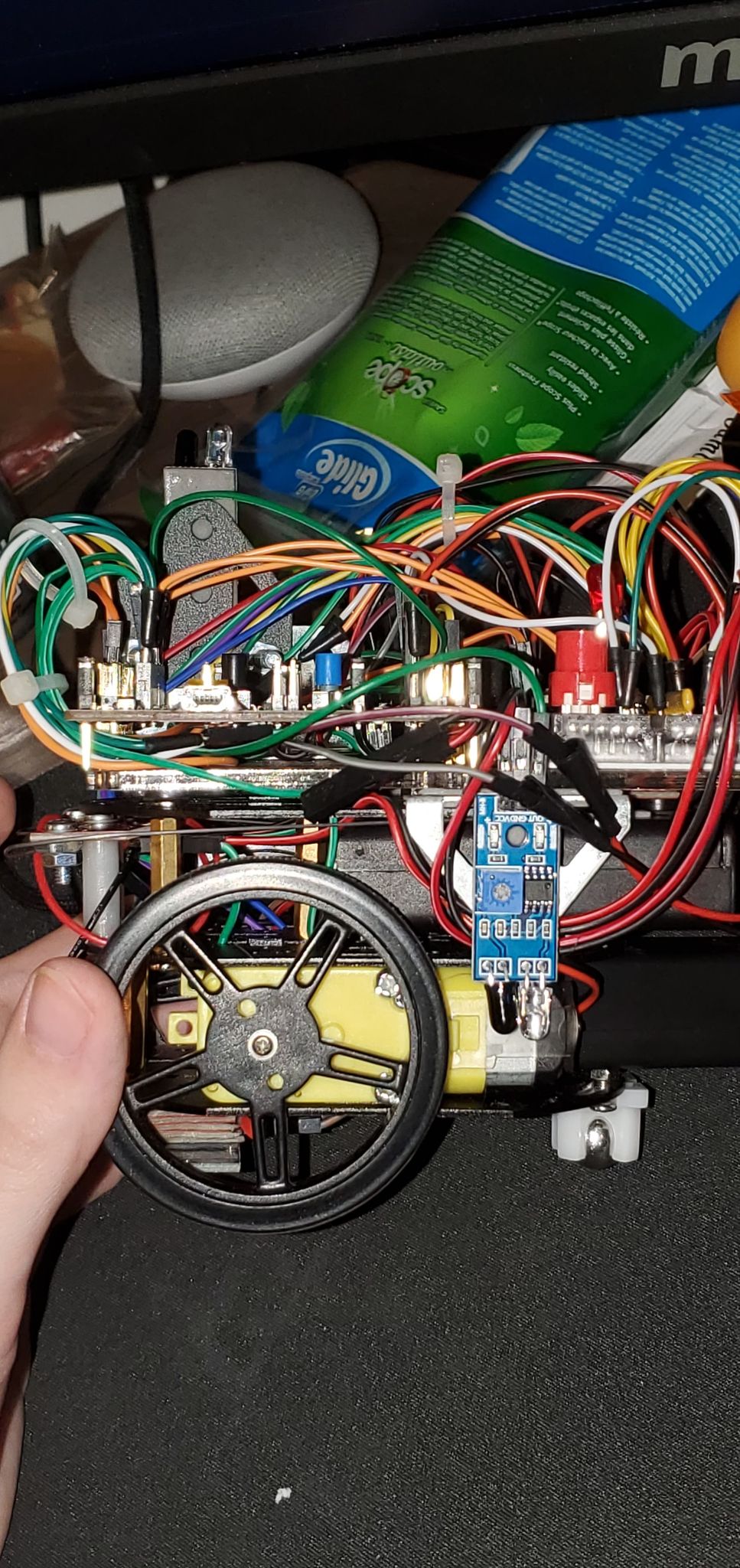
# 

* 1. **Device Photos**
     1. **Top-Down & User Interface**



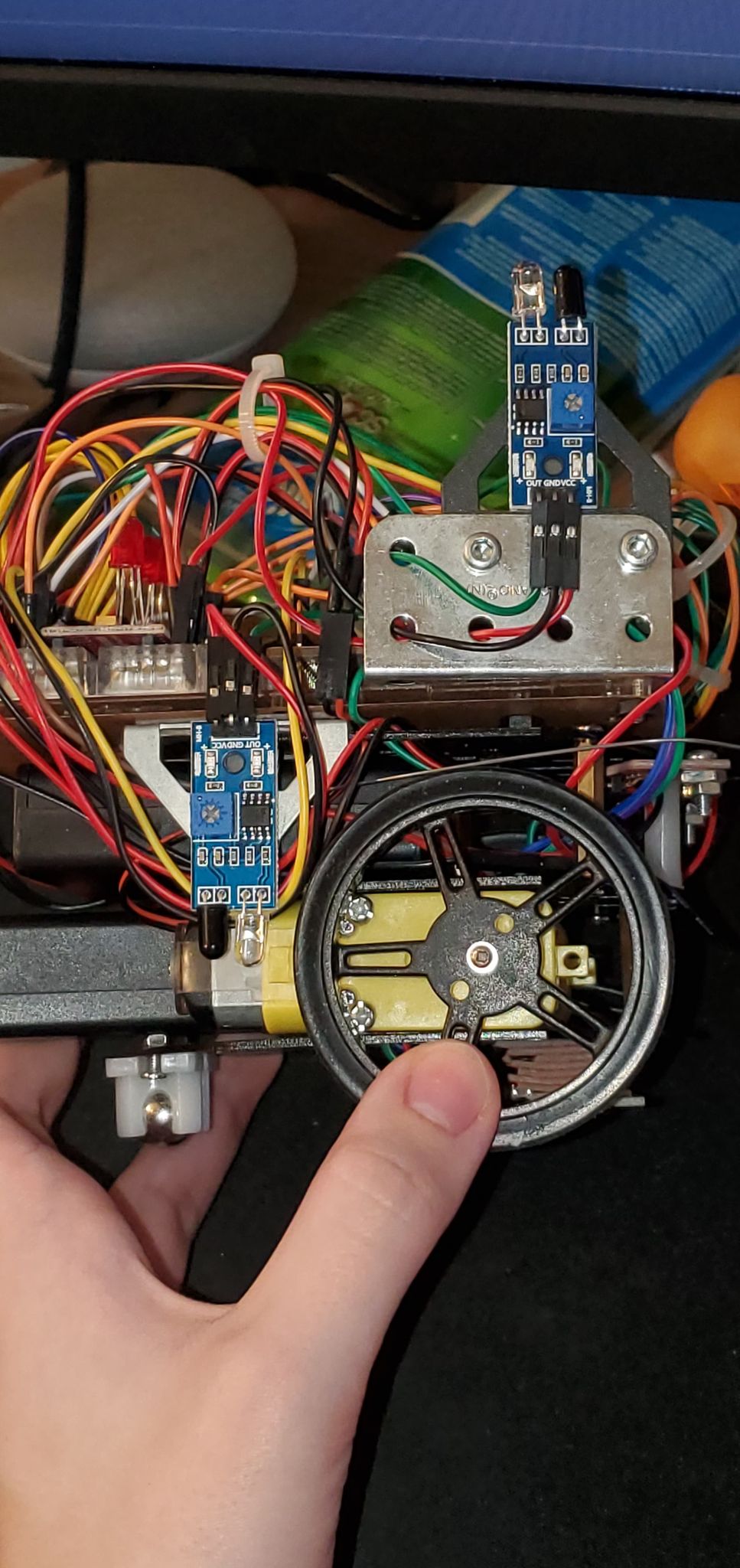
This image shows the device from the top-down, including its revised UI

