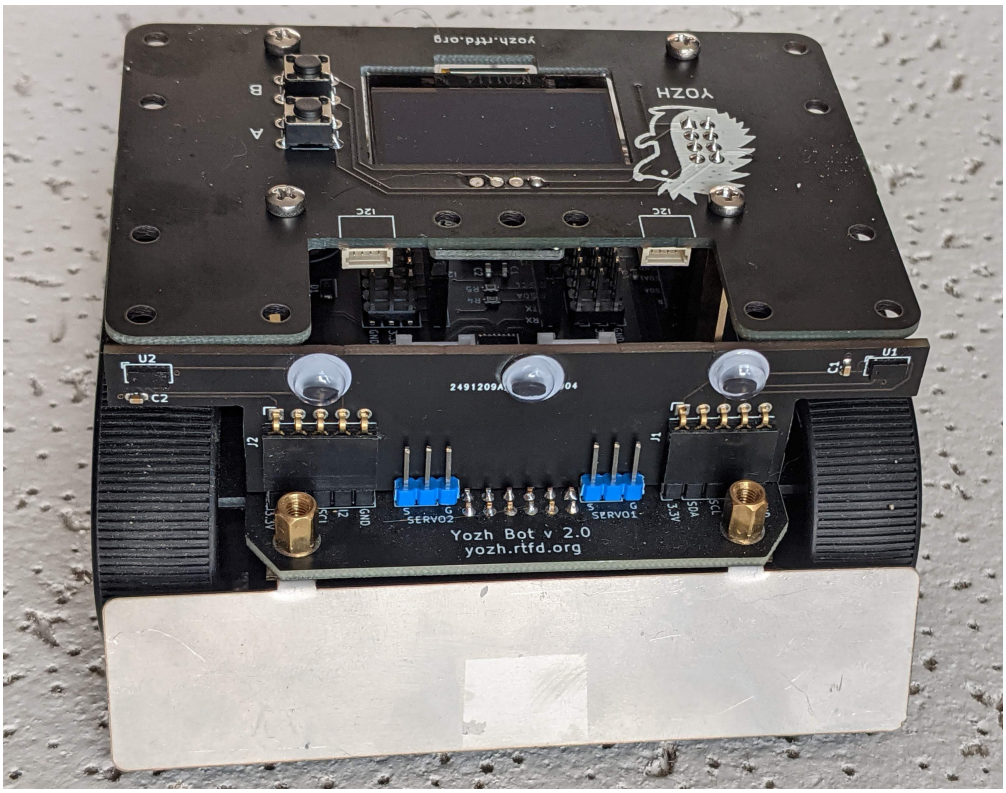
Yozh Robot

Alexander Kirillov

Oct 28, 2023

TABLE OF CONTENTS



Yozh is a small (under 10cm*10cm) robot, based on Pololu's Zumo chassis. It was created by shurik179 for a robotics class at SigmaCamp. Below are the key features of this robot.

The robot consists of the following components:

- Zumo chassis by Pololu
- Power source: 4 AA batteries (NiMH rechargeable batteries recommended)
- Two micro metal gearmotors by Pololu (6V, HP, 75 gear ratio), with motor encoders
- Custom electronics board, containing a slave MCU (SAMD21) preprogrammed with firmware, which takes care of all low-level operations such as counting encoder pulses, controlling the motors using closed-loop PID algorithm to maintain constant speed, and more
- ItsyBitsy RP2040 by Adafruit, which serves as robot brain. It plugs into the main board and is programmed by the user in CircuitPython, using a provided CircuitPython library to communicate with the slave MCU over I2C. This library provides high-level commands such as *move forward by 30cm*
- Included sensors and electronics:
  - Top plate with 128*64 **OLED display** and 2 **buttons** for user interaction
  - Bottom-facing **reflectance array** with 8 sensors, for line-following and other similar tasks
  - Two front-facing **distance sensors**, using VL53L0X laser time-of-flight sensors, for obstacle avoidance
  - A 6 DOF **Inertial Motion Unit** (IMU), which can be used for determining robot orientation in space for precise navigation
  - Two RGB **LEDs** for light indication and a **buzzer** for sound signals
  - Two ports for connecting **servos**

- There are plenty of pins available for connection additional electronics. We also provide several standard connectors for users convenience (Qwiic/Stemma QT connector for I2C devices, Grove connectors)
- Yozh is compatible with mechanical attachments (grabber, forklift,...) by DFRobot.

All robot design is open source, available in github repository under MIT License, free for use by anyone. We also plan to create a Yozh kit which would be sold on Tindie for those who want to build the robot but do not have time or skill to assemble their own PCBs.

You can view photos and videos of Yozh here:

https://photos.app.goo.gl/ERGFzz6CUv6od8WP6

CHAPTER

ONE

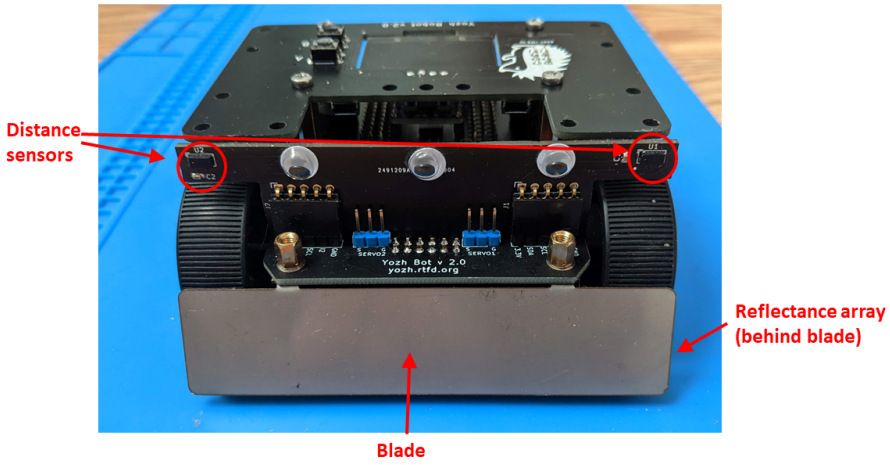
QUICK START GUIDE

If you got the Yozh robot as a kit or are building your own from scratch, please follow the instructions in *Yozh Assembly Guide* to put your robot together.

Once the robot is assembled, follow the steps below to get started quickly.

1.1 Yozh at a glance

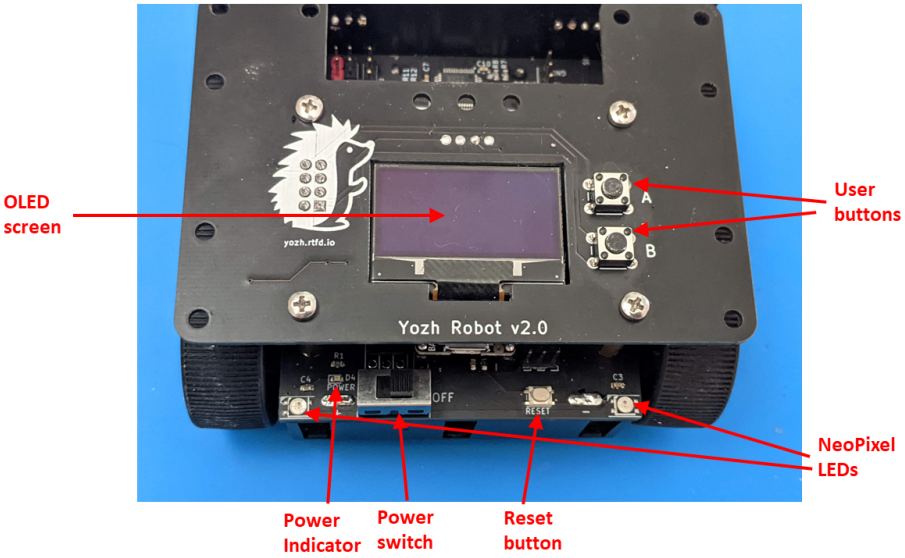
The photos below show main features of Yozh:

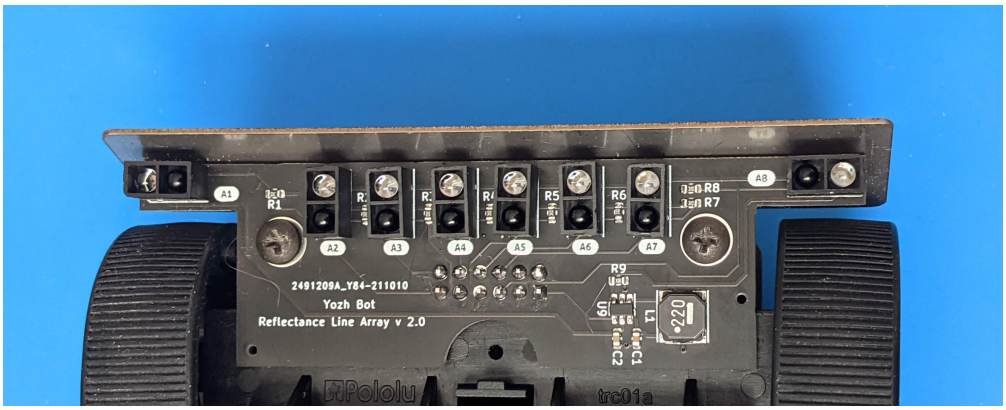


1.2 Circuit Python library installation

Yozh is intended to be programmed in CircuitPython 7 - an implementation of Python programming language for microcontrollers, created by Adafruit (based on Micropython, another Python implementation). For general background on Circuit Python, please visit What is CircuitPython? page.

