



HeRo V2 Robot Build Tutorial

Version: 1.0
September 27, 2025
By: Paulo Rezeck, Ph.D

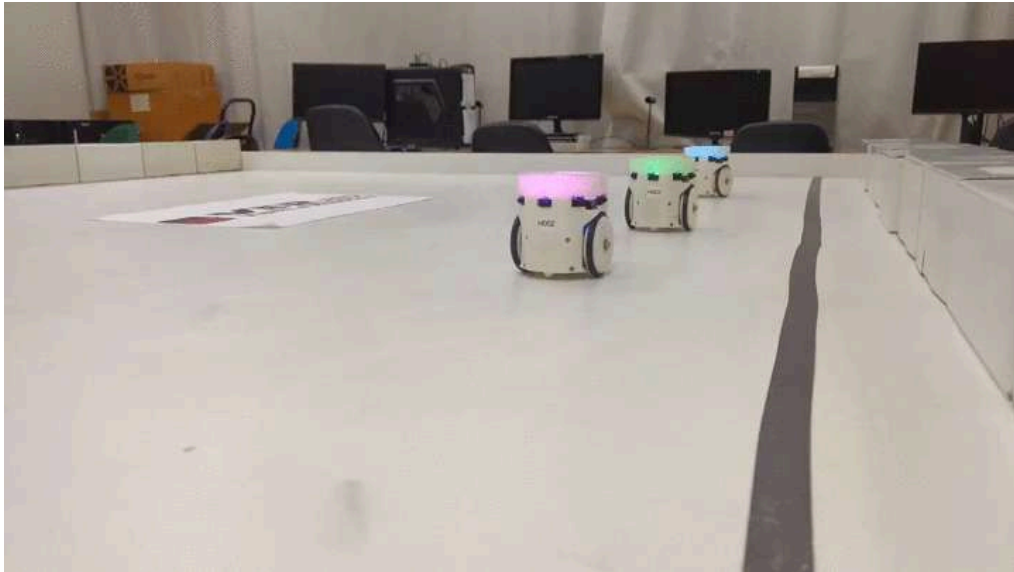
[🔗 Official HeRo page](#) · [👤 Paulo Rezeck](#)



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

Introduction

Welcome, future robot builders! Get ready to embark on an exciting journey to construct your very own HeRo V2 robot. This tutorial will guide you step-by-step through the assembly process, from unboxing to your first successful boot-up. **Let's get building!**



What You'll Need

- ☐ HeRo V2 Robot Parts
- ☐ Phillips and flat-head screwdrivers (small/medium)
- ☐ Pliers: needle-nose, diagonal cutters
- ☐ Wire cutters/strippers
- ☐ Precision knife / box cutter
- ☐ Soldering iron (fine tip), solder, flux
- ☐ Desoldering pump or braid
- ☐ Multimeter (continuity/diode)
- ☐ ESD-safe tweezers & mat
- ☐ Safety glasses
- ☐ Bench PSU with current limit (optional)
- ☐ Hot air rework station (optional)
- ☐ Isopropyl alcohol & brush
- ☐ Helping hands / PCB vise

Structure

Introduction	2
What You'll Need	2
Structure	3
Part I – PCB Board	4
A. Overview & Required Tools	4
B. Bill of Materials (BOM) – PCB subset	4
C. Soldering Order (SMD → THT)	6
D. Post-Assembly Electrical Checks	11
E. Firmware bring-up (smoke test)	12
Part II – Robot Assembly	12
A. Printed/Off-the-Shelf Parts	13
B. Main Assembly Sequence	14
C. Troubleshooting	21
Part III – Software Setup & Programming	22
A. Installation (Docker or source)	22
B. Firmware (Arduino/ESP8266)	24
C. ROS Networking & Bring-up	28
D. Robot Simulation (Gazebo)	32
Part IV – Calibration	35
A. Enter Configuration Mode	35
B. Motor PWM Calibration	36
C. Position Controller PID	36
D. Velocity Controller PID	36
E. IR Distance Sensor Calibration	36

Part I – PCB Board

Reading time: 20 min

Building time: 130 min

A. Overview & Required Tools

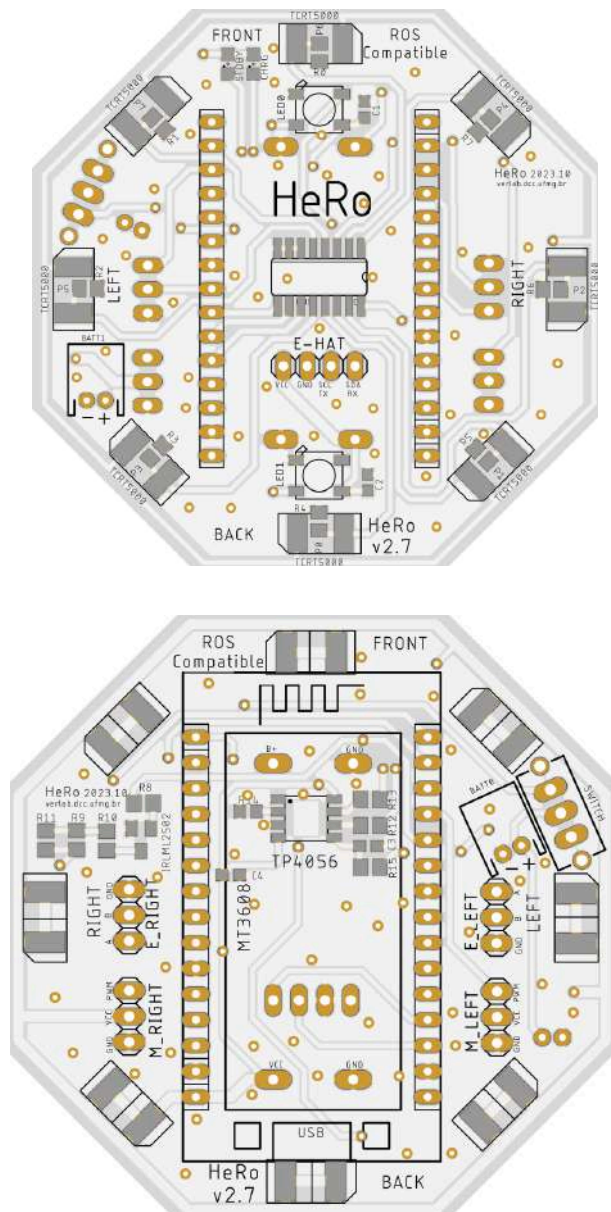
Reading time: 2 min

Building time: 0 min

Goal: Assemble the HeRo v2 main PCB, verify power, I²C/UART, LEDs, IR sensors, motor/encoder headers, and charging (optional).

Tools: fine-tip soldering iron, hot air (optional), flux pen, tweezers, magnifier, side cutters, multimeter (continuity & diode), bench PSU (current-limited), USB-TTL (if needed), ESD mat.

Safety: work current-limited (≤ 300 mA until smoke test), Li-ion handling precautions.



B. Bill of Materials (BOM) – PCB subset

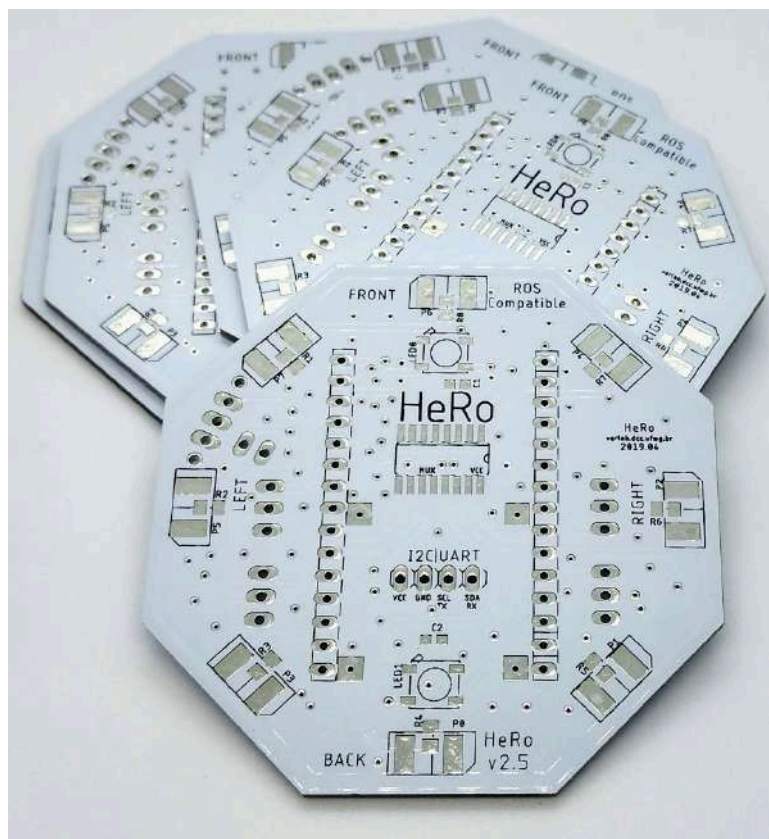
Reading time: 3 min

Building time: 0 min

Comprehensive BOM: [Materials \(HeRo v2\)](#)

(We'll keep the comprehensive BOM separate; below is a condensed checklist to drive assembly.)

- **Passives:** R0–R8 (4k7), R9 (0Ω), R10 (18Ω), R15 (0Ω), C1–C2 (100 pF), *R11 NM*.
- **Indicators:** LED0–LED1 (WS2812B 5050), STDBY (optional - green 0805), CHRG (optional - red 0805).
- **ICs/Modules:** IRLML2502 (SOT-23), CD4051 (SOIC-16), TP4056 (SOP-8, optional charger section).
- **Sensors/Headers:** TCRT5000L x8, JST-PH-2 x2 (BATT0/1), 1x4 female (I²C|UART), 1x3 male x4 (E/M LEFT/RIGHT), slide switch SS12F23.
- **MCU/Power (modules):** ESP-12E / NodeMCU-ESP8266 (CP2102), MT3608 boost, MPU6050/GY-521 IMU, TP4056 charging module (optional-if using external board).
- **PCB (blank):** HeRo v2 main board where all components are mounted. If you still need to fabricate it, see [PCB Order](#).



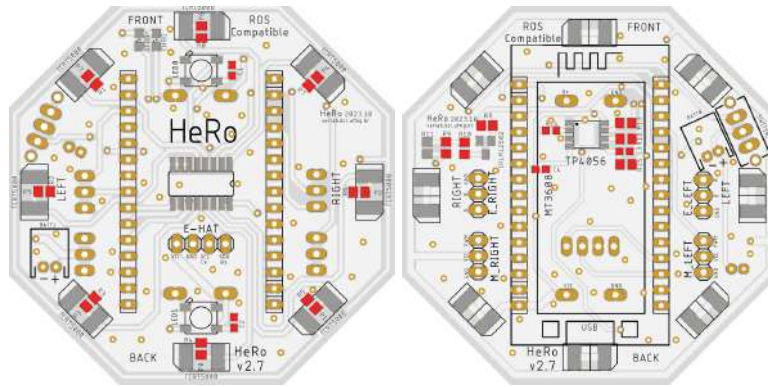
C. Soldering Order (SMD → THT)

Reading time: 10 min

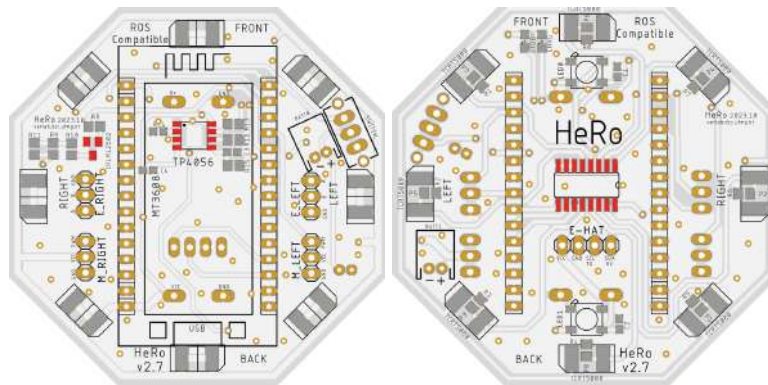
Building time: 90 min

Use this exact order to minimize rework. Place parts, tack two pads, reflow with flux. Red means where to put the solder.

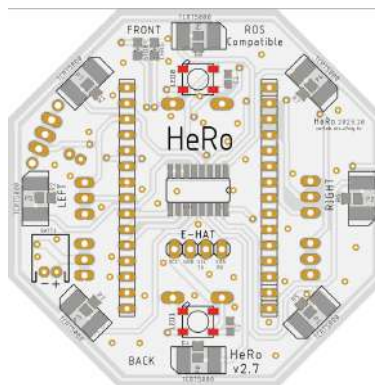
1. **Small passives:** C0603 → R0805 (C1–C2, C3–C4 optional; R0–R10, R12–R15). **Note R11 NM.**



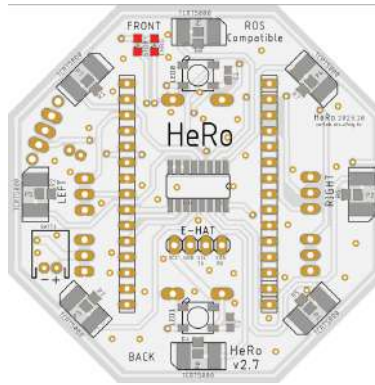
2. **ICs:** TP4056 (optional charger), CD4051, IRLML2502 (orientation dot to pin-1). Inspect bridges.