* + 1. **Left IR**



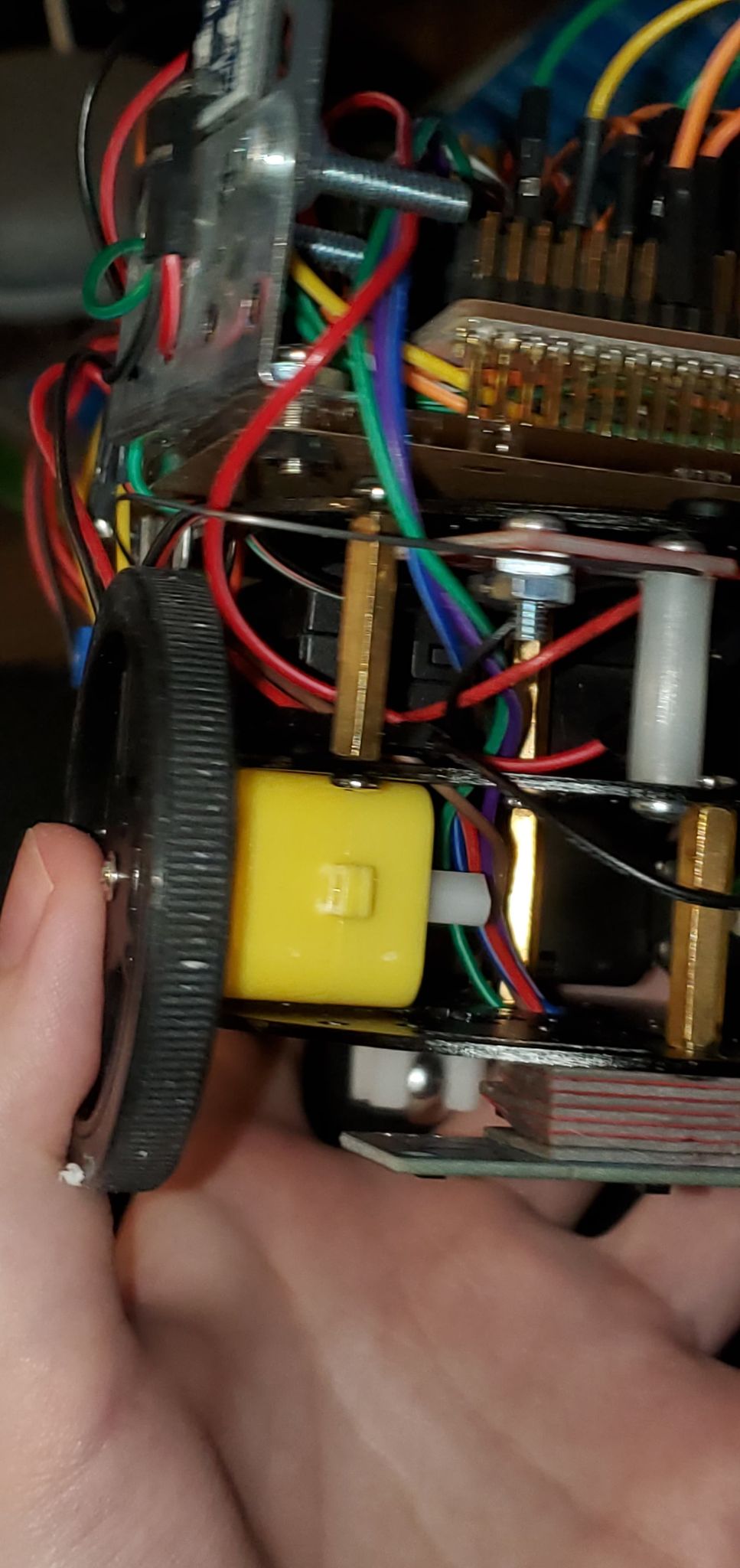
This image showcases the device from its left side - as well as its sensor