You must have already installed CircuitPython on the ItsyBitsy microcontroller serving as the brains of Yozh robot during assembly. If not, please do so now following Adafruit's instructions (you will need to remove the top plate to access the BOOTSEL button of ItsyBitsy).


OLED
screen
User
buttons
A
B
yozh.rtfd.io
Yozh Robot v2.0
OFF
RESET
NeoPixel
LEDs
Power
Indicator
Power
switch
Reset
button


2491209A_Y84-211010
Yozh Bot
Reflectance Line Array v 2.0
Pololu

Next step is installing some CircuitPython libraries. Download the Adafruit's library bundle as a zip file from https://circuitpython.org/libraries. Unzip the archive to your hard drive.

Connect Yozh robot to the computer using a micro USB cable. (It should be plugged into the USB port on the ItsyBitsy RP2040.) It should appear in your file browser as an external drive with the name *CIRCUITPY*. Open it to view contents. It might contain a folder named *lib*; if not, create one. Now, install the necessary libraries by copying the following files and folders from Adafruit Circuit Python bundle to *lib* folder in *CIRCUITPY* drive:

- *adafruit_bus_device*
- *adafruit_register*
- *adafruit_display_text*
- *adafruit_displayio_ssd1306.mpy*
- *simpleio.mpy*
- *adafruit_vl53l0x.mpy*

Next, you need to install Yozh Circuit Python library. Go to github repository and click on green `Code` button to download the zip file containing all Yozh designs and software. Extract the zip file to your computer. Find in the extracted archive file *python_library/yozh.py* and copy this file to *lib* folder. Now, you have all the libraries you need.

Please note that extracted Yozh archive also contains a folder *python_library/examples*. Move this folder to some convenient location on your computer - you will use it shortly.

1.3 Mu editor

We suggest using Mu editor for creating and editing programs for your robot. Please follow the instructions here to install Mu editor on your computer.

Experienced programmers can use their favorite text editor instead - but please check this page for some common problems and list of recommended editors.

To verify your installation, connect Yozh to the computer using a USB cable (using the USB connector of ItsyBitsy board) and the start Mu editor. It should ask you to select mode; please choose *CircuitPython*. You are now ready to run your first program for the robot!

1.4 First program

The *CIRCUITPY* drive which was created during installation of CircuitPython on ItsyBitsy RP2040 contains a special file, *code.py*. This file always contains the code of the current program running on the board. As soon as you turn the robot on or hit reset, the robot starts executing this program.

If you edit this file with Mu editor (or any other editor), the robot automatically restarts code execution as soon as the file is saved. You **do not** need to hit reset button or disconnect the robot from the computer.

It also means that if you write a program that involves the robot moving, you should put some code that waits for press of a button in your program - otherwise, your robot will start motion while still connected by USB cable to the computer, which is probably not what you want.

To test the robot, connect it to the computer, using the microUSB connector of the ItsyBitsy RP2040. Make sure the robot switch is in ON position. Start the Mu editor and open *code.py* file in CircuitPython drive, using **Load** button. Erase everything in that file so it is blank.

Now, find the folder with examples from Yozh library you downloaded previously. In that folder, find the file *basic_test.py* and open it in another tab of Mu editor, again using **Load** button. Copy the whole contents of *basic_test.py*

file and then paste it in *code.py* file. (Unfortunately, Mu doesn't have *Save as* command, so you must use copy-and-paste.)

Now save *code.py* file and your robot will execute your first program! Look at the OLED screen, read the prompts, press the buttons, and have fun.

The code is amply commented, so it is easy to make changes. Try modifying the code (e.g. changing the text printed to screen) and then re-save it.

1.5 Serial console

For debugging the program, one needs to print some information such as variable values and error messages. Python has built-in command *print()* which does that. The output of print command is sent to *serial console* - which in practice just means that it is sent over USB to the computer.

Mu editor has built-in serial console, so you can see these messages. To enable the console, click on **Serial** icon in the toolbar. The serial console will appear at the bottom of the screen.

After activating the serial console, you will probably want to restart the program. The easiest way to do that is by clicking **Save** again, even if you didn't make any changes. This will cause the file to be re-saved and program execution restarted.

For more information about using serial console please check Adafruit documentation: https://learn.adafruit.com/adafruit-itsybitsy-rp2040/connecting-to-the-serial-console . Among other features it provides is the ability to enter Python commands interactively in the console, without saving them to a file - this is very useful for testing various things. This is called REPL (Read-Evaluate-Print Loop); see https://learn.adafruit.com/adafruit-itsybitsy-rp2040/the-repl for more info.

1.6 More examples

Now that you have learned how to write and save programs to the robot, it is time to explore Yozh capabilities. To help with that, we have provided a number of examples, which can be found in *examples* folder of the Yozh library you had downloaded previously. Try opening and running them to see what the robot can do.

Below is the list of provided examples (as of Jan 8, 2022):

- *basic_test.py* - basic test of robot operation, including OLED display, LEDs, and buttons
- *motor_test.py* - testing basic operation of motors and encoders
- *servo_test.py* - testing servos (if you have any attached).
- *imu_test.py* - testing IMU
- *pid_test.py* - testing PID control of motors
- *drive_test.py* - testing higher-level drive commands, such as *go forward for 10cm* or *turn 90 degrees*
- *distancesensors_test.py* - testing operation of front-facing distance sensors
- *linearray_test.py* - testing reflectance sensor array
- *line_following.py* - following a line (requires a black line 1-3 cm wide, on white background)
- *obstacle_avoidance.py* - moving around and using distance sensors to avoid obstacles

All of these examples are amply commented, so it should be easy to understand how the code works and how to modify it.

1.7 Next steps

These examples give you some idea of what Yozh is capable for. But if you need to go deeper, check *Library guide* for full list of available commands, and *Yozh feature description* for a detailed description of Yozh hardware and specs.

And if you have any questions or comments, please reach out to us at irobotics.store@gmail.com.

CHAPTER

TWO

YOZH ASSEMBLY GUIDE

This document describes the assembly of Yozh robot.

2.1 Parts and materials

To build Yozh Robot, you will need the following parts and materials:

- Zumo chassis by Pololu
- Two 6V N20 size micro gear motors with extended shafts. We recommend these motors from Pololu
- Magnetic encoder disk fitting on the extended shaft of the motor. (You only need two, but Pololu sells them in packs of 5). Note that you do not need to buy encoders, only the disks - encoder sensors are built into the main Yozh board.
- Yozh robot kit, available from Tindie (SOON - link coming). Contents of this kit, together with alternatives for those who prefer to build your own, is described below.
- Four AA batteries. Usual alkaline batteries will work but won't last long, so use of heavy-duty NiMH rechargeable batteries such as Panasonic Eneloop Pro is strongly recommended. You will also need a charger for these batteries.

2.1.1 Yozh kit contents

Below is the contents of the Yozh kit, together with alternative sources for buying these parts if for some reason you prefer not to use the kit.