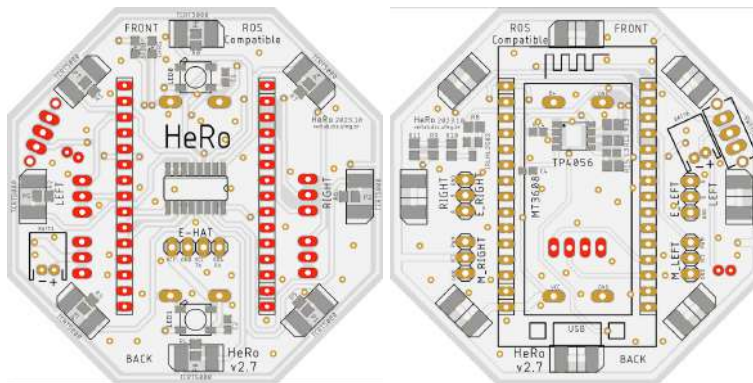
3. **LEDs:** WS2812B (arrow DIN/DOUT orientation, aligned to silk polarity). Brief continuity test to check soldering quality.



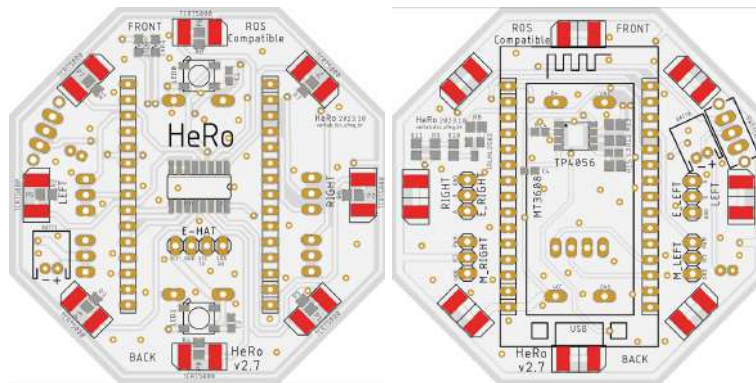
4. **Status LEDs:** STDBY (G), CHRG (R) aligned to silk polarity.



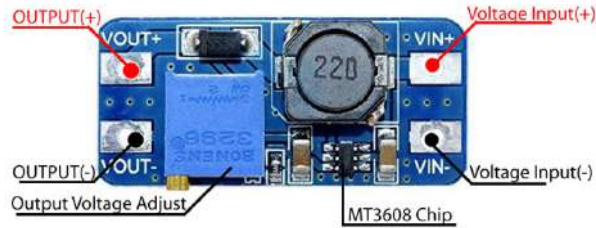
5. **Connectors/Switch:** JST-PH-2 x2, 1x4 female (I²C|UART), 1x3 males (E/M) x4, 1x16 female (ESP) x2, slide switch.



6. **Sensors:** TCRT5000L x8 (uniform standoff and coplanarity; align gap to line-track direction).

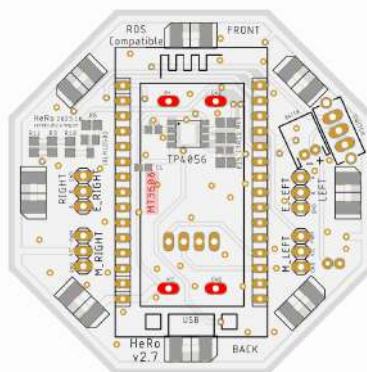
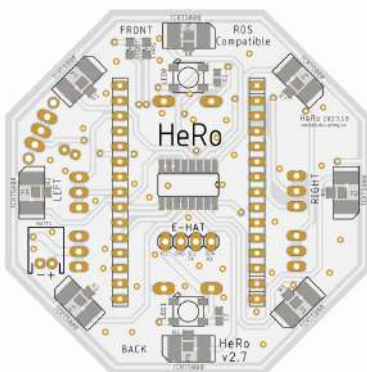


7. **Step-up module (MT3608) – PRE-TUNE ONLY:** Off-board, trim the module to **5.00 V** (bench PSU current-limited; no load or light dummy load). Do **not** solder to the PCB yet.



- PSU → module input:** PSU **+** to **VIN+**, PSU **GND** to **VIN-** on the MT3608.
 - Multimeter (DC volts) → module output:** Red probe to **VOUT+**, black probe to **VOUT-**. Confirm meter is on **V** (not A/Ω).
 - Optional second meter (if available): measure **PSU input current** in series with **VIN+** for sanity checks.
 - Procedure:**
 - Set bench PSU to **4.0 V**, current limit **0.30 A**. Power the module.
 - Measure **VOUT+/VOUT-** with the DMM.
 - Turn the trimpot (typically **CCW ↑Vout**). Approach **5.00 V** slowly; avoid overshoot.
 - Add **dummy load** and re-trim under load:
 - Light check:** 100 Ω (≈50 mA)
 - Robot check:** 6.2 Ω (≈0.8 A) or 5 Ω (≈1.0 A) *only in short bursts* to confirm stability.
 - Observe **input current** as sanity check:

$$I_{in} \approx (V_{out} \cdot I_{out}) / (V_{in} \cdot \eta) \rightarrow \text{for } 5\text{ V} \cdot 1\text{ A} @ 3.7\text{ V and } \eta \approx 0.85 \Rightarrow \sim 1.6\text{ A in}$$
(beyond small modules for continuous use). If the module overheats or **sags**, derate to **≤0.8 A** or upgrade the converter.
 - Power off. Mark the module **"5 V"**.
 - Secure the trim:** place a **small drop of clear nail polish** or **low-strength (blue) threadlocker** on the trimpot screw to prevent drift. Avoid high-strength grades; ensure the sealant is **non-conductive** and keep it off pads.
8. **Solder step-up module:** Verify VIN/VOUT polarity and footprint, then solder the pre-tuned module to the PCB. Inspect joints and clearance.

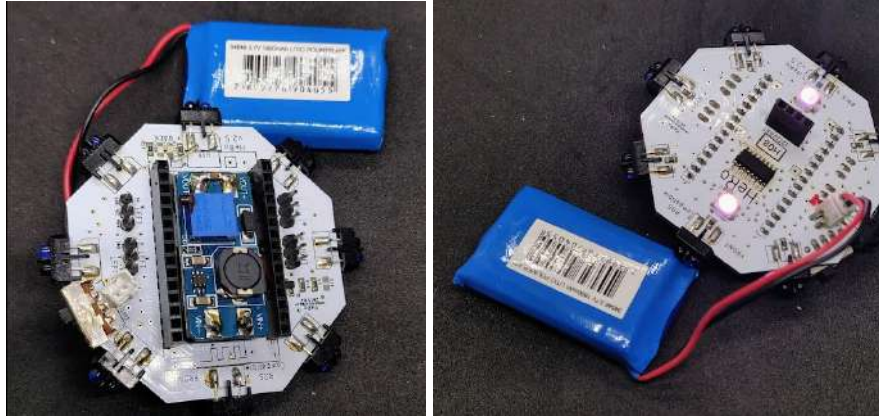


D. Post-Assembly Electrical Checks

Reading time: 3 min

Building time: 10 min

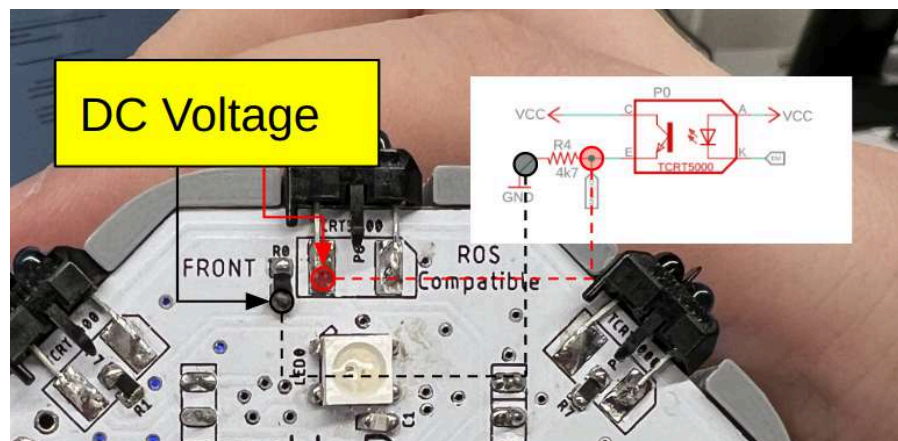
- **Continuity (unpowered):** GND plane integrity; 0 Ω jumpers R9/R15; no shorts 5 V \leftrightarrow GND, 3V3 \leftrightarrow GND.
- **Power-up (bench PSU or Battery):** set 5.0 V/0.3 A limit \rightarrow switch ON \rightarrow quiescent draw note. No component overheating.



- **LED test:** drive WS2812B with simple test pattern (rainbow or blink).



- **I²C pull-ups:** verify ~ 4.7 k Ω to 3V3; scan bus \rightarrow MPU6050 ACK.
- **MUX:** exercise CD4051 channels with test script; check IR phototransistor voltages with a white/black card.

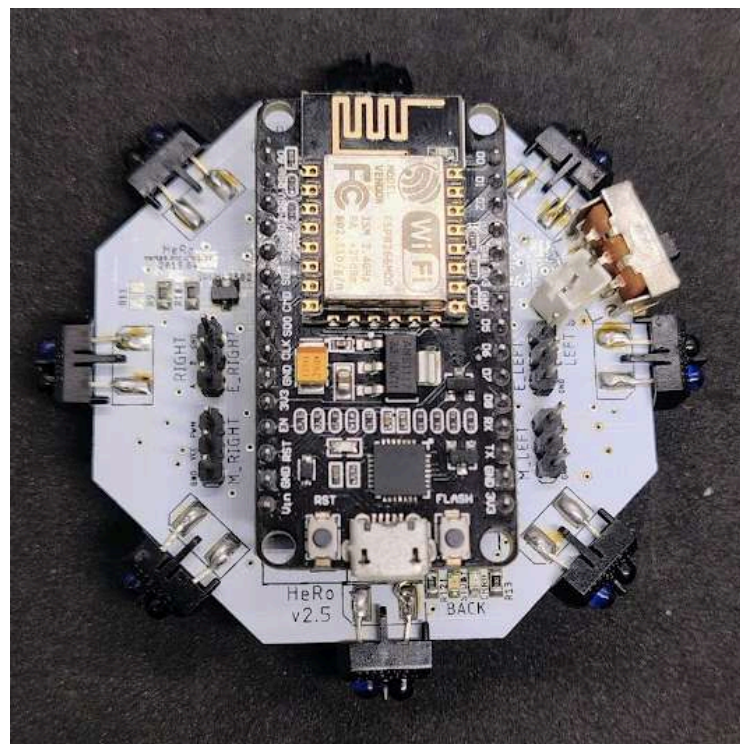
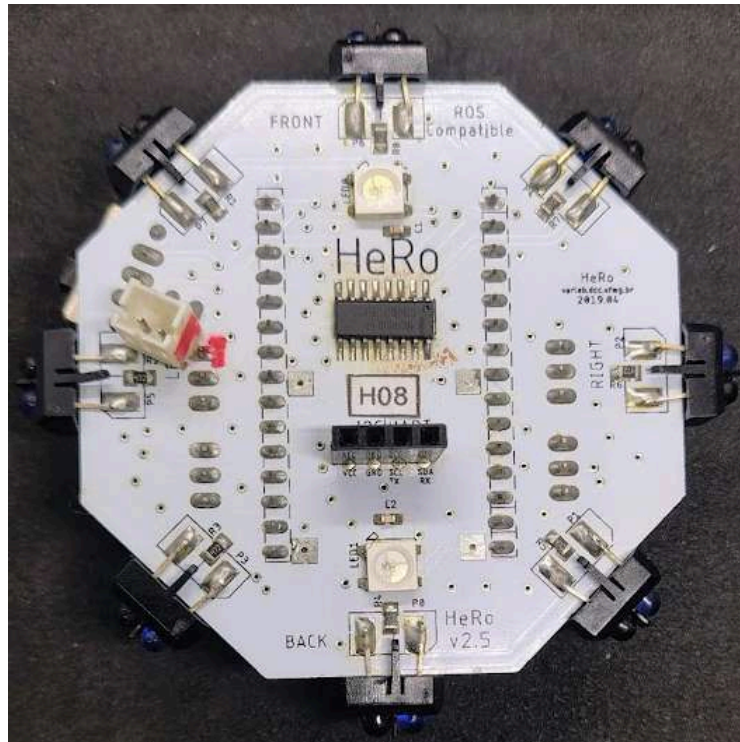


E. Firmware bring-up (smoke test – optional)

Reading time: 3 min

Building time: 30 min

Flash minimal firmware on ESP8266 (blink WS2812B, I²C scan, print ADC across IR channels via 4051). Ensure UART via CP2102, Wi-Fi AP boots. Repository tests are [here](#):



Part II – Robot Assembly

Reading time: 45 min

Building time: 110 min

A. Printed/Off-the-Shelf Parts

Reading time: 3 min

Building time: 0 min

- **Materials & BOM:** https://verlab.github.io/hero_common/materials/
- **3D printed parts (STLs & guidance):** https://verlab.github.io/hero_common/printing/
- **Print tips:** 200 µm layer height; PLA/ABS recommended; deburr holes; lightly sand mating faces; verify fit before final assembly.