* + 1. **Right IR & Upwards IR**



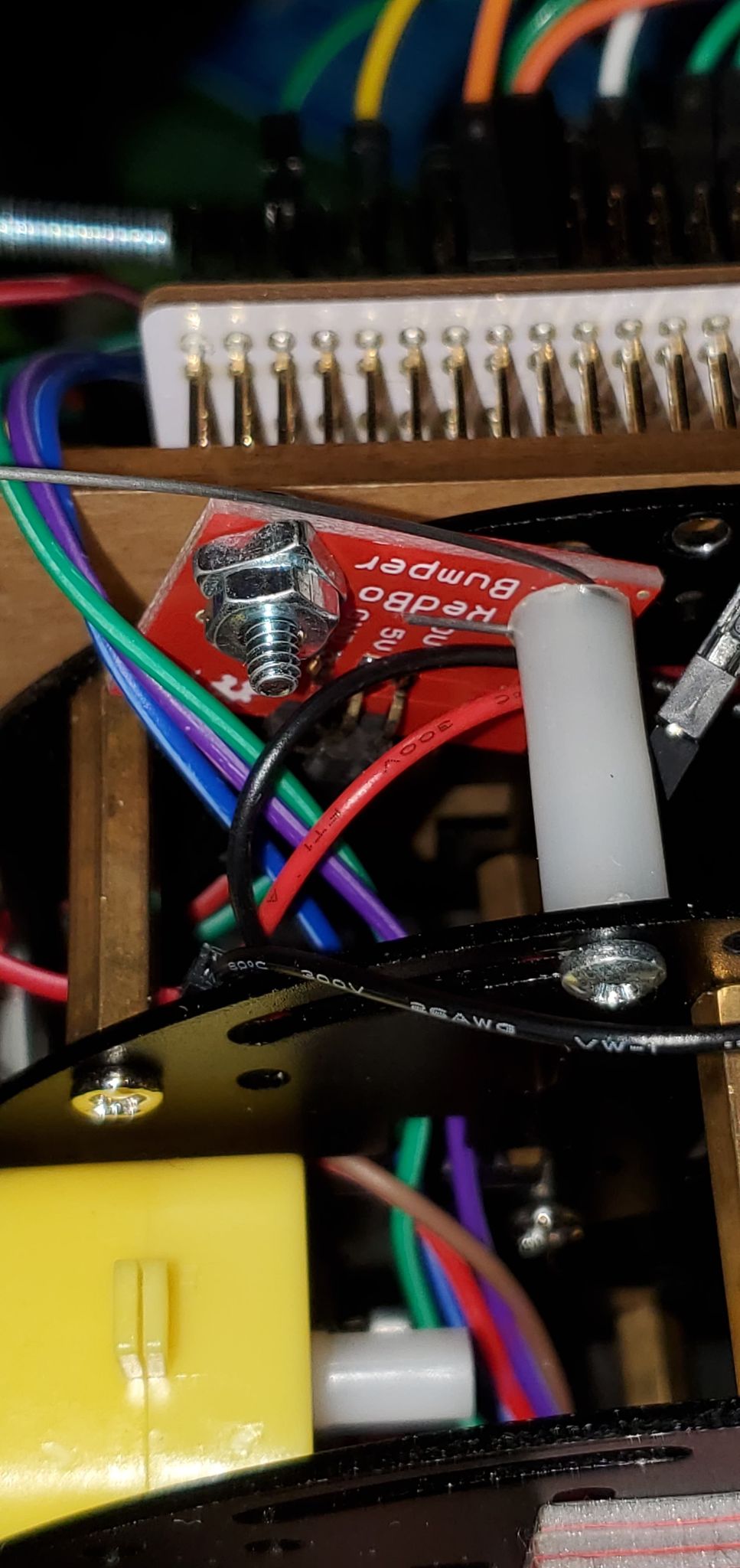
This image showcases the device from its right side - as well as its sensor. You may also observe the