- Itsy Bitsy RP2040 microcontroller (available from Adafruit) and male headers
- Five custom PCBs:
  - Main electronics board
  - Spacer board
  - Reflectance sensor array board
  - Top cover with OLED display
  - Front distance sensor board

  Design files and BOM for these boards are available from Yozh github repository.
- Steel front blade to be attached to the robot. The blade is made of 0.030" stainless steel; DXF file for the blade is in github repository. You can use Pololu's lasercutting services to cut the blade for you.
- Mounting hardware:

  - Four 22mm M2.5 female brass standoffs and 8 M2.5 screws (8mm length), for mounting the top plate
  - Two 15mm M3 brass female standoffs and two M3 screws, for attaching the reflectance sensor board
  - Two 6mm M3 M-F standoffs. Note: it is required that the male thread length be at least 6mm, which is somewhat unusual. You can find such standoffs on AliExpress, e.g. here: https://www.aliexpress.com/item/32872847199.html?spm=a2g0s.9042311.0.0.27424c4dmfu9xI (choose option **M3 (thread 8mm)**, and length 6mm).
- Two long (15mm) 4-pin male headers

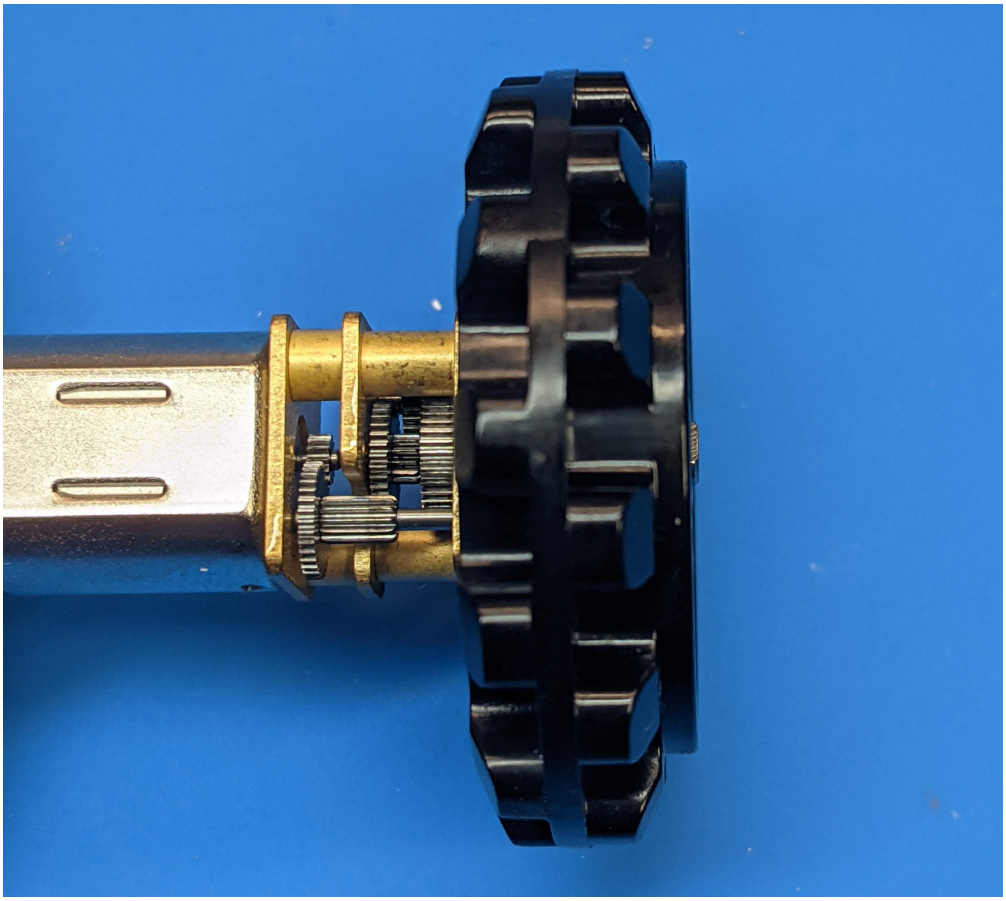
2.1.2 Tools

You will also need to have a computer to program the robot, and some basic tools: soldering iron, wire strippers, flush cutters, screwdriver, pliers.

2.2 Preparing motors

Prepare four pieces of solid core 22AWG wire. Each piece needs to be about 5cm long, about half of which should be stripped. (It is easiest to first strip 2.5cm from a roll of wire and then cut.) Use pliers to form a short (about 3mm) 90 degree bend at **one** end of each of these pieces of wire, as shown in the photo below.

Now, take the motors. Attach the sprocket to each motor, paying attention to orientation: the side of the sprocket with the "lip" should be facing outwards.



Place the motors on the table, using sprockets as supports, and solder the motor leads to each motor. (You probably want to use a third hand or similar device to support the wires while soldering.)

Important: pay attention to proper orientation of the motors! One of motor terminals is marked +; make sure that this terminal is on the right side as shown in the photos.

After soldering the leads, cut off the unstripped part of each motor lead - it was only needed to make it easier to hold the wire while soldering.

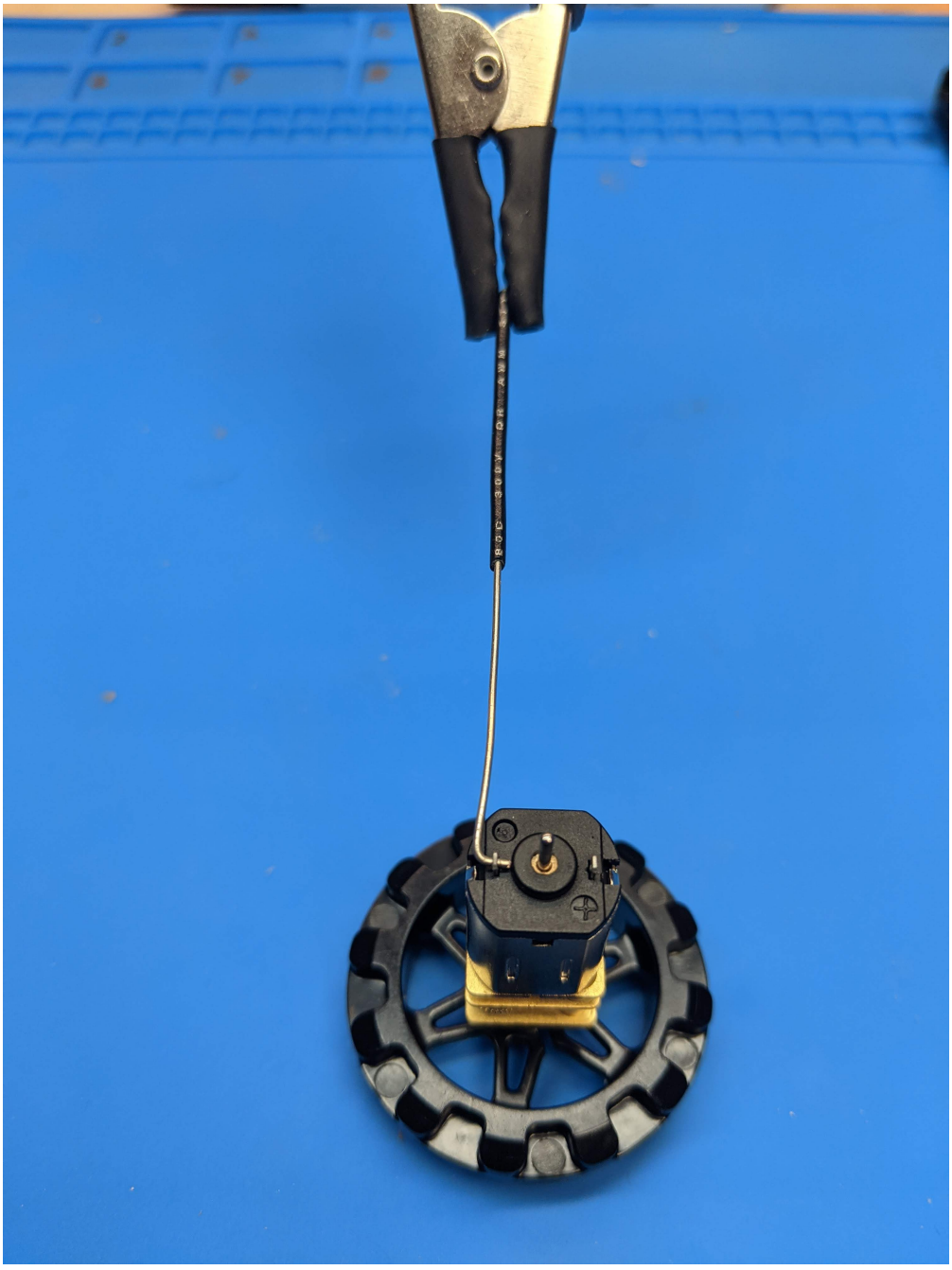
Finally, put the magnetic encoder disks on the rear shafts of the motors.