Printed set (checklist):

- ☐ **Castor wheel:** Screw-on castor wheel for robot balance adjustment.
- ☐ **Chassis B:** Chassis that supports the mainboard and encoders;
- ☐ **Chassis A:** Chassis that supports the motors and wheels;
- ☐ **Cover:** Dust protection cover;
- ☐ **Cover usb:** Sliding part for access to the USB connector;
- ☐ **E-hat:** e-hat component to enclose mainboard expansions;
- ☐ **E-hat glove:** Adapters for e-hat component;
- ☐ **Encoder shaft:** Spur gear/shaft encoder for wheel-encoder transmission;
- ☐ **Motor gear:** Coupled to the servo motor shaft and acts directly on the wheel;
- ☐ **Motor shaft:** Wheel shaft supports the connection of the wheel to the chassis;
- ☐ **Wheel:** Wheel spur geared;.

Off-the-shelf:

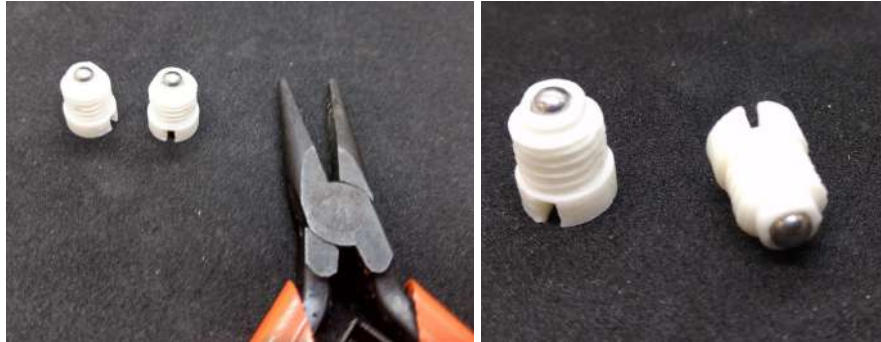
- ☐ **Drive:** SG90 continuous-rotation servos ×2 (w/ horns & screws)
- ☐ **Locomotion:** O-ring tires ×2, 623ZZ bearings ×2
- ☐ **Support:** steel balls 4 mm ×2 (casters)
- ☐ **Fasteners:** M2.5 self-tapping screws, assorted M3 screws/nuts/washers
- ☐ **Electronics:** Li-Po 103450 (1S), pcb board, IMU sensor, encoders sensor

B. Main Assembly Sequence

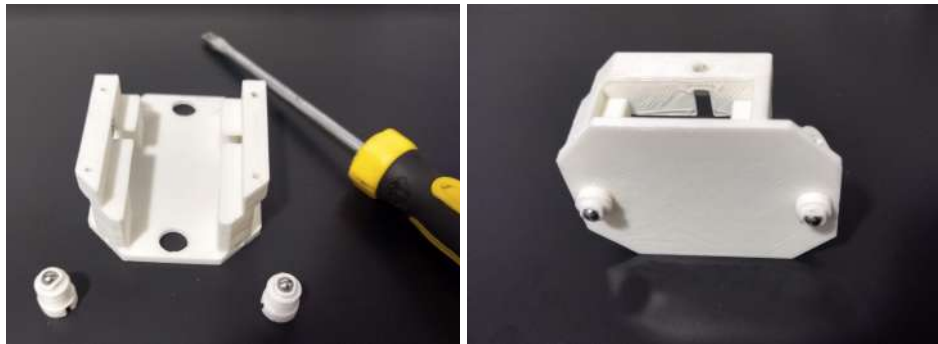
Reading time: 35 min

Building time: 110 min

1. **Caster wheels:** take the printed caster shell; insert a **4 mm steel ball** into the cup. Use pliers to gently crimp the rim just enough to retain the ball while keeping free rotation.



2. **Mount casters to Chassis A:** align the two casters to the front/back positions on **Chassis A** and fasten with the specified screws. Confirm ground clearance and smooth swivel.



3. **Prepare motors (SG90 CR - ⚠ if needed):** ideally purchase **SG90 servos already adapted for continuous rotation**, since most stock SG90 units are angular-limited.

⚠ Be aware there are different internal variants (encoder types) even with the same SG90 form factor. The green-label variant is recommended, as it allows easy CR adaptation without replacing resistors or other components.

🔧 If you only find a standard (non-CR) type, follow a trusted tutorial such as [this video](#) to perform the continuous-rotation modification. After confirming CR function, power the servo and send **~1500 μ s** to center it. Then trim/select the horn that best seats in the **printed gear adapter**.



4. Couple the motor gear to the servo shaft (gear adapter method):

a) After converting the servo for **continuous rotation** (if needed), choose a stock **servo horn** that best fits inside the printed **motor gear/adaptor**. **Cut the horn**, leaving only the **cylindrical hub/connector**.

⚠ Attention: wear safety glasses when cutting, as small plastic pieces can eject unexpectedly and pose a risk to eyes or face.



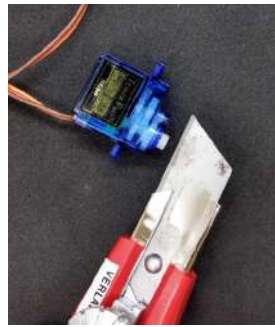
c) **Trim the cylindrical hub** evenly until it **press-fits** into the printed motor gear (snug, not loose). Then, apply a **small drop of cyanoacrylate (super glue)** at the interface to lock the hub in the gear; avoid excess glue near teeth.

⚠ Attention: handle super glue carefully to avoid contact with eyes, mouth, or skin; always work in a ventilated area and use protective equipment if possible.

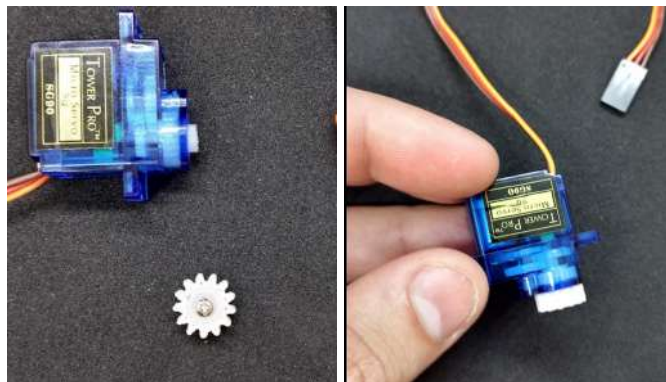


e) If the servo shaft protrudes, **shorten ~1.00–1.20 mm** so the gear sits as close as possible; make the cut **flat** (file if needed).

⚠ Attention: when cutting the shaft with a precision knife, wear gloves and keep the cut clean and perpendicular. Work slowly to avoid slipping and damaging the plastic housing or injuring yourself or others nearby.

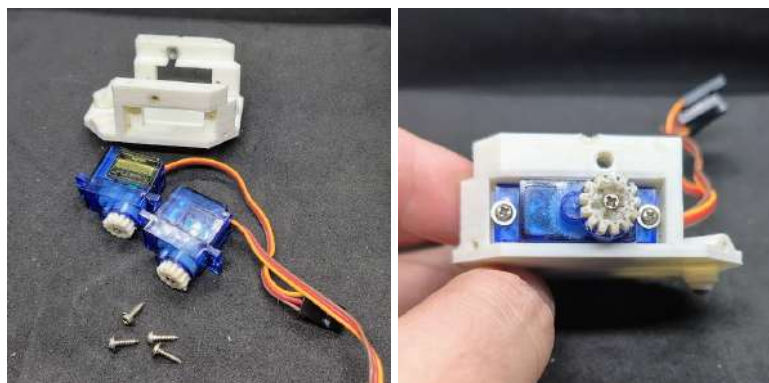


f) **Fasten** the gear/hub to the servo output with the **original horn screw** from the servo kit.



🔧 Tips: After fastening, avoid overtightening the horn screw—too much torque increases friction and can damage the gear interface. Connect the servo to the board (or any PWM controller) and run it in both directions to check constant smooth rotation. Adjust screw tightness until rotation is stable. Lightly press the gear by hand to ensure the glue bond holds. The fit must be neither too tight nor too loose; gear teeth should drive the wheel smoothly with minimal backlash.

5. **Install motors on Chassis A:** position each servo in its printed mount on **Chassis A** and fasten. Verify both outputs are aligned when commanded to stop (1500 μ s).



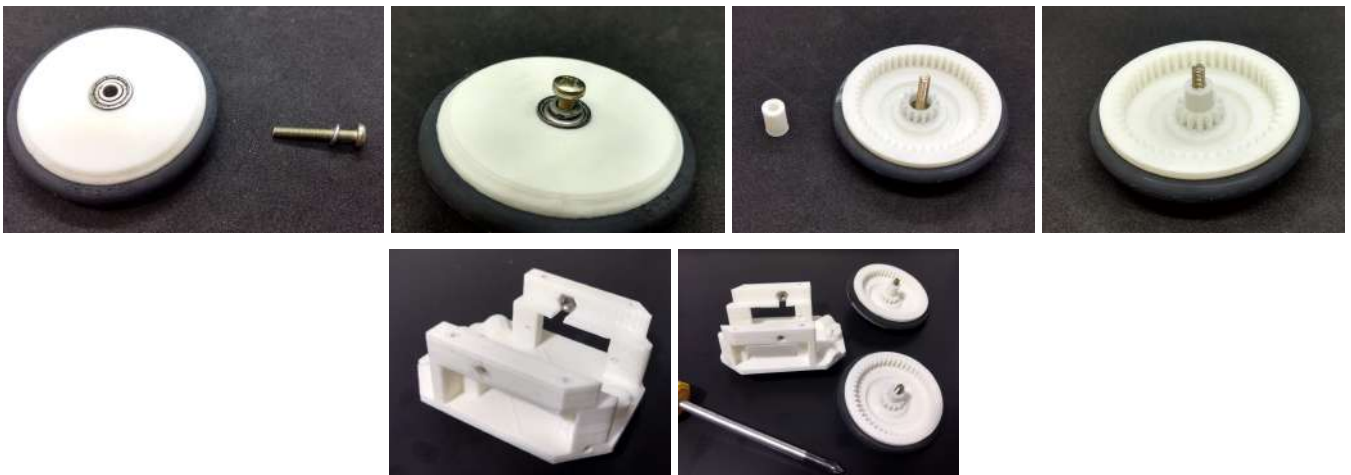
🔧 Tip: do not overtighten the mounting screws—leave them slightly loose to allow a bit of flexibility in the shaft-to-wheel transmission. Carefully align both motors on each side for proper symmetry. Trim motor cables to about **10 cm** length to simplify cable routing inside the robot.

6. **Build wheels:** press-fit a **623ZZ** bearing into each printed wheel hub (use a vise/press; avoid striking directly—if needed, tap gently through a flat spacer). Fit the **O-ring** tire around the rim.



7. **Install wheels & support screw:** place the wheel onto the gear adapter/output, add the designated **support screw/axle** on the opposite side if your design uses one, and secure. The gear mesh between motor gear and wheel should be smooth with minimal backlash.

⚠ Attention: the 3D-printed shaft sleeve that couples the screw and the motor shaft must allow the wheel to rotate smoothly with almost no backlash. Ensure a tight but precise fit—avoid wobble, as it is critical for proper transmission.



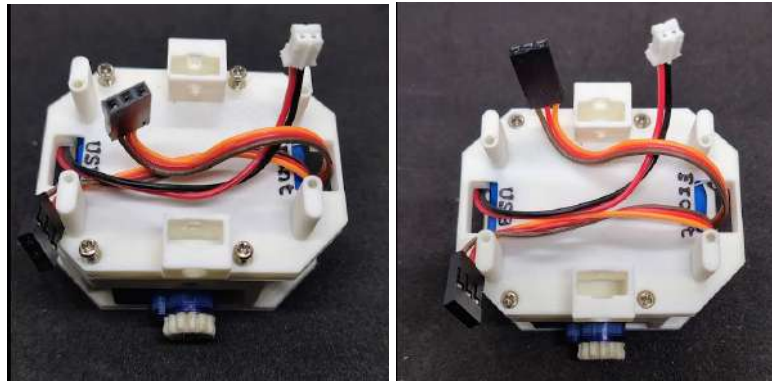
🔧 Tip: add a tiny drop of glue to the support screw where it threads into Chassis A, to prevent it from loosening or falling out during handling and wheel installation.