upper IR and associated cable shroud

* + 1. **Cable Shroud & Management**



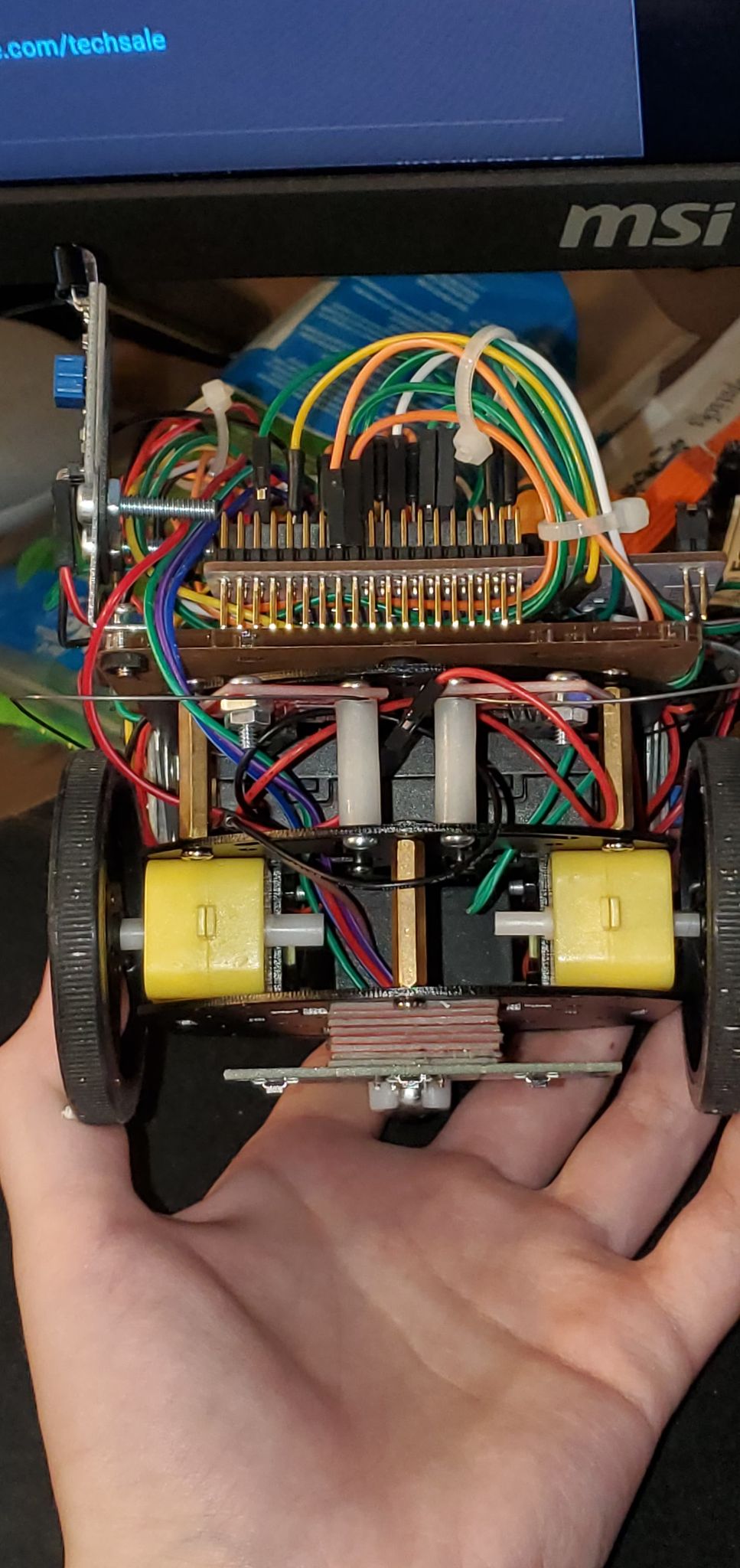
A look into the wires hidden under the shroud and Nucleo64