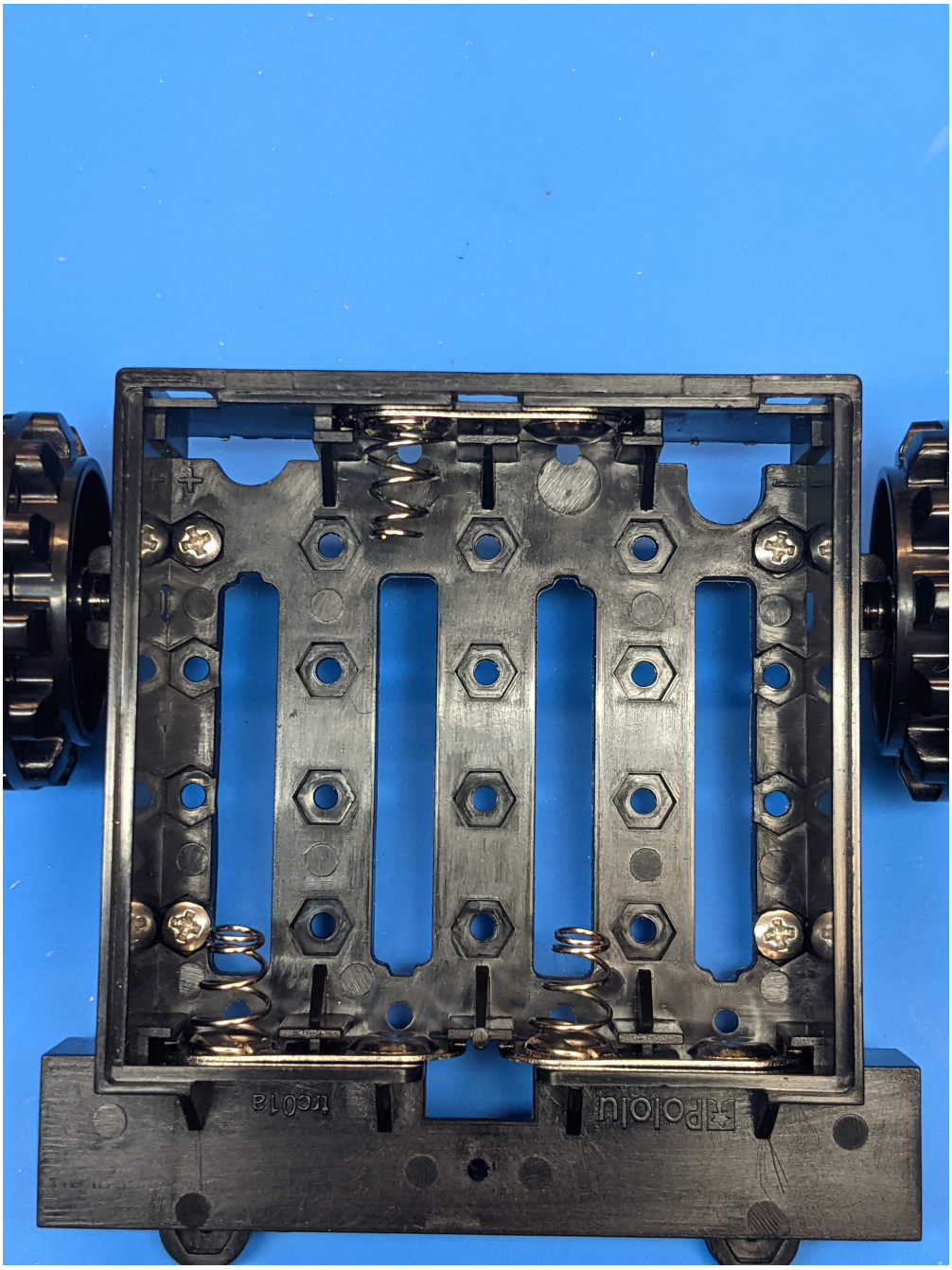
2.3 Assembling the chassis

Take the Zumo chassis kit and open it. Make sure you have all components as listed in Zumo chassis user guide. Discard the acrylic plate - we wont' be using it.

Now follow these steps:

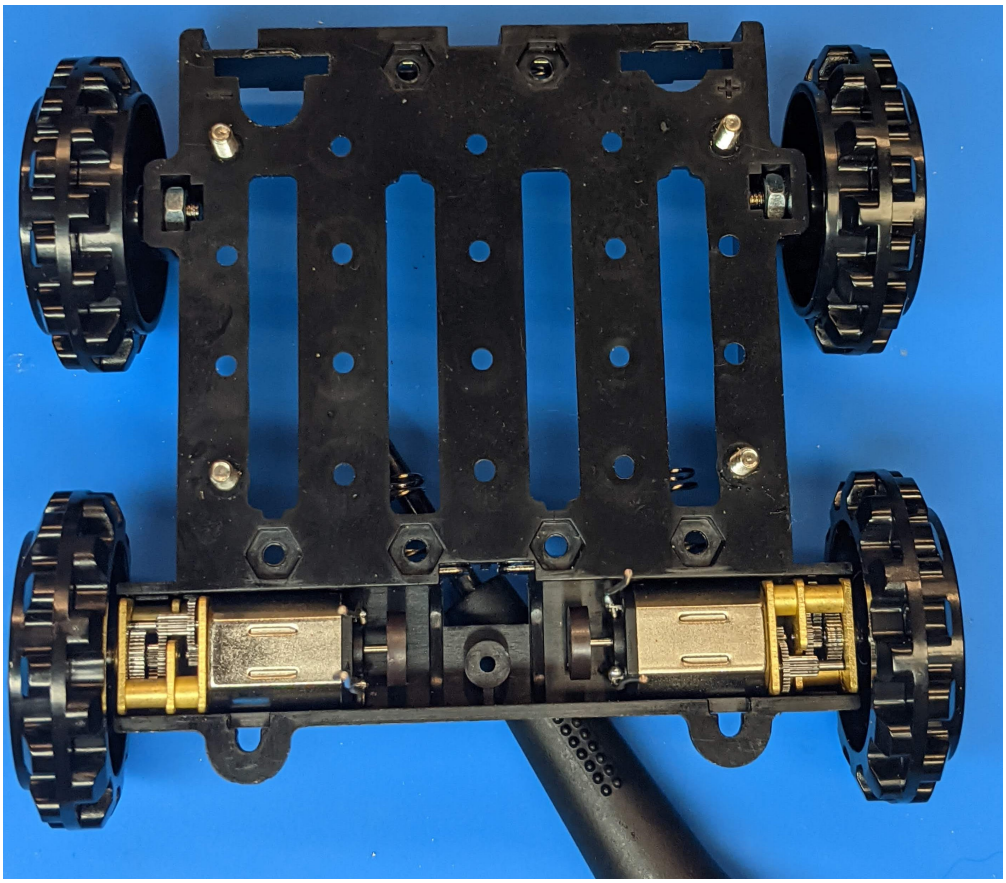
1. Turn the chassis upside down. Remove the cover from the battery compartment of the chassis and insert the double battery contacts in the slots as shown in the photo below, paying attention to orientation. Leave two single contacts for later.
2. Insert 4 M2.5 screws from Yozh kit in the holes as shown in the photo below. It should be a tight fit – use a screwdriver to drive the screws all the way in.