8. **Battery placement & routing:** mount the **Li-Po** between Chassis **A** and **B** (in the battery cradle). Route the **JST** cable up to the electronics bay on top of **Chassis B**; secure with ties.

⚠ Attention: observe the labels on Chassis B ("USB" and "Front"). Always pass the battery cable through the **USB hole** and the motor cables through the **Front hole** to ensure safe and clean routing.



9. **Add Chassis B (upper):** stack **Chassis B** over **Chassis A** and secure with **four screws**.



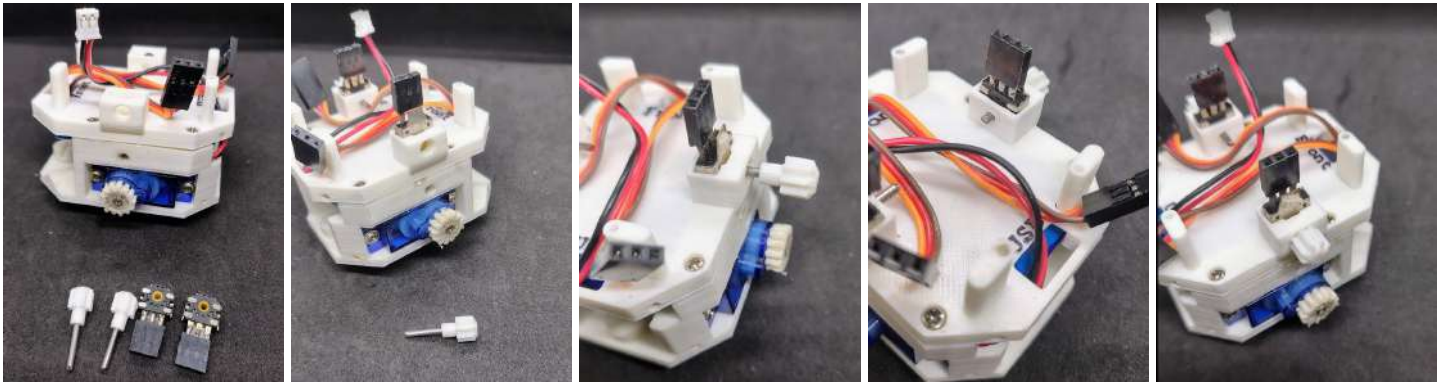
10. **Prepare the quadrature encoder:** use a mouse-style mechanical quadrature encoder. Remove the lateral metal spring/detent to eliminate the tactile "clicks" (bend or cut the spring carefully) so rotation is smooth. Solder a **JST-PH** pin header to the encoder terminals for easy wiring.

⚠ Attention: when cutting or trimming the encoder shafts, wear safety glasses and protective gloves to prevent small parts from ejecting and injuring yourself or others nearby. Always direct the cutting tool away from people, and keep bystanders clear.



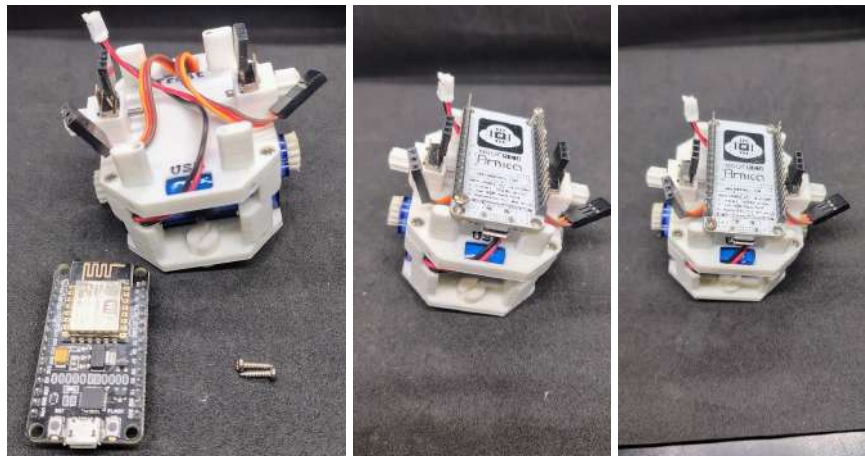
🔧 Tip: cut the encoder pins/shafts as precisely as possible to the required length. When bending or disabling the detent, verify that the shaft rotates completely smooth, with no tactile steps—any residual friction can cause missed counts or noisy readings. For wiring, solder a JST-PH female header to the encoder terminals, trimming it to the correct size so it fits cleanly into the PCB connector.

11. **Install encoder into Chassis B:** insert the encoder into its pocket on **Chassis B** directly above the wheel position.



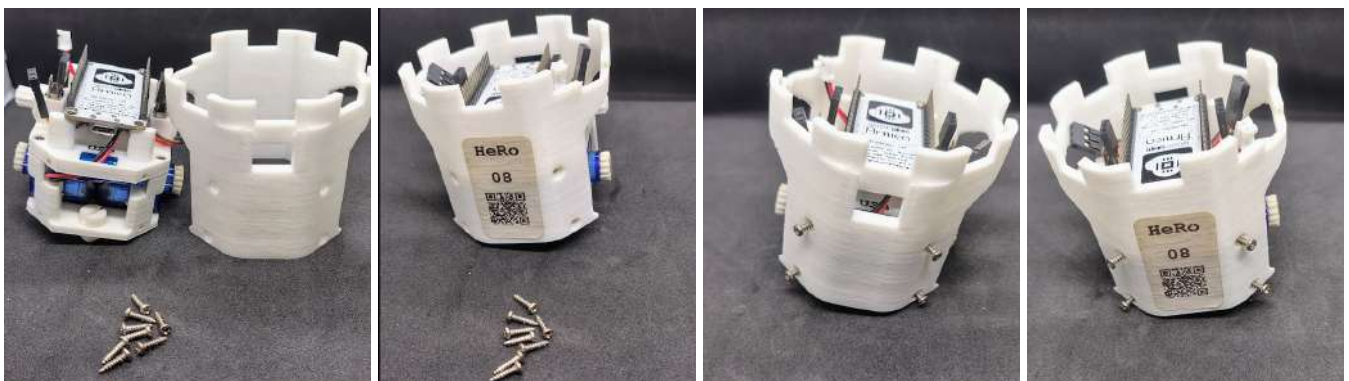
🔧 **Tip:** after assembling the encoder with its shaft and small gear, gently rotate the encoder shaft to verify smooth rotation. It must not bind or create excess friction, and it should not be too loose either. Check for lateral play: the shaft should not wobble vertically or transversely—aim for a firm but smooth fit. If, later on, the motor-wheel-encoder transmission does not run smoothly, lightly sand the gears to improve meshing and reduce backlash.

12. **Controller board:** mount your controller (**ESP8266/NodeMCU**) on standoffs above **Chassis B**.



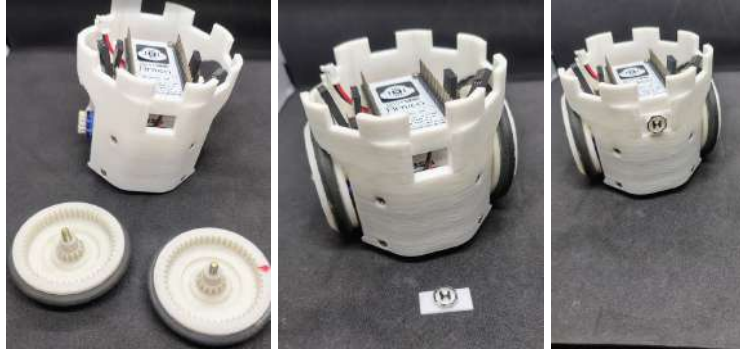
13. **Top cover:** place the protective cover over the assembly and secure it with eight screws. **Be careful not to overtighten**—use moderate torque so screws seat but don't crack plastic.

⚠️ Also **ensure no screw penetrates or presses on the battery** beneath; a mislocated or oversize screw may pierce the Li-Po and cause a short or fire risk.



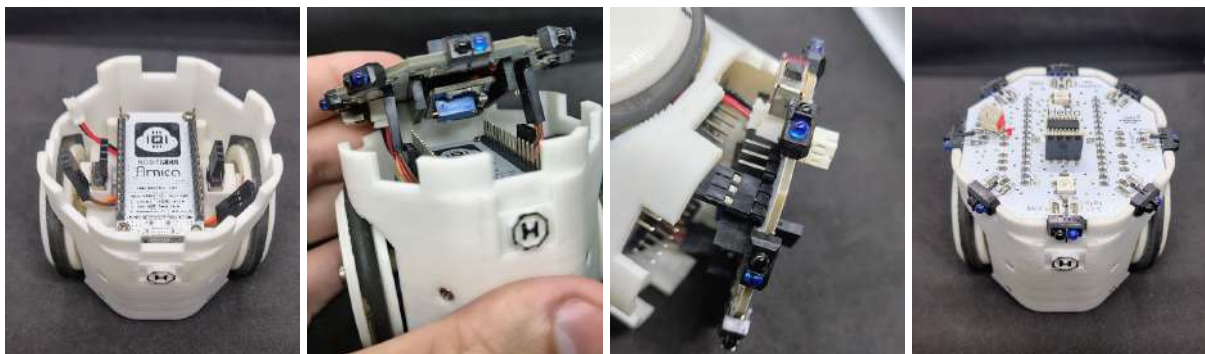
14. **Final wheel installation and coupling with encoder:** insert the wheel onto the shaft from Chassis B at this stage (although wheels can be temporarily fitted earlier for local tests).

🔧 **Tip:** Secure with the proper screw, ensuring the gear train from motor → wheel → encoder runs smoothly with minimal backlash. Gently turn the wheel by hand at low force to check for pendulum or wobble. If rotation is rough or too tight, lightly sand the gear teeth for a better fit. The transmission should remain coaxial, smooth, and free of binding.

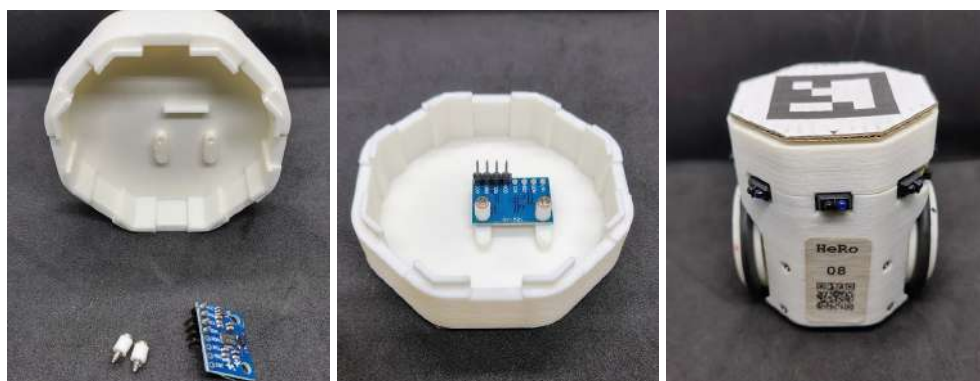


15. **Electrical connections:** plug both **servo** leads (signal to PWM, red to 5 V, brown/black to GND) and the **battery JST** into the main PCB. Seat the PCB into its mounts as shown in the figure, and double-check polarity and strain relief.

⚠️ **Attention:** never connect the plugs inverted; a reversed connection may damage the battery, motors, or sensors.



16. **E-hat module:** solder male header pins onto the IMU board (gyro+accelerometer). Mount the IMU securely into the 3D-printed e-hat case with small screws as shown in the figure. Then fasten the printed e-hat case above the PCB board. The header pins should align so the I²C and power lines plug directly into the I²C connector on the main PCB.



17. **Final mechanical check:** wheels free-spin, encoder linkage aligned, casters swivel, no cable rub. Proceed to Part III for calibration.



C. Troubleshooting

Reading time: 5 min

Building time: 0 min

Common issues & fixes:

- **Motors not spinning / weak torque:** check servo wiring (PWM–5 V–GND), confirm boost module is holding 5 V under load, verify battery voltage >3.6 V.
- **Wheels wobble or backlash excessive:** re-seat wheel hub, ensure bearings are fully pressed, lightly sand or shim gear adapter for tighter fit.
- **Encoder counts intermittent:** confirm detent spring fully removed, check JST-PH header solder joints, verify no lateral wobble on shaft.
- **IR sensors not responding:** check orientation, LED polarity, and verify current-limiting resistors populated.
- **Board does not power:** double-check MT3608 tuned to 5.0 V, confirm JST polarity (never reverse Li-Po), inspect switch soldering.
- **Overheating boost converter:** load >1 A continuous; derate to ≤ 0.8 A or upgrade module, add heatsink if needed.

Part III – Software Setup & Programming

Reading time: 60 min

Building time: 105 min

A. Installation (Docker or source)

Reading time: 10 min

Building time: 30 min

Setting up the HeRo software environment is an essential step before you can calibrate or program the robot. You have two main paths: an easier Docker-based installation, or a manual build from source. Both approaches share the same software dependencies.