* + 1. **Right Bumper Sensor**



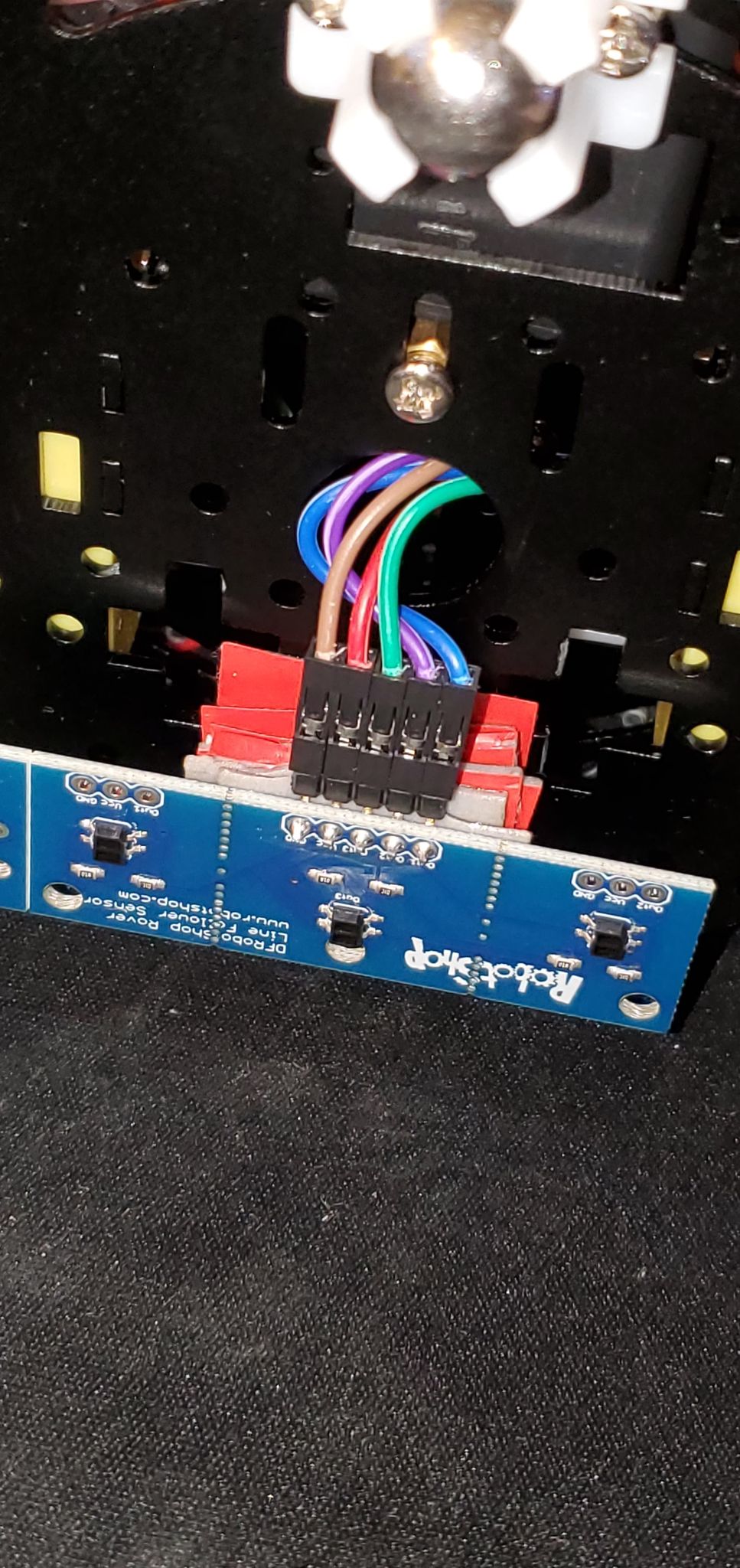
One of the two bumpers

* + 1. **Front**