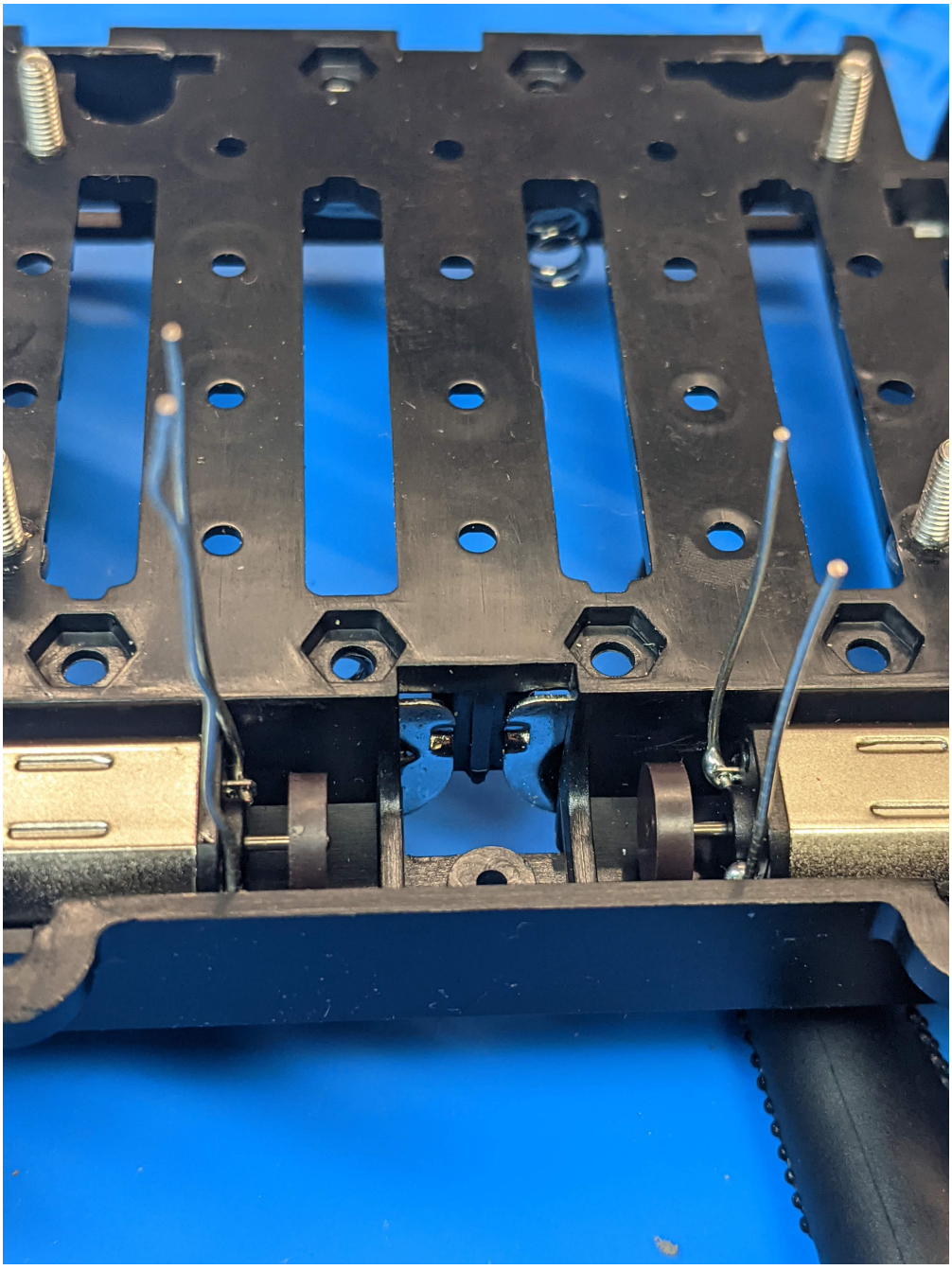


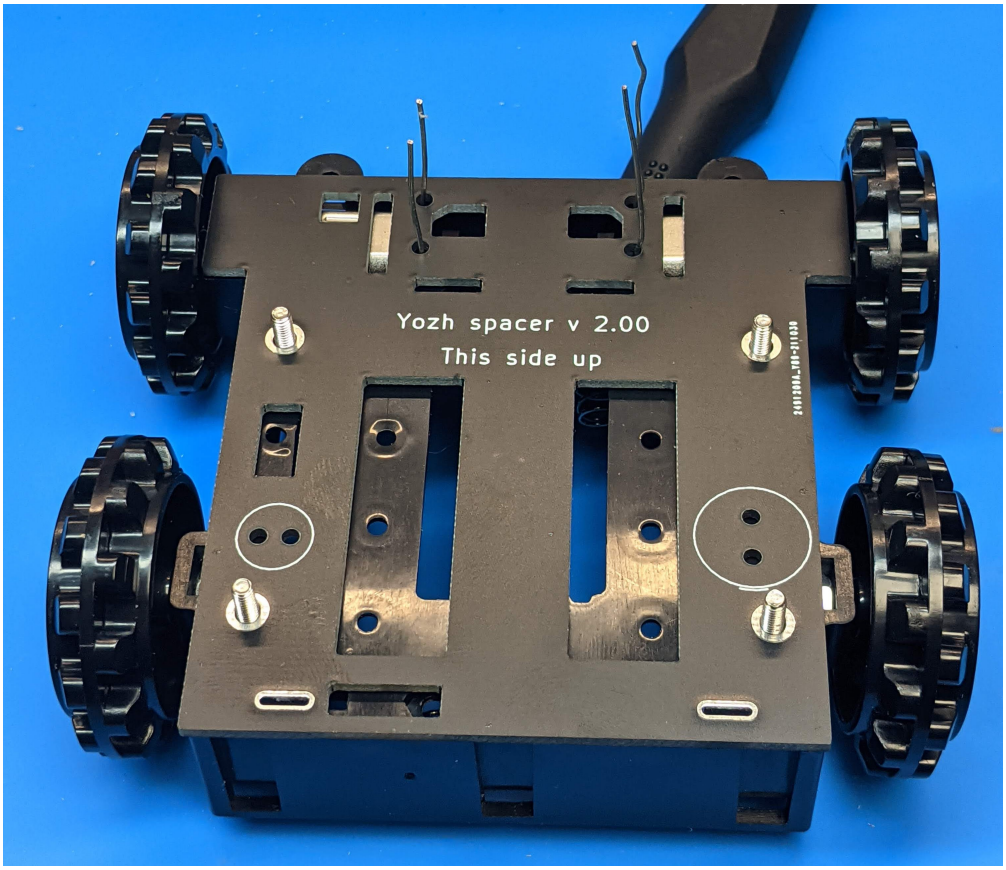


3. Turn the chassis right side up. Attach the idler sprockets following the directions of Zumo user guide.
4. Insert the motors with leads in the channel at the front of the chassis, aligning the gearbox with the grooves in the channel. The front plate of the gearbox should be even with the edge of the chassis. The motor leads you had soldered previously should be facing up.

   (Tip: place a pencil under the center of the chassis)
5. Take the spacer PCB and place it on the chassis, carefully threading the motor leads through the holes in PCB. The screws you had inserted should match the holes in the spacer PCB. **Warning**: spacer PCB is not symmetric - pay attention to **This side up** marking.
6. Repeat the same with the main board. Make sure that this board lies flat against the spacer board, with no gaps







anywhere.

7. Tightly screw the 22mm M2.5 standoffs from the Yozh kit onto the screws protruding through the main board.
8. Put the silicone tracks on wheels (requires a little effort).

2.4 Soldering the chassis

Before continuing, make sure that power switch on the main board is in **OFF** position.

Using the soldering iron, solder the motor leads protruding through the main board to the board. Be careful with nearby connectors - try to avoid touching them with your iron. Use flush cutters to trim remaining length of motor leads.

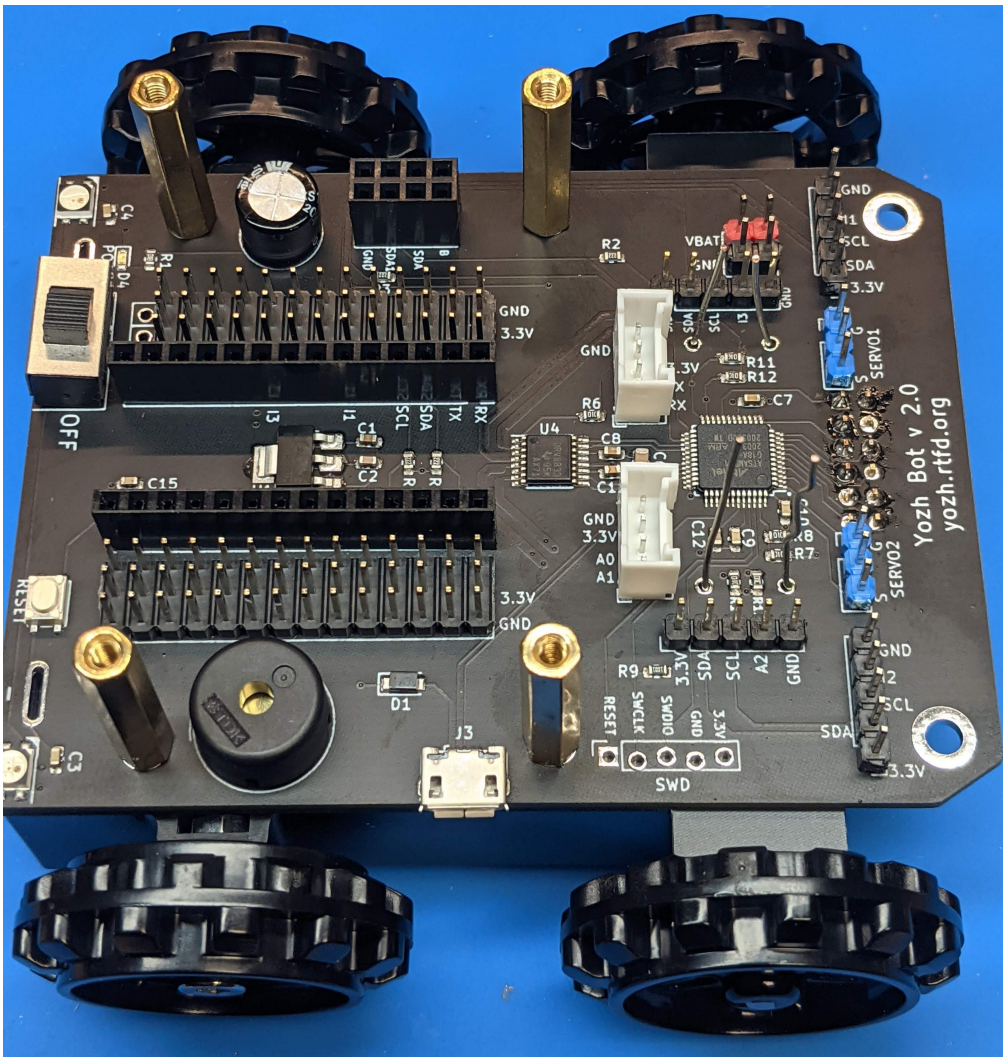
Now, turn the chassis over and insert the two individual battery contacts as shown in ` this drawing <https://a.pololu-files.com/picture/0J9740.1200.png>`__ by Pololu. The solder tabs of the two individual contacts should go through the holes in battery compartment and into the slots in the spacer board and the main board.