Dependencies

You will need to have the following software available:

- ROS (tested with **Melodic** and **Noetic**)
- roserial for microcontroller communication
- Gazebo for simulation
- Qt for GUI tools
- Arduino IDE (only required if you plan to flash firmware directly)

Method 1 – Docker (recommended for beginners)

Docker provides a contained environment with all the dependencies already packaged, which reduces compatibility issues and setup time.

1. Install Docker if not already present. Follow the [official guide](#).
2. Clone the repository to your machine:

```
$ git clone https://github.com/verlab/hero_common.git
```


3. Go to the directory:

```
$ cd hero_common/hero_common
```

4. Build the container with Docker Compose:

```
$ docker compose up ros1 --build # for ROS 1 (Noetic) or
```

```
$ docker compose up --build # for ROS 2 (Foxy)
```

 ROS 2 compatibility is still limited. You can still run the package on ROS 1 and bridge to ROS 2 using [ros1_bridge](#).

After building, you can launch the containers and interact with ROS nodes as if you were running them locally, but without dependency conflicts.

Method 2 – Build from Source (for advanced users)

If you prefer direct control, or you plan to modify the source code extensively, you can set up the packages in your own ROS workspace.

1. First, make sure you have a catkin workspace configured (e.g., `~/catkin_ws`). If not, follow the [ROS installation guide](#) and [environment configuration tutorial](#).

2. Clone the repository into your `src` folder:

```
$ cd ~/catkin_ws/src
```

```
$ git clone --depth 1 --branch noetic-devel https://github.com/verlab/hero\_common.git
```

3. Resolve dependencies with `rosdep`:

```
$ cd hero_common/hero_common
```

```
$ rosdep install --from-paths src/hero_common --ignore-src -r -y
```

4. Compile the workspace:

```
$ catkin_make # or catkin build
```

5. Source your workspace so the packages are available:

```
$ source devel/setup.bash
```

✨ Tips and Recommendations

- Docker is the fastest way to get started if you just want to run the robot and simulations without tinkering too much.
- Source builds are more flexible if you intend to dive into development or debugging.
- Always confirm your ROS distro matches the branch you checked out (e.g., `melodic-devel`, `noetic-devel`).
- Keep `rosdep` updated regularly with `rosdep update`.
- Consider setting `source devel/setup.bash` in your `~/.bashrc` to load your workspace automatically when opening a terminal.

B. Firmware (Arduino/ESP8266)

Reading time: 20 min

Building time: 30 min

Flashing the HeRo firmware is required for the robot's ESP8266-based controller to communicate with ROS and execute commands. This process involves setting up the Arduino IDE with the correct board support, drivers, and project files.

Step 1 – Install Arduino IDE

Download and install the latest Arduino IDE from the official website. The IDE provides the compiler and uploader required to burn firmware onto the ESP8266.

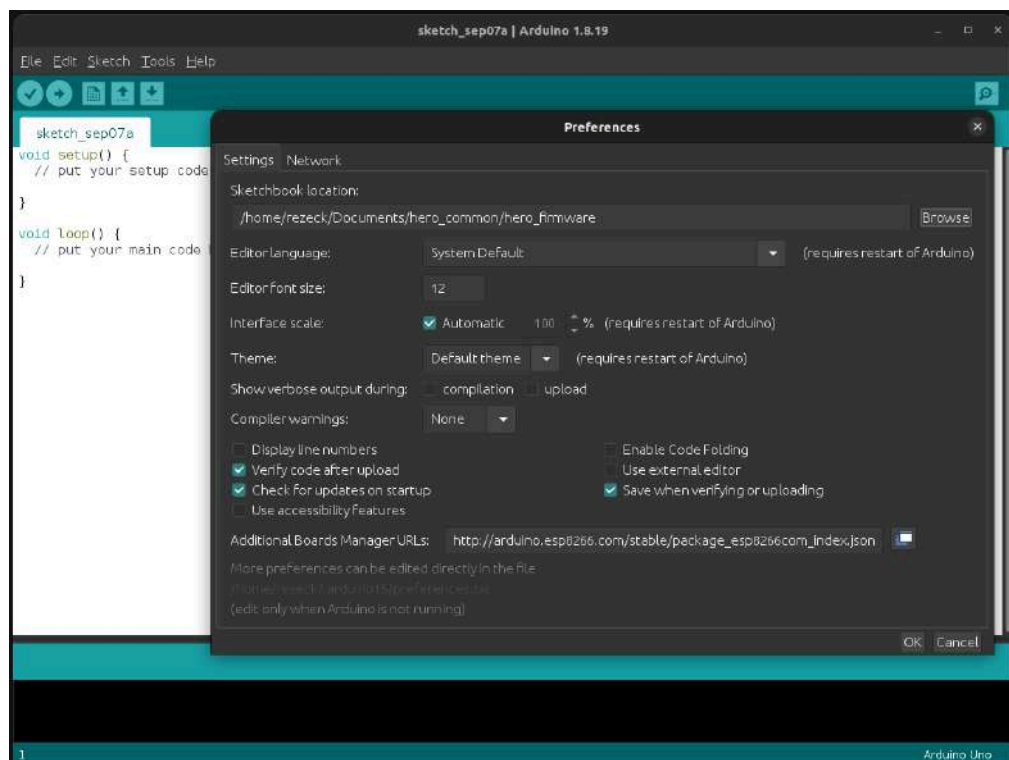
Step 2 – Install USB Driver

Most Linux distributions already include the CP2102/CH340 drivers for NodeMCU modules, but on Windows and macOS you may need to install them manually. Ensure the driver is properly installed so your computer recognizes the board.

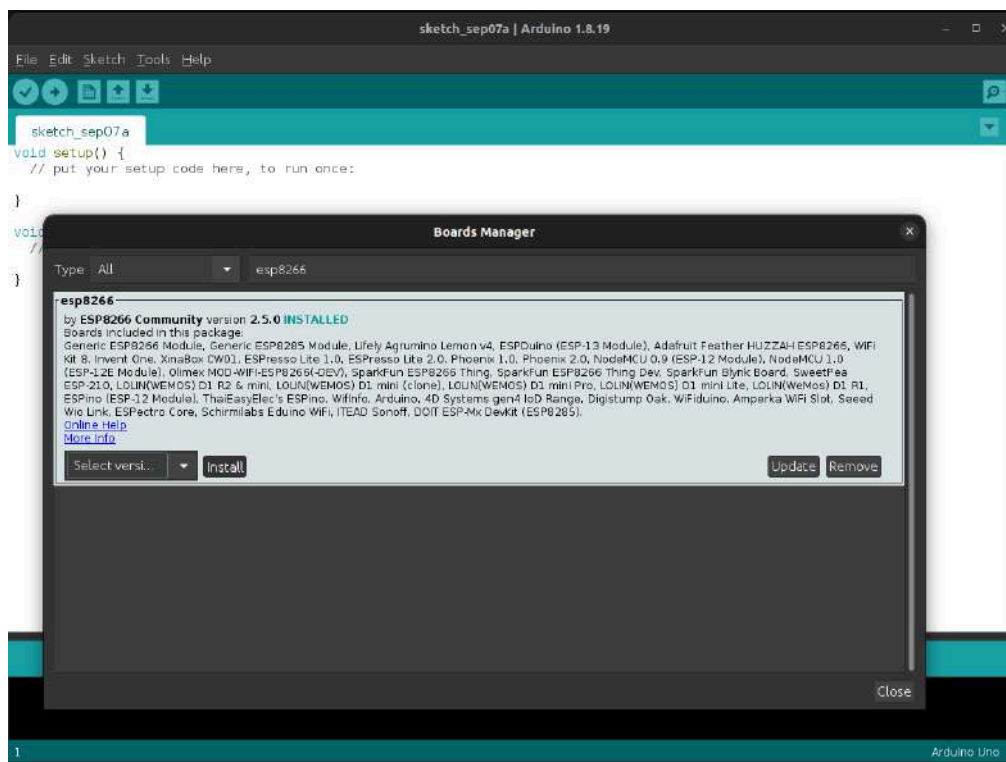
Step 3 – Configure ESP8266 Board Support

1. Open Arduino IDE → *File* → *Preferences*.
2. Add this URL under *Additional Boards Manager URLs*:

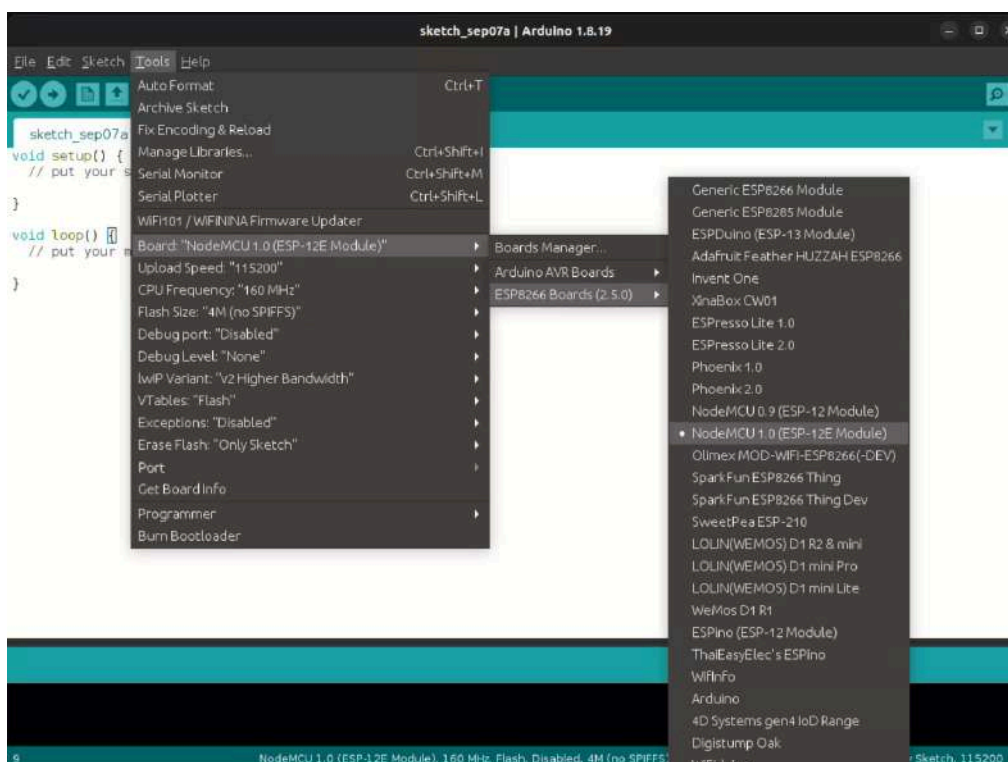
http://arduino.esp8266.com/stable/package_esp8266com_index.json



3. Restart the IDE.
4. Go to *Tools* → *Board* → *Boards Manager* → search for **esp8266** and install **version 2.5.0** (recommended for compatibility).



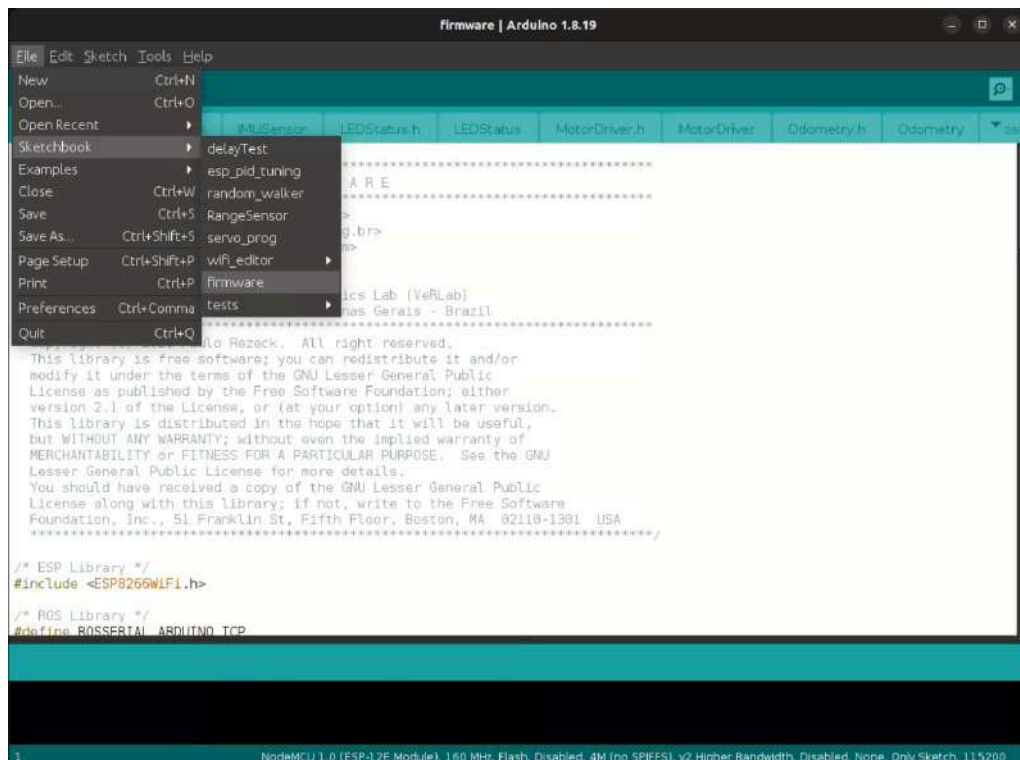
- From *Tools* → *Board*, select **NodeMCU 1.0 (ESP-12E Module)**. Confirm the other default settings (160 MHz, 4M/3M SPIFFS, 115200 baud).