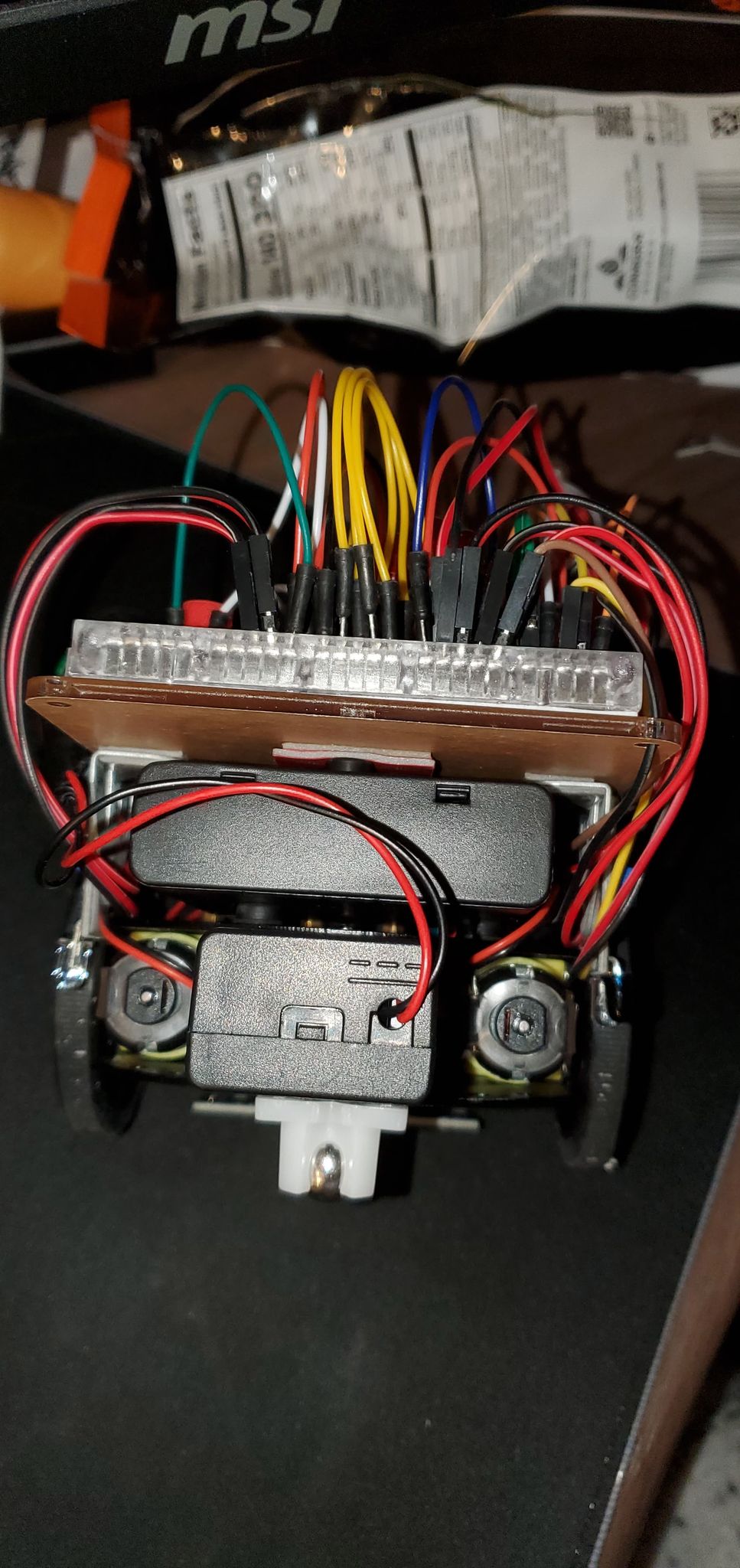
Front view of the device showcasing bumpers, motor and line follower