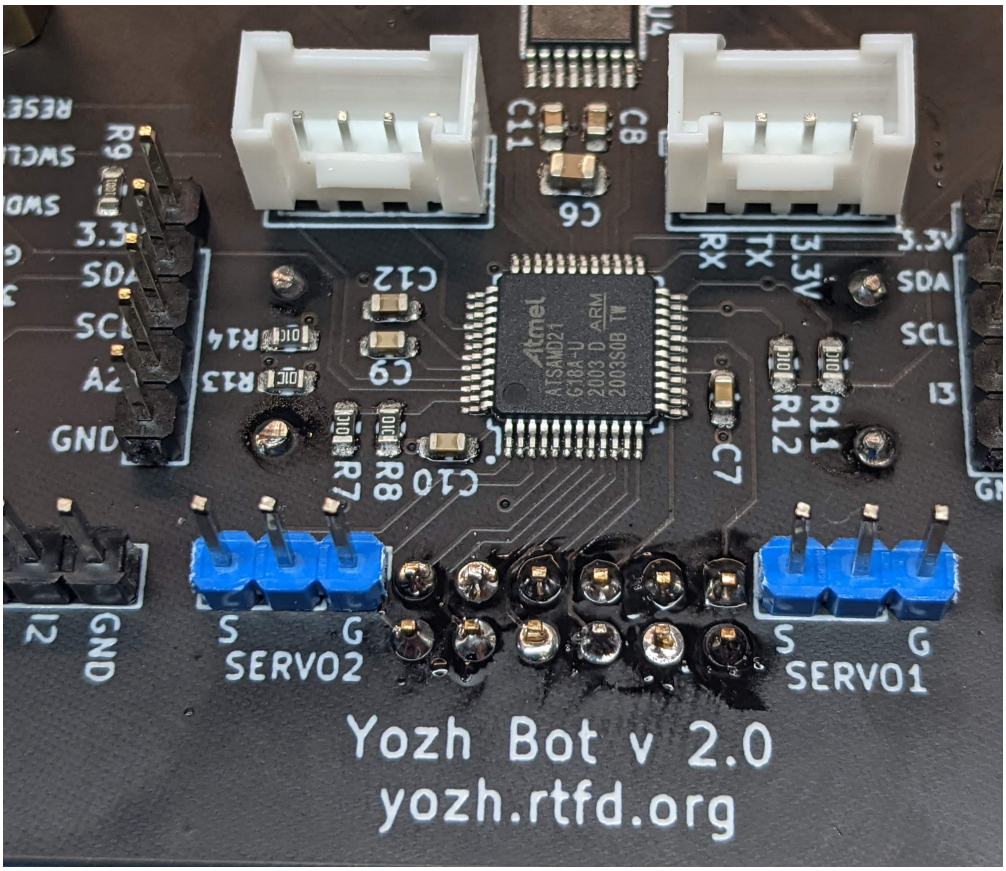
Insert 4 AA batteries (making sure they are in correct polarity) to hold the battery terminals in place until you solder them and close the battery compartment cover. Turn the chassis over again and inspect the battery terminals; the top of solder tab should be level with the top of the main board. If necessary, adjust the battery terminals.

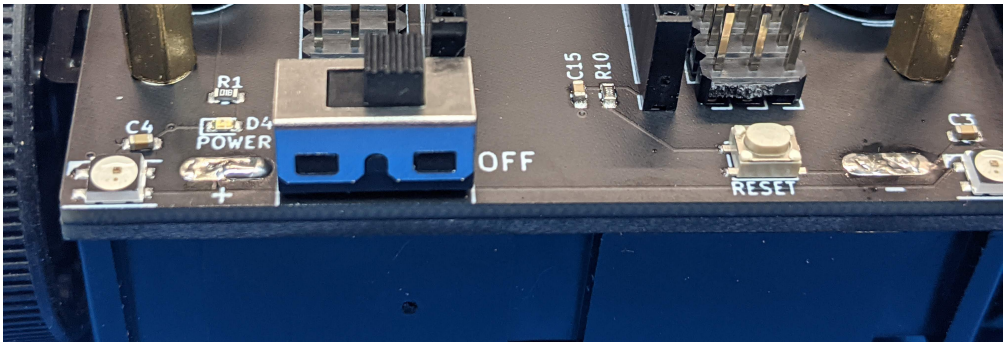
Solder the tabs of the battery contacts to the main board. The easiest way to do that is by using a larger iron tip, heating the plating of the slot in the main board, and then filling it with solder.

Once the solder cools, turn the switch on. The LEDs on the robot should light up, indicating that the robot is powered.

Turn the robot off again.


Yozh Bot v 2.0
yozh.rtfd.org
OFF
RESET
SWD
SERVO1
SERVO2
GND
3.3V
SDA
SCL
VBAT
U4
D1
J3

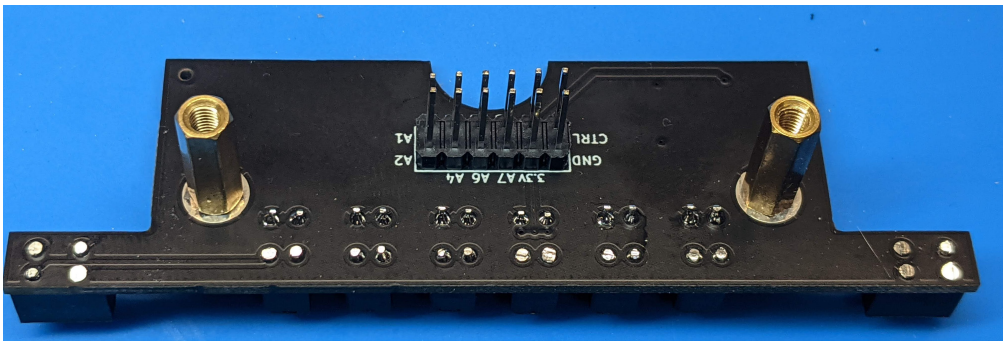

Yozh Bot v 2.0
yozh.rtfd.org
SERVO2
SERVO1
GND
3.3V
SDA
SCL
TX
RX


POWER
OFF
RESET

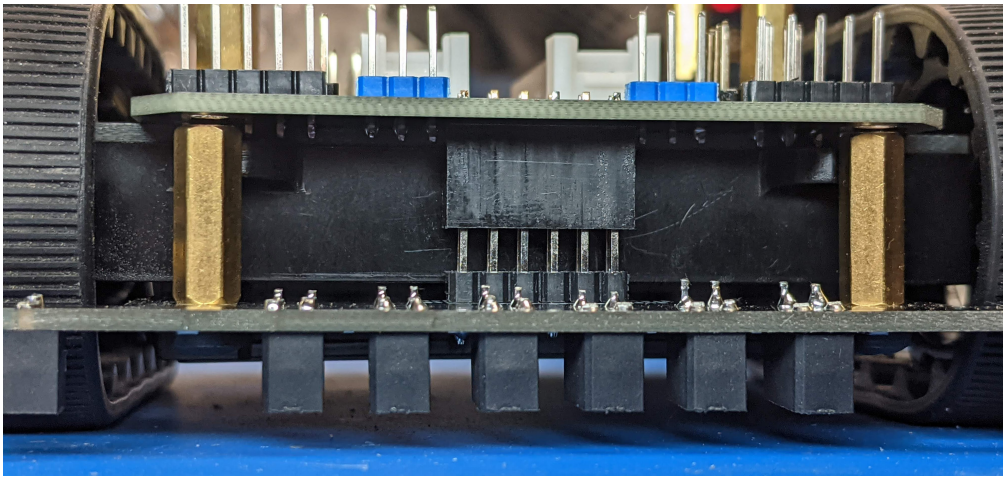
2.5 Reflectance array and the blade

Next step is attaching the reflectance sensor array to the front of the robot.