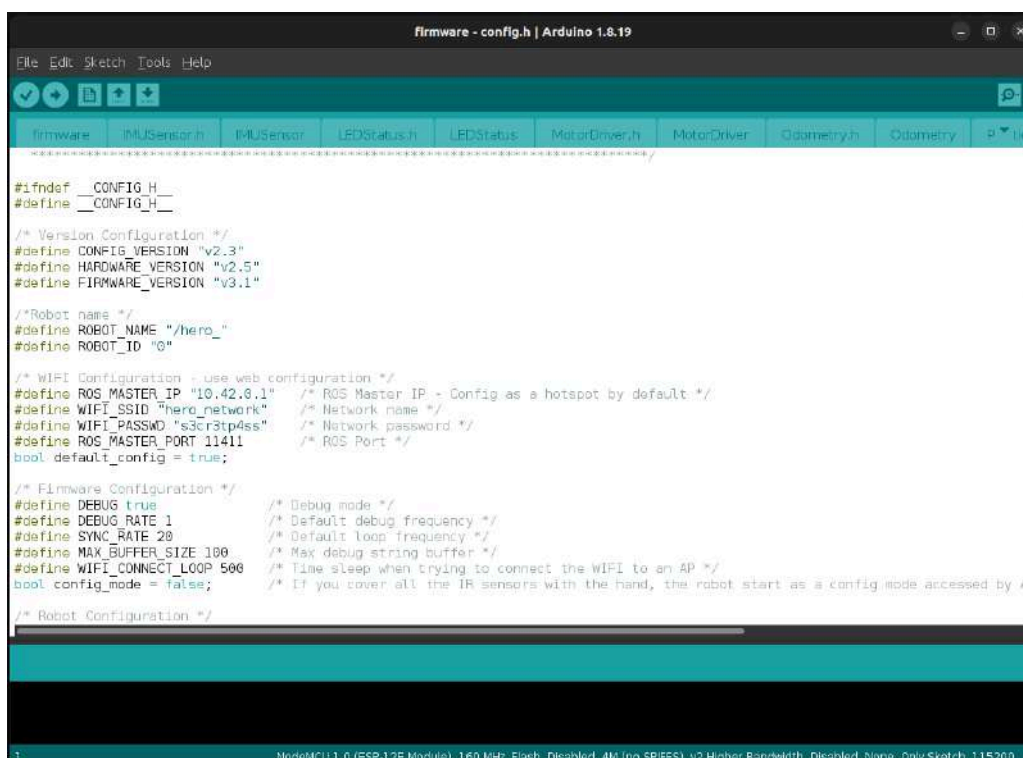
⚠ Attention: Installing other versions of the ESP8266 core may cause incompatibility. Stick to 2.5.0 unless you know what you are doing.

Step 4 – Open HeRo Firmware Project

1. In Preferences, set the Sketchbook location to the **firmware** folder inside the cloned **hero_common** repository.

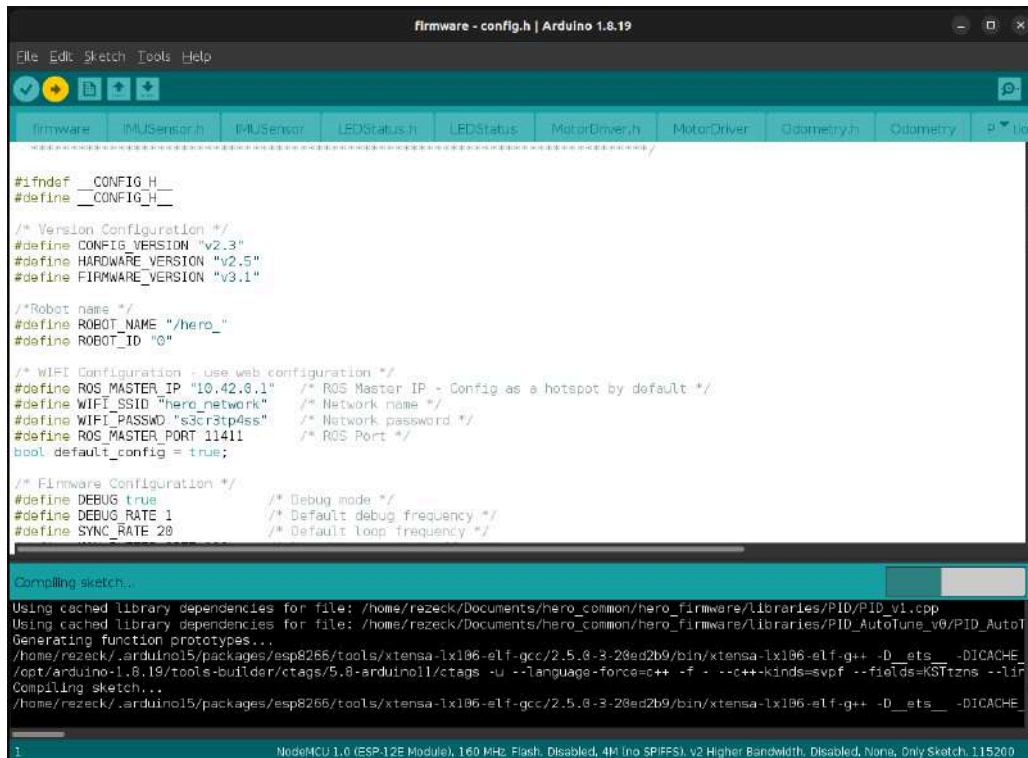


2. Restart the IDE and open the HeRo firmware sketch.
3. Review **config.h** for parameters such as Wi-Fi SSID, password, and robot ID.
 - **Tip:** avoid editing dynamic parameters directly in **config.h**. Use the built-in web interface after flashing for changes.



Step 5 – Upload Firmware

1. Connect the HeRo board to your computer via USB.
2. In Arduino IDE, select the proper serial port under *Tools* → *Port*.
3. Click **Upload**. The code will compile and be flashed to the ESP8266.



```
firmware - config.h | Arduino 1.8.19
File Edit Sketch Tools Help

firmware IMUSensor.h IMUSensor.h LEDStatus.h LEDStatus.h MotorDriver.h MotorDriver.h Odometry.h Odometry.h

/* Robot Configuration */
#define ROBOT_NAME "hero_"
#define ROBOT_ID "0"

/* WiFi Configuration - use web configuration */
#define ROS_MASTER_IP "10.42.0.1" /* ROS Master IP - Config as a hotspot by default */
#define WIFI_SSID "hero_network" /* Network name */
#define WIFI_PASSWORD "s3cr3tp4ss" /* Network password */
#define ROS_MASTER_PORT 11411 /* ROS Port */
bool default_config = true;

/* Firmware Configuration */
#define DEBUG true /* Debug mode */
#define DEBUG_RATE 1 /* Default debug frequency */
#define SYNC_RATE 20 /* Default loop frequency */

Compiling sketch...
Using cached library dependencies for file: /home/rezeck/Documents/hero_common/hero_firmware/libraries/PID/PID_v1.cpp
Using cached library dependencies for file: /home/rezeck/Documents/hero_common/hero_firmware/libraries/PID_AutoTune_v0/PID_AutoTune_v0.cpp
Generating function prototypes...
/home/rezeck/.arduino15/packages/esp8266/tools/xtensa-lx106-elf-gcc/2.5.8-3-28ed2b9/bin/xtensa-lx106-elf-g++ -D __ets__ -DICACHE_FLASH -I /opt/arduino-1.8.19/tools-builder/ctags/5.8-arduino11/ctags -u --language-force=c++ -f - --c++-kinds=svpf --fields=KSTtzns --lto --lto-partition --lto-partition
Compiling sketch...
/home/rezeck/.arduino15/packages/esp8266/tools/xtensa-lx106-elf-gcc/2.5.8-3-28ed2b9/bin/xtensa-lx106-elf-g++ -D __ets__ -DICACHE_FLASH -I /opt/arduino-1.8.19/tools-builder/ctags/5.8-arduino11/ctags -u --language-force=c++ -f - --c++-kinds=svpf --fields=KSTtzns --lto --lto-partition --lto-partition
1 NodeMCU 1.0 (ESP-12E Module), 160 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Higher Bandwidth, Disabled, None, Only Sketch, 115200
```

 **Attention:** Always confirm battery is disconnected during USB flashing to avoid power conflicts.

Tips:

- If upload fails, try lowering the baud rate (e.g., 57600) or resetting the board.
- Keep a backup of `config.h` with your Wi-Fi and robot ID settings.
- After flashing, open the Serial Monitor at 115200 baud to confirm boot messages.
- Use the firmware's web interface to configure Wi-Fi credentials and dynamic parameters without recompiling.

C. ROS Networking & Bring-up

Reading time: 10 min

Building time: 15 min

In order to control HeRo robots remotely with ROS, you must configure the network parameters so that the ESP8266 connects to your ROS master. This requires setting Wi-Fi credentials and the ROS master IP/port on the robot.

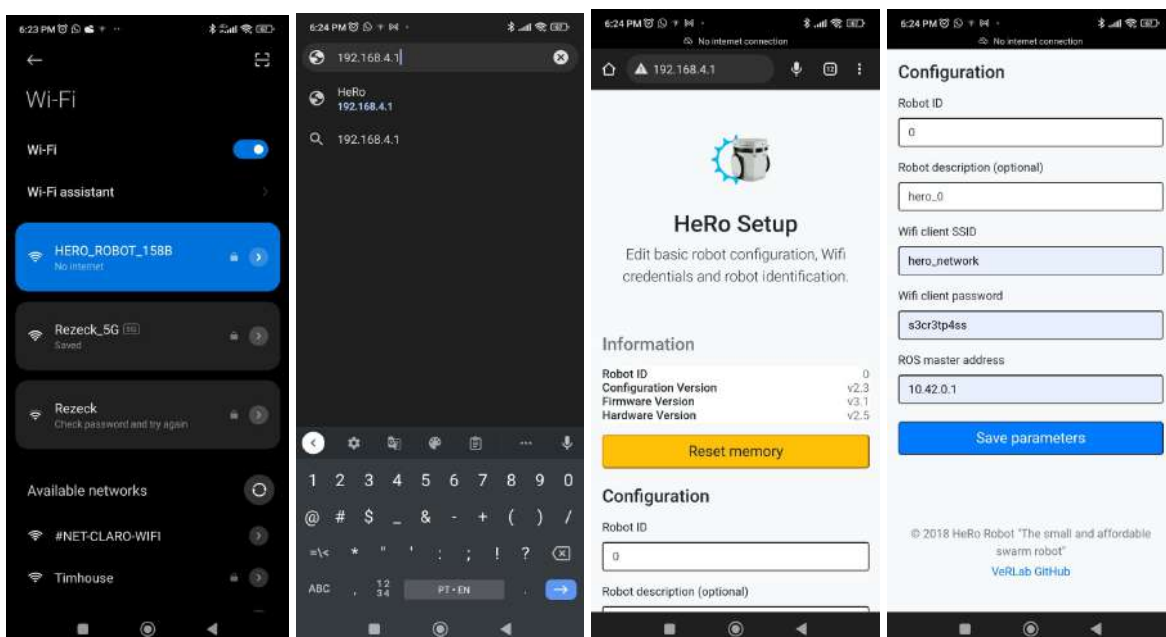
Network Setup

HeRo supports only **2.4 GHz Wi-Fi** networks. You can either create a hotspot on your laptop or use a standard Wi-Fi router. The firmware has two operation modes: **Config Mode** and **ROS Mode**.

- **Entering Config Mode:** Cover all IR sensors and power on HeRo. The RGB LED will flash pink. In this mode the robot creates its own Wi-Fi access point.