* + 1. **Line Follower Sensor**



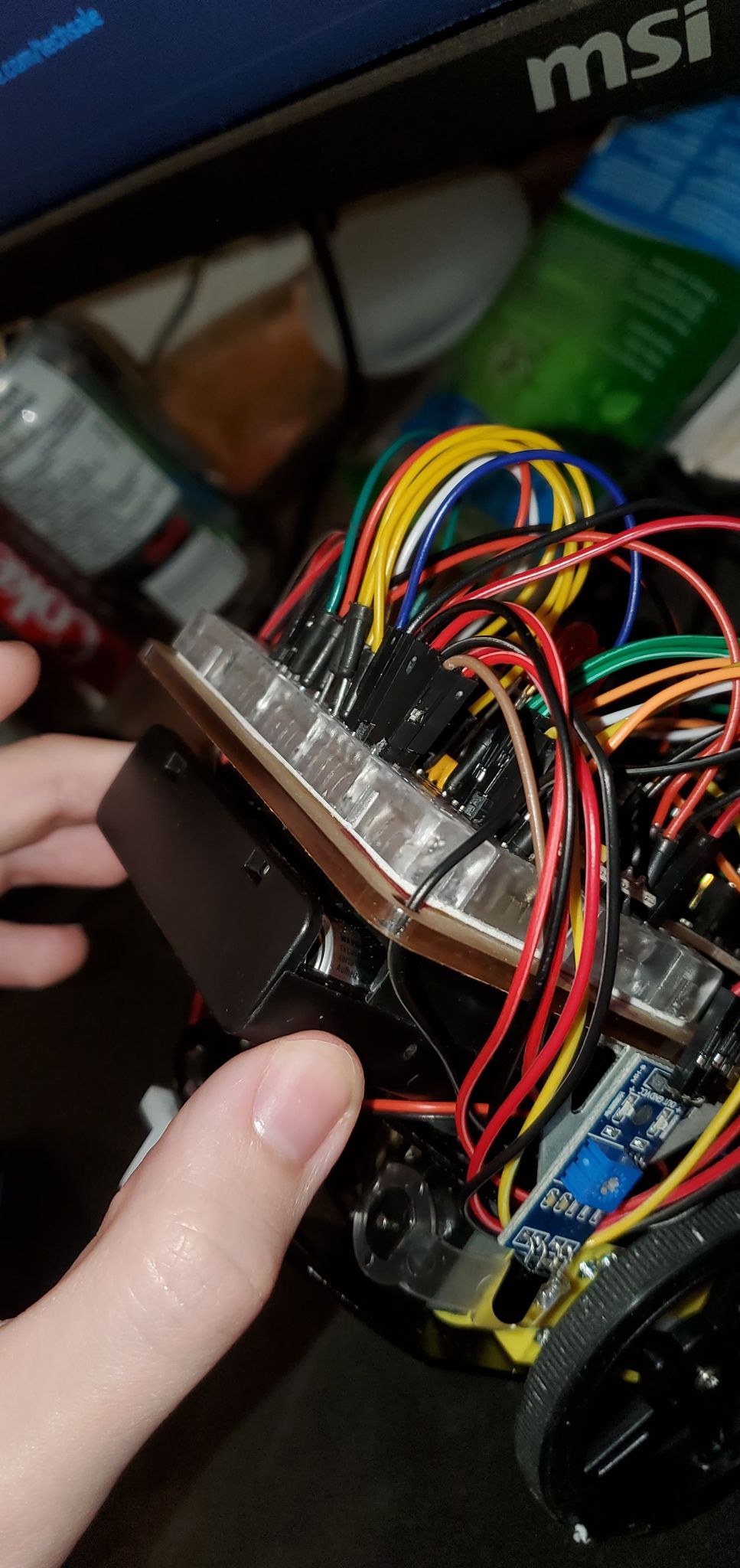
Bottom photo of the line follower sensor and its cable management