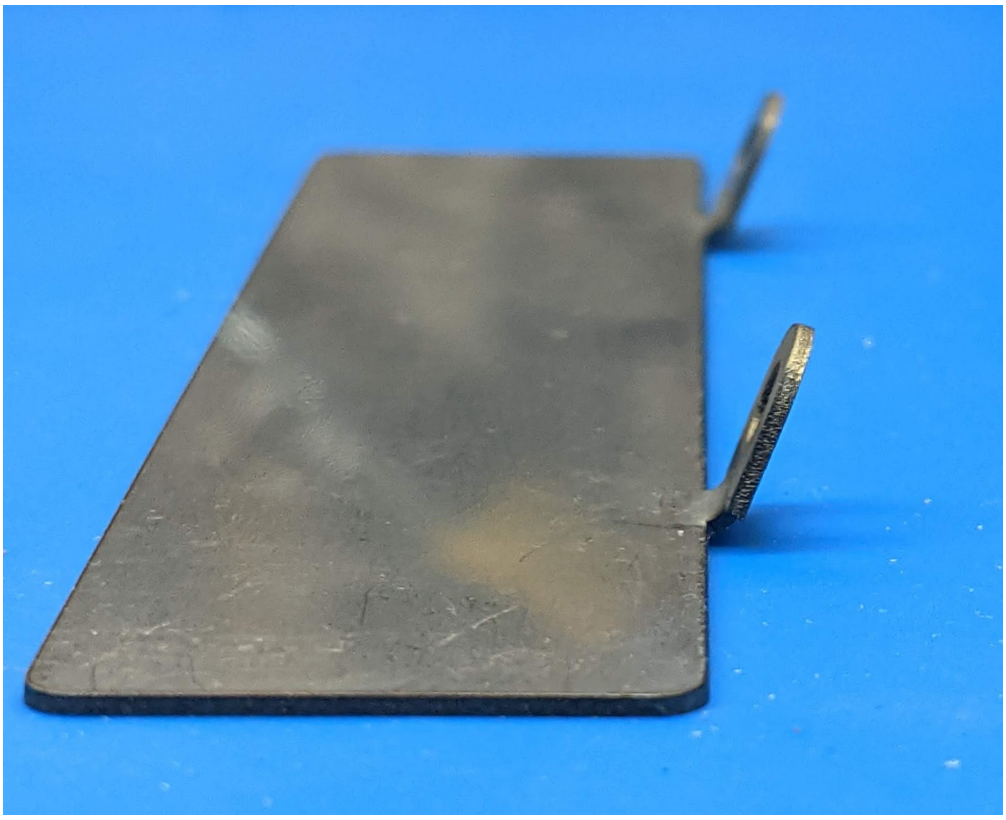
1. Get the reflectance sensor array and inspect the pins of the soldered header. If they are bent, try to gently straighten them.
2. Use M3 screws to attach the two 14mm M3 standoffs to the reflectance sensor array as shown in the photo.

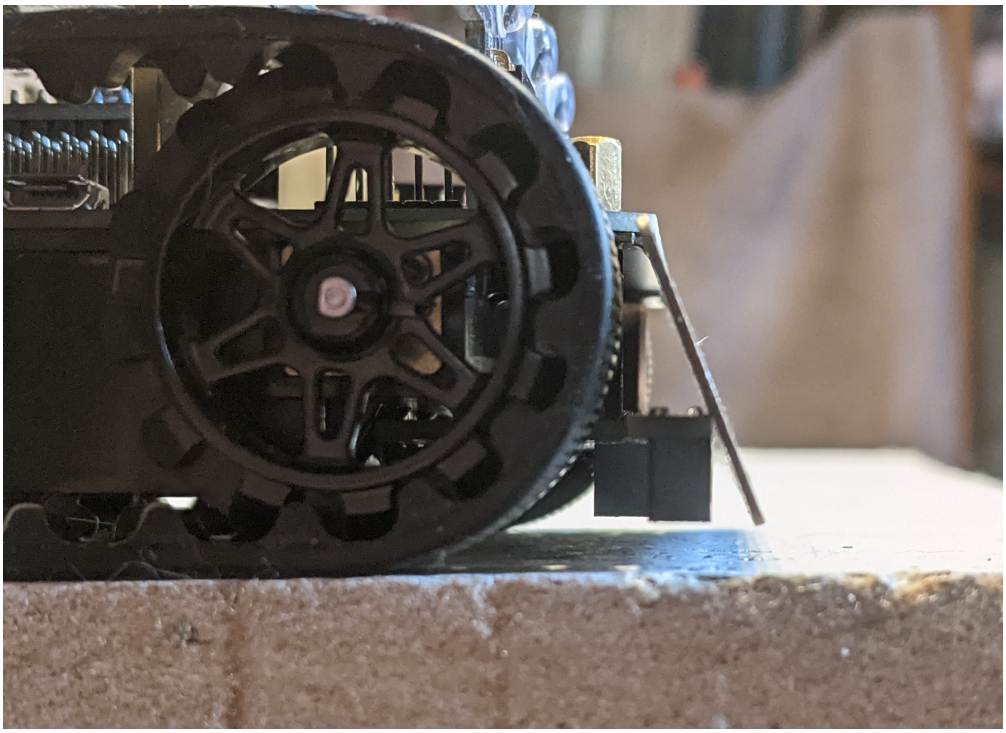


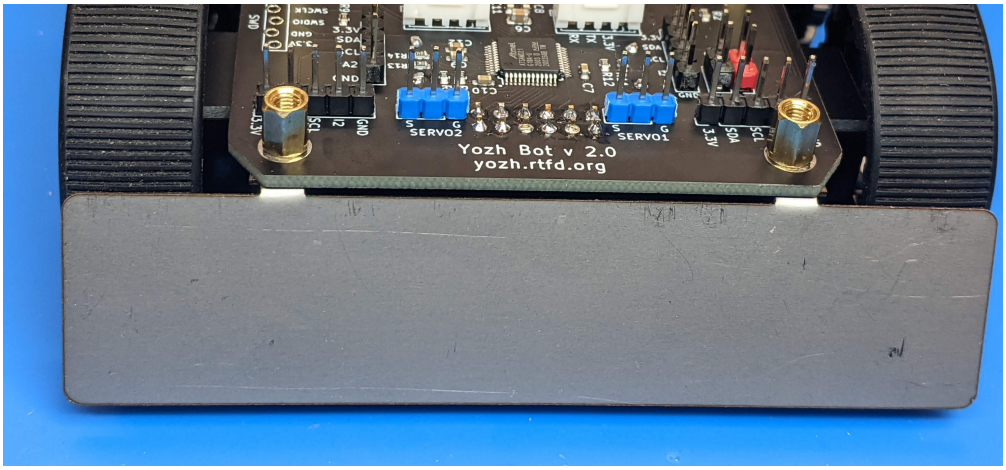
3. Plug the reflectance array into the female header at the bottom of the robot main board. Pull it slightly out so that there is a small gap (about 1mm) between the top of the standoff and the bottom of the main board.



4. Get the blade. Using pliers, bend the two tabs to about 70 degree angle.
5. Insert the blade so that the tabs fit in the gap between the main board and the standoffs. If you bent it to the correct angle, the front of the array will be touching the blade. If necessary, remove the blade and use pliers again to adjust the angle.
6. Insert the short M-F M3 standoffs in the holes at the top of main board, through the hole in the blade tabs and into the long M3 standoff. Tighten by hand.