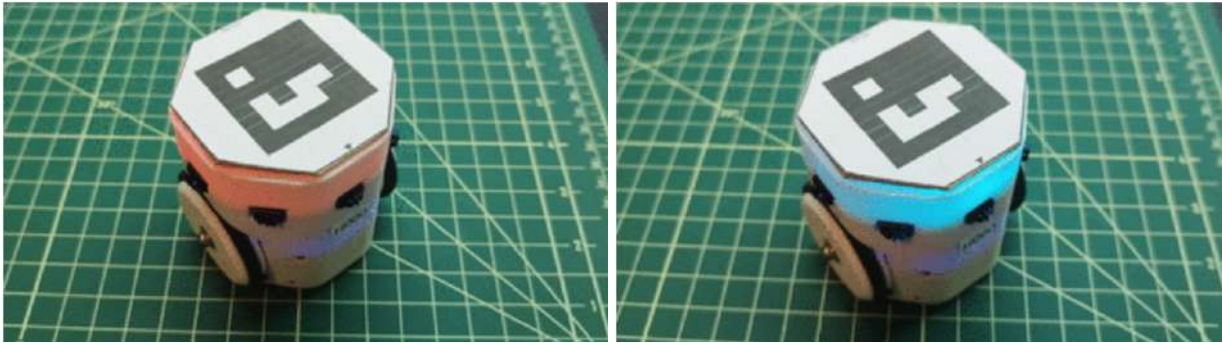


- **Connecting:** On your computer or smartphone, connect to the AP:
 - ESSID: **HERO_ROBOT_*******
 - Password: **s3cr3tp4ss**
- Once connected, open <http://192.168.4.1/> in a browser. Fill in your Wi-Fi SSID, password, and the ROS master IP/port. Save settings, then turn HeRo off.



🚀 Connecting HeRo to Your Network

Power HeRo on normally. If it connects to Wi-Fi successfully, the LED blinks blue briefly. If it fails, it will blink red continuously. Once configured, the robot will auto-connect whenever the network is available. Reconfigure only if you change Wi-Fi credentials or ROS master address.



🚀 Launching ROS Nodes

On the ROS master PC, launch the bring-up file:

\$ roslaunch hero_bringup hero_bringup.launch

```
root@desktop:/catkin_ws# roslaunch hero_bringup hero_bringup.launch
... logging to /root/.ros/log/750fb9d8-2ef8-11ed-8fde-f079596eb8df/roslaunch-desktop-6969.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://desktop:38929/

SUMMARY
=====

PARAMETERS
* /robot_description: <?xml version="1....
* /rostdistro: noetic
* /rosversion: 1.15.14

NODES
/
  roserial_server (roserial_server/socket_node)

auto-starting new master
process[master]: started with pid [6977]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 750fb9d8-2ef8-11ed-8fde-f079596eb8df
process[rosout-1]: started with pid [6998]
started core service [/rosout]
process[roserial_server-2]: started with pid [7001]
[ INFO] [1662588017.326981156]: Listening for roserial TCP connections on port 11411
[ INFO] [1662588051.809615866]: [/hero_0] Firmware: v1.1 | Hardware: v2.5 | Config: v2.3 | IP: 192.168.18.34
[ INFO] [1662588052.165814982]: [/hero_0] IMU hat detected!
[ WARN] [1662588053.582378161]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588612070]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588648494]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588673221]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588693830]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588719775]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.588736173]: Received message with unrecognized topicId (2).
[ WARN] [1662588053.570092174]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.570119566]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.571437495]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.572588462]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.572588487]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.573813570]: Received message with unrecognized topicId (3).
[ WARN] [1662588053.573834215]: Received message with unrecognized topicId (3).
[ INFO] [1662588054.261031596]: [/hero_0] Connected at time 35540
[ INFO] [1662588055.234984363]: [/hero_0] Connected at time 36541
[ INFO] [1662588056.240554635]: [/hero_0] Connected at time 37543
[ INFO] [1662588057.243904932]: [/hero_0] Connected at time 38546
[ INFO] [1662588058.246719092]: [/hero_0] Connected at time 39547
```

Keep this running. Then power on the robot. If successful, you'll see connection logs in the terminal.

📡 Available ROS Topics

Check with `rostopic list`:

- `/hero_0/cmd_motor` – motor PWM commands
- `/hero_0/encoder` – encoder feedback
- `/hero_0/imu` – gyro + accelerometer + orientation

- `/hero_0/laser` – IR distance sensor array
- `/hero_0/led` – RGB LED control
- `/hero_0/odom` – odometry estimation
- `/hero_0/position_controller/cmd_vel` – position controller
- `/hero_0/velocity_controller/cmd_vel` – velocity controller
- `/tf` – HeRo's TF tree

Available Services

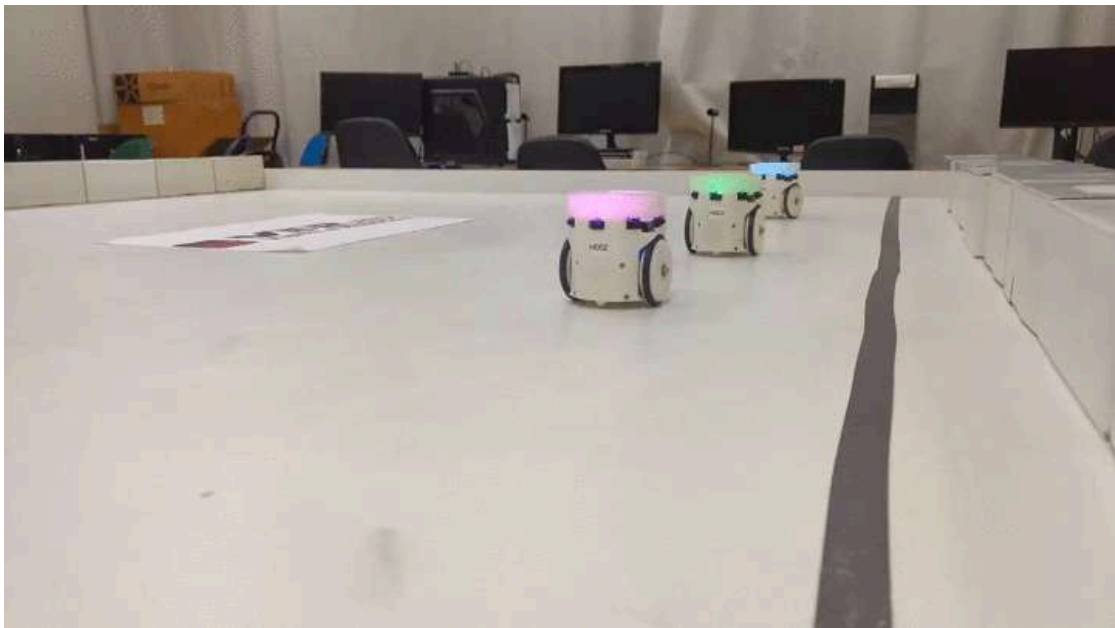
- `/hero_0/pid_calibration`
- `/hero_0/set_frequency`
- `/hero_0/set_odom`
- `/hero_0/set_pid_parameters`


Teleoperation Demo

Open another terminal and run:

```
$ roslaunch hero_bringup hero_teleop.launch id:=0
```

Replace `id:=0` with your robot ID. Use keyboard input to drive the robot around and verify connectivity.



 **Attention:** Ensure your firewall does not block ROS master communication. If using multiple robots, assign unique IDs and confirm each has a unique IP on the network.

Tips:

- Test connectivity with `ping <robot_IP>` before launching ROS nodes.
- If connection drops, check Wi-Fi strength and confirm the robot is within router range.
- Use `rqt_graph` to visualize node/topic connections and confirm the robot is publishing correctly.

D. Robot Simulation (Gazebo – optional)

Reading time: 20 min

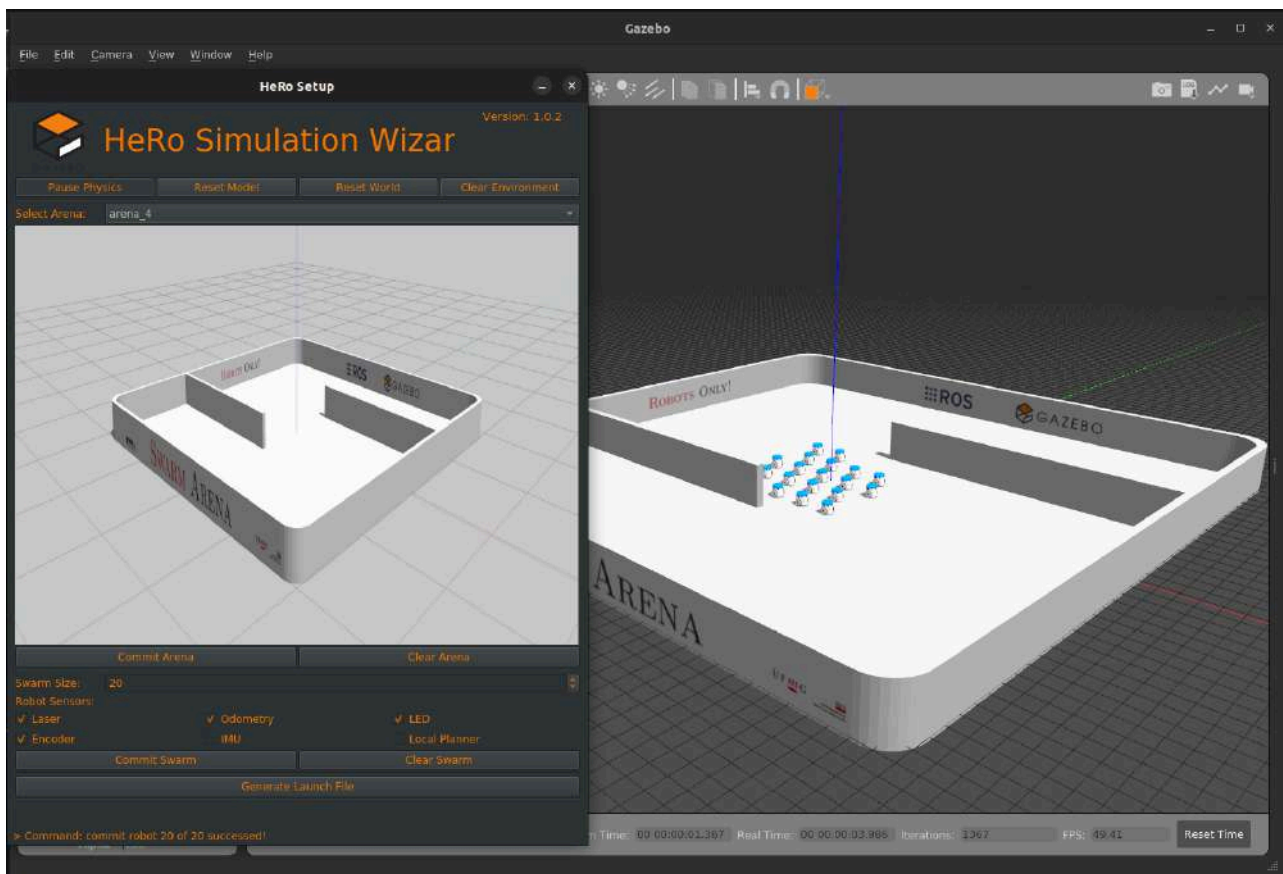
Building time: 30 min

HeRo can be fully simulated in the Gazebo simulator. This is useful for testing algorithms, verifying robot behavior, and developing code before deploying on real hardware.

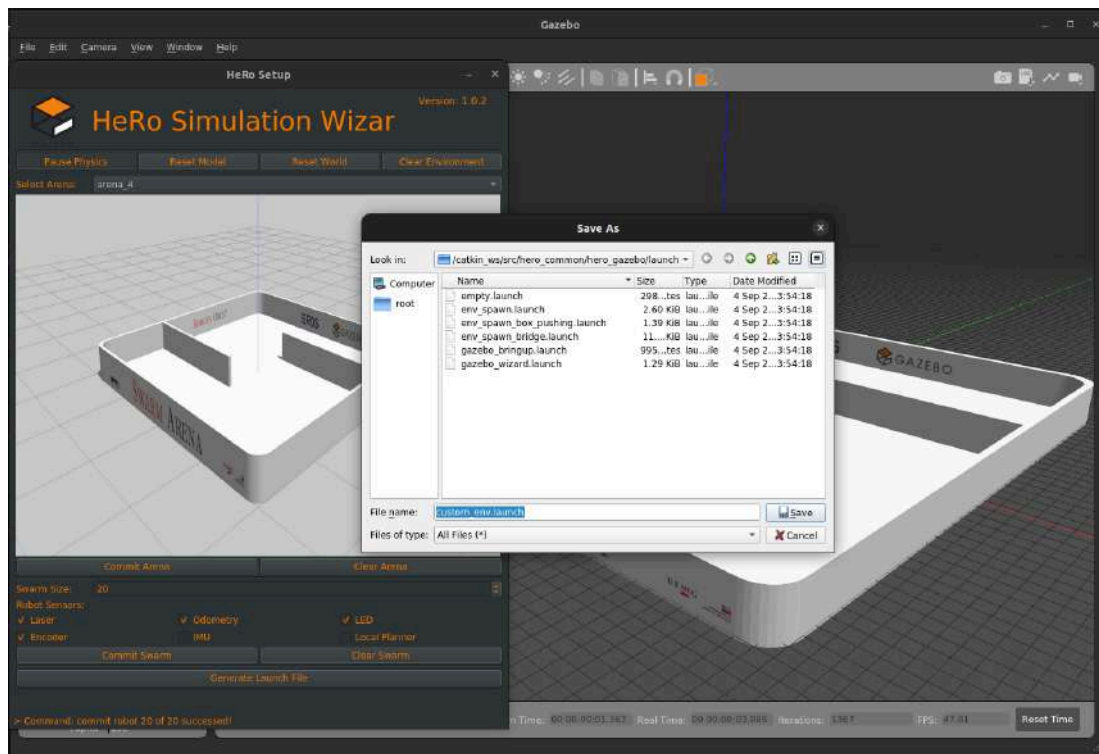
HeRo Wizard Interface

A graphical wizard is available to simplify environment creation and robot spawning:

1. Launch the wizard with:
\$ roslaunch hero_gazebo gazebo_wizard.launch



2. A GUI will appear alongside Gazebo. From here you can:
 - Select an arena from the list.
 - Commit or clear the arena in Gazebo.
 - Choose the number of robots.
 - Enable or disable specific sensors.
 - Commit the swarm (spawn robots) or clear them.
 - Adjust the scene directly in Gazebo.
 - Generate a launch file for reuse with the configured setup.