* + 1. **Rear**



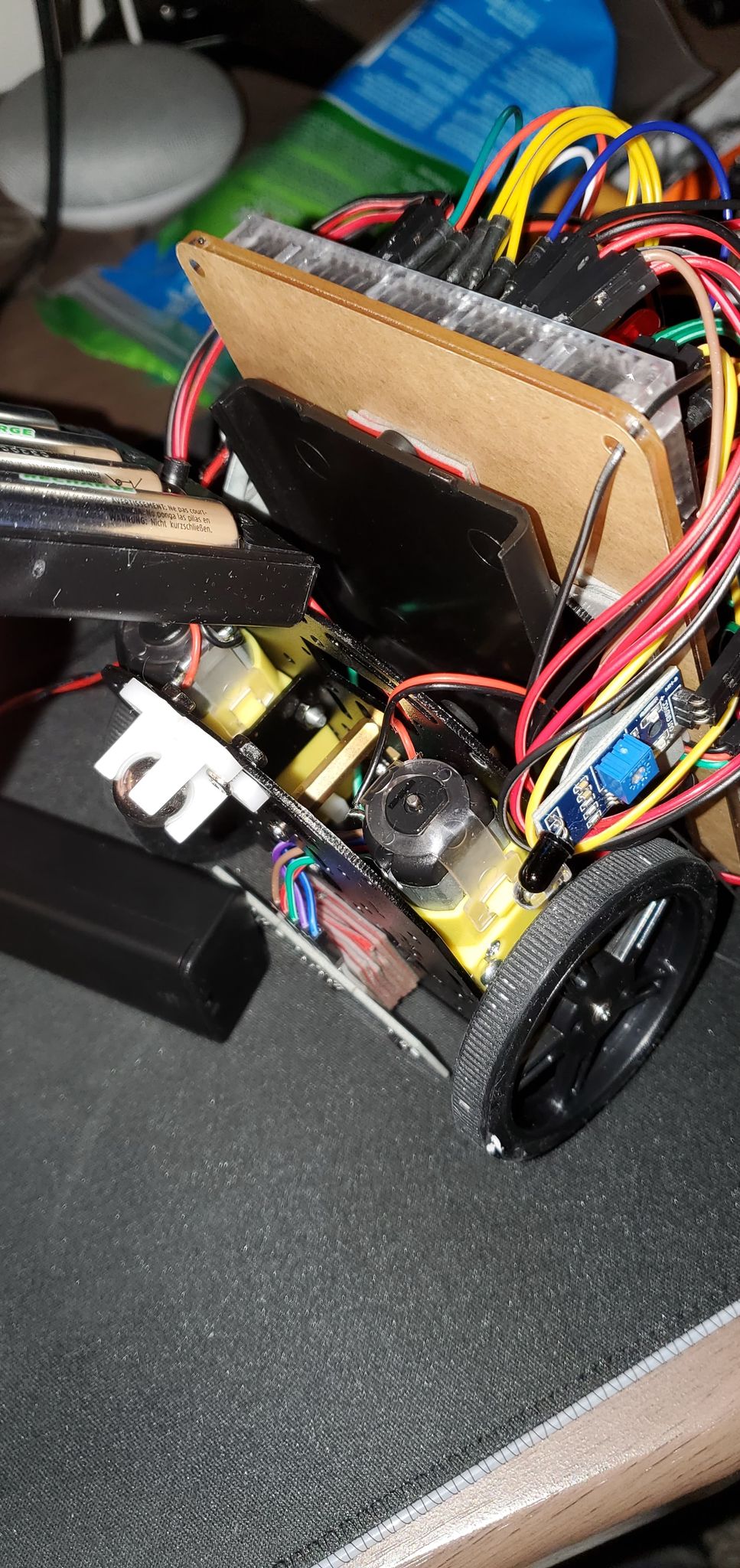
Rear photo of the device and its battery packs, including motor/battery wiring

* + 1. **Battery Rail System**



Battery rail system guiding the unit inward

* + 1. **Removed Batteries**



Batteries removed from their cradles for changing