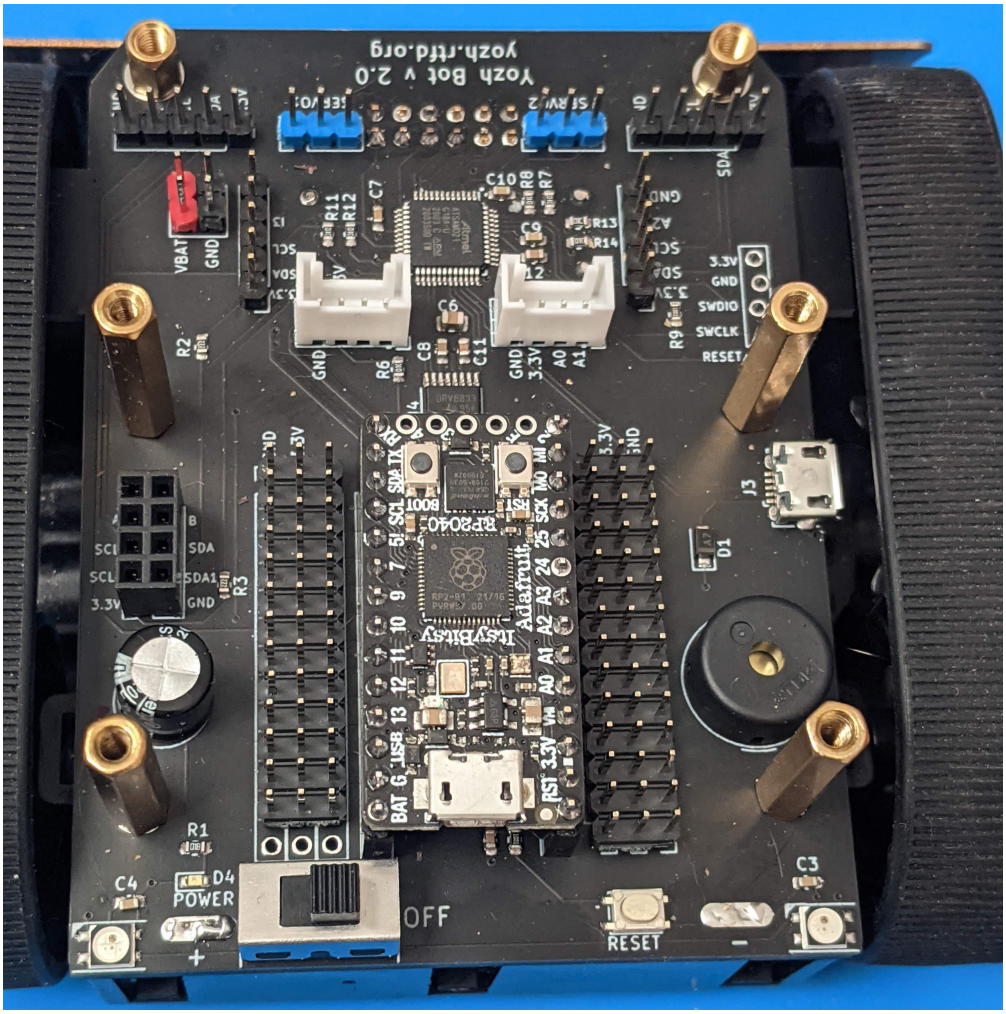
2.6 Itsy Bitsy and the top board

Before proceeding, make sure that the main switch on Yozh robot is in OFF position.

1. Install CircuitPython 7 on your Itsy Bitsy RP2040 microcontroller as described in Adafruit Learn Guide.
2. Solder two 14-pin male headers to the Itsy Bitsy board. If you need help with that, check this guide - it is about a different board, but the process is the same, except that Itsy Bitsy uses two 14-pin headers. **Do not solder headers to the short edge!**
3. Insert the Itsy Bitsy board into the female headers on the main Yozh board, with the USB connector facing out. Double-check that it is plugged correctly - it is easy to make a mistake, shifting the board by one position.
4. Insert the two 4-pin 15mm male headers into the 2*4 female header on the main Yozh board.
5. Take the top plate board, with the OLED display, and place it on top of the robot so that the male headers from the previous step are plugged into the 2*4 female header on the bottom of the top plate. Double-check that it is plugged correctly by looking from the sides. Now double-check again.
6. Use the M2.5 screws to attach the top plate to the standoffs on the Yozh robot.
7. Plug the front distance sensor board into male headers at the front of the robot, with the sensors facing out.
8. (optional) Attach googly eyes!

2.7 This is it!

Your robot is now complete. Check the *Quickstart guide* to start programming and using your Yozh!

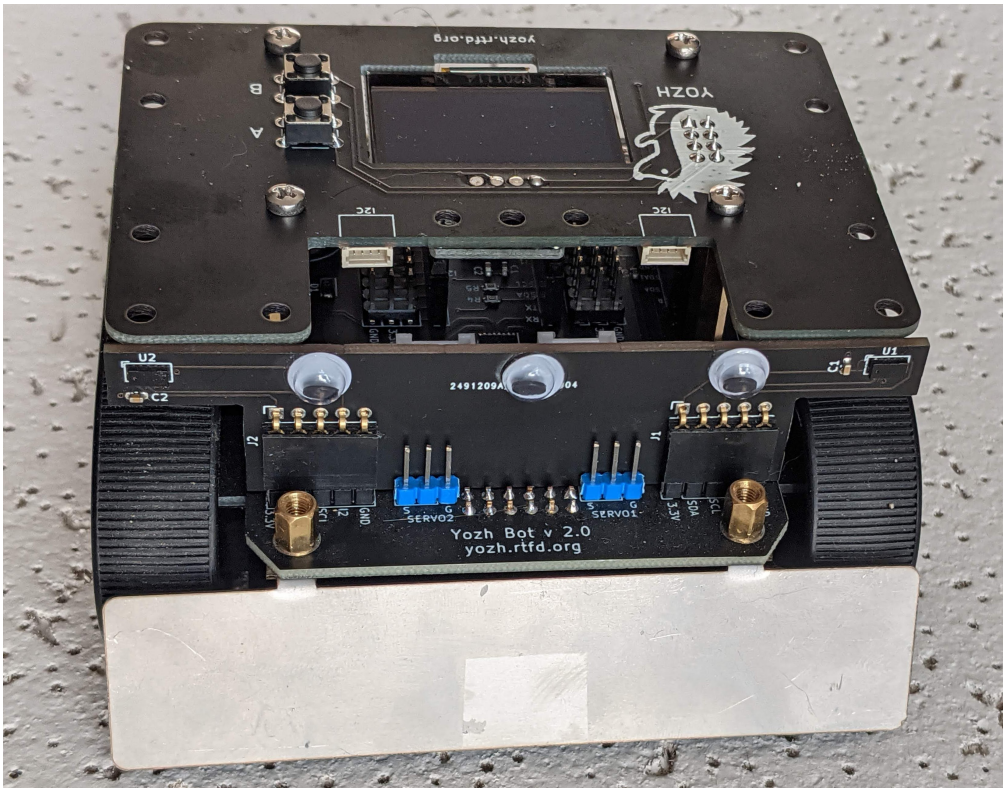

yozh.rtfd.org
Yozh Bot v 2.0
VBAT
GND
SCL
SDA
3.3V
SWDIO
SWCLK
RESET
R1
R2
R3
C3
C4
C6
C7
C8
C10
C11
D1
D4
J3
POWER
OFF
RESET
Adafruit
TrinkeyBerry
RP2040


Yozh Robot v2.0
B
A
yozh.rtfd.io
SERVO2
SERVO1
Yozh Bot v 2.0
yozh.rtfd.org

CHAPTER

THREE

YOZH FEATURES IN DETAIL



3.1 Dimensions and 3d model

Yozh dimensions are 10cm * 10 cm *5.5cm.

You can view the 3d model, created in Fusion360, at https://a360.co/3mLJR2U

Note that some components are missing from the 3d model - e.g. the main power switch.

3.2 Power

Yozh is powered by 4 AA (also known as LR6) batteries, inserted in the battery compartment at the bottom of the robot. It is highly recommended that you use heavy-duty rechargeable NiMh batteries such as Panasonic Eneloop Pro. You will also need a charger.

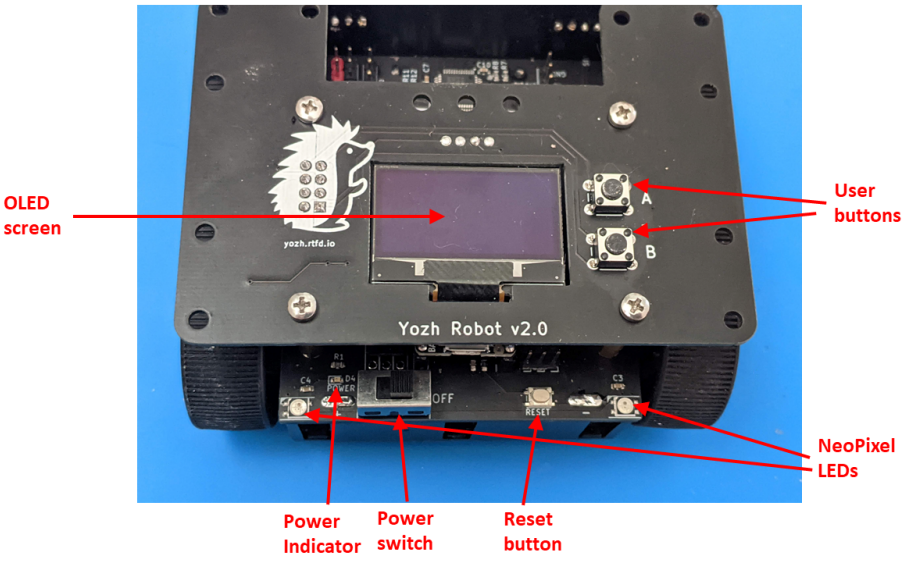
The robot also contains power switch, for disconnecting the battery, and a power