After setup, you may close Gazebo and the wizard. The generated launch file allows you to recreate the same scenario later without going through the wizard again.

🚀 Using Simulation Without Wizard

If you already have a prepared environment or prefer not to use the wizard:

\$ roslaunch hero_gazebo gazebo_bringup.launch # starts Gazebo

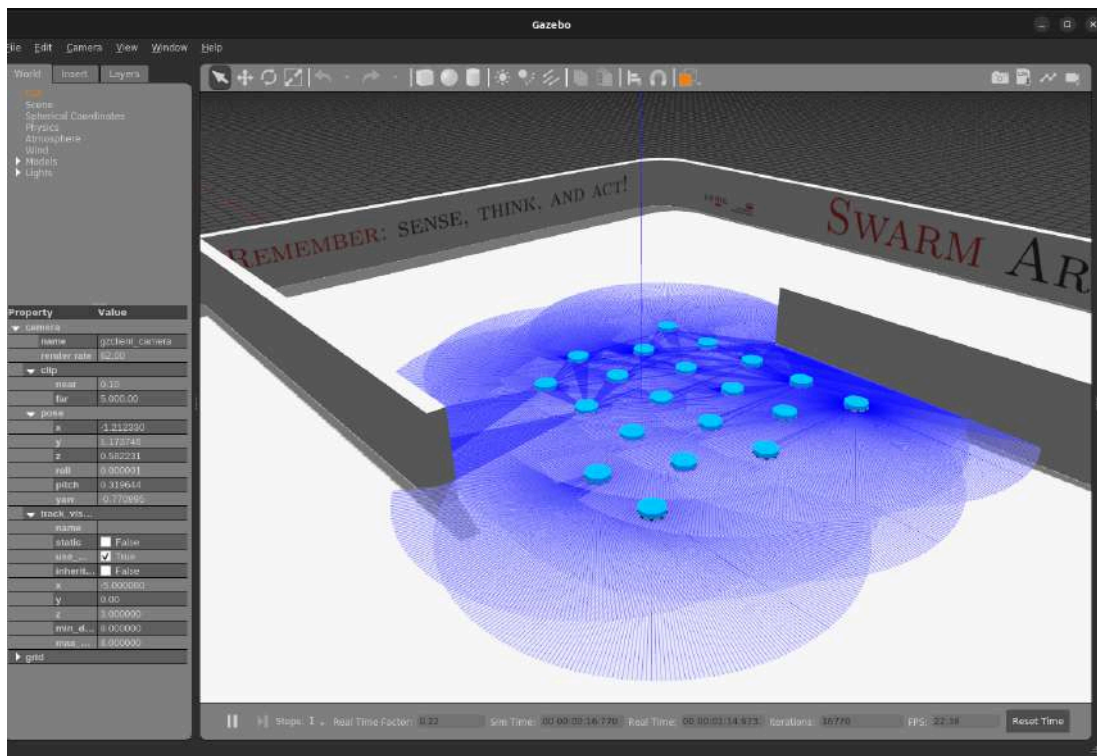
\$ roslaunch hero_gazebo env_spawn.launch # spawns robots + arena

Now the robots are active inside Gazebo. Verify with:

\$ rostopic list

You will see topics such as:

- `/hero_0/cmd_vel`
- `/hero_0/laser`
- `/hero_0/led`
- `/hero_0/odom`
- `/hero_1/cmd_vel`, etc.
- `/joint_states`, `/tf`, `/clock`, and standard Gazebo topics.



⚙️ Features Available in Simulation

- Differential drive controller
- Odometry
- IMU data
- IR proximity sensors (simulated as laser beams)

⚠️ **Attention:** Simulation can be resource-intensive. Ensure your system has adequate CPU/GPU support and avoid running too many robots simultaneously if performance drops.

✨ Tips:

- Use `rqt_graph` to confirm simulated topics are active before running control code.
- Save frequently used scenarios as launch files to avoid repeating setup.
- Test control strategies in simulation before deploying to hardware to minimize risk.
- A short video tutorial is available [here](#).

Part IV – Calibration

Reading time: 25 min

Building time: 60 min

Calibration is essential to ensure that HeRo operates smoothly and predictably. This section details how to configure motor PWM, tune controllers, and calibrate the IR distance sensors.

A. Enter Configuration Mode

Reading time: 5 min

Building time: 5 min

To access calibration mode, launch HeRo in configuration mode instead of the standard bring-up mode:

\$ roslaunch hero__bringup hero__confmode.launch

When in config mode, the robot exposes extra ROS services for calibration. Verify that the expected topics and services appear:

\$ rostopic list

- /diagnostics
- /hero_2/cmd_motor
- /hero_2/encoder
- /hero_2/imu
- /hero_2/laser
- /hero_2/led
- /hero_2/odom
- /hero_2/position_controller/cmd_vel
- /hero_2/velocity_controller/cmd_vel
- /rosout
- /rosout_agg
- /tf

\$ rosservice list

- /hero_2/ir_calibration
- /hero_2/motors_parameters
- /hero_2/position_controller/pid_calibration
- /hero_2/position_controller/pid_parameters
- /hero_2/set_odom
- /hero_2/velocity_controller/pid_calibration
- /hero_2/velocity_controller/pid_parameters
- /hero__bringup/get_loggers
- /hero__bringup/set_logger_level
- /rosout/get_loggers
- /rosout/set_logger_level

B. Motor PWM Calibration

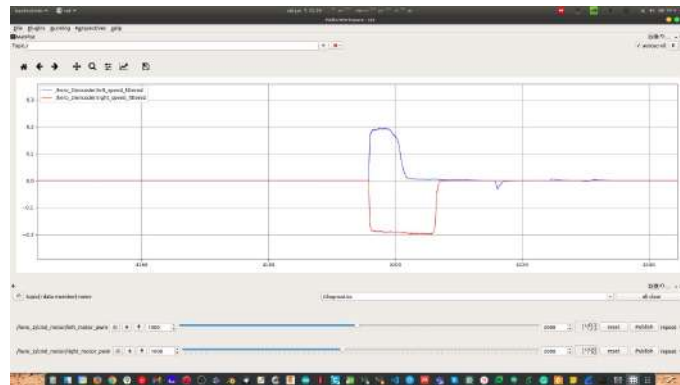
Reading time: 5 min

Building time: 10 min

The SG90 continuous servo motors are controlled with PWM signals. In theory, $1500\ \mu\text{s}$ should stop the motor, while values around $1000\ \mu\text{s}$ and $2000\ \mu\text{s}$ rotate in opposite directions. In practice, the neutral value varies per motor.

Steps:

1. Run RQT and use *Plot* to visualize wheel speed.
2. Use *Easy Message Publisher* to send PWM values incrementally.
3. Find the exact PWM values where each wheel stops without jitter.



Example command to save calibration:

```
$ rosservice call /hero_2/motors_parameters "left_motor_pwm: 1493 right_motor_pwm: 1448"
```

C. Position Controller PID

Reading time: 2 min

Building time: 10 min

After PWM calibration, tune the PID that controls wheel **position**.

```
$ rosservice call /hero_2/position_controller/pid_parameters "{lkp: 5.0, lki: 9.0, lkd: 0.1, rkp: 5.0, rki: 9.0, rkd: 0.1}"
```

These defaults provide good response, but you can fine-tune empirically.

D. Velocity Controller PID

Reading time: 2 min

Building time: 10 min

The velocity controller uses another PID loop for smooth speed control. Example tuning:

```
$ rosservice call /hero_2/velocity_controller/pid_parameters "{lkp: 1200.0, lki: 2300.0, lkd: 0.1, rkp: 1200.0, rki: 2300.0, rkd: 0.1}"
```

Again, start with defaults and adjust slightly if needed.

E. IR Distance Sensor Calibration

Reading time: 10 min

Building time: 20 min

The IR proximity sensors require calibration to provide reliable range estimation. This is done via the ROS service `/hero_*/ir_calibration`.

Manual calibration procedure:

1. Place an object (preferably white) at a known distance d from the IR sensor.
2. Run:

```
$ rostopic echo /hero_*/laser
```

to read the intensity.

[illegible]

3. Compute calibration parameter as:

calibration = intensity $\times d^2$

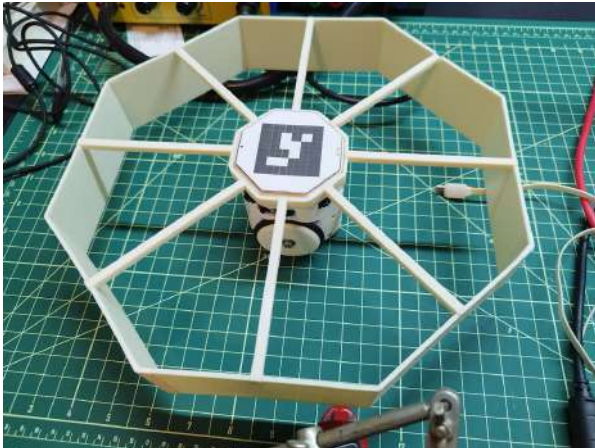
4. Repeat for all 8 sensors.

5. Save values with:

```
$ rosservice call /hero_*/ir_calibration "{IR0: 0.39, IR1: 0.28, IR2: 0.45, IR3: 0.29, IR4: 0.52, IR5: 0.22, IR6: 0.23, IR7: 0.28}"
```

Alternative batch method:

- Place 8 objects at the same distance in front of all sensors.

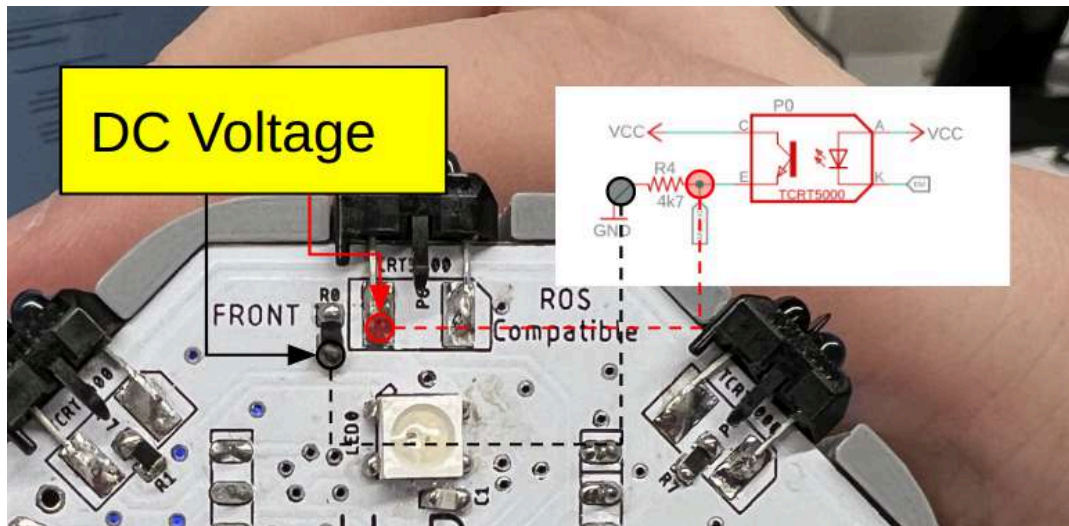


- Use the provided calibration script:
\$ **roslaunch hero_examples hero_ir_calib.py hero_***
- A [3D-printed alignment tool](#) is available to simplify this step.

IR Sensor Test:

If calibration fails, test the hardware:

- Use cellphone camera/webcam to check IR LED emission (should glow purple).
- With a multimeter, measure voltage variation across the phototransistor while covering/uncovering it.



- Replace faulty TCRT5000 sensors if no variation is detected.

⚠ Attention: Wear protective glasses when cutting sensor legs or handling tools. Always keep the robot on a stable surface during calibration.

💡 Tips:

- Perform calibration in consistent lighting conditions.
- Avoid reflective or very dark surfaces during calibration.
- Re-calibrate after hardware changes or sensor replacement.
- Video expected behavior [here](#).
- Video for manual calibration [here](#).
- Video with fallback test using other RangeSensor firmware [here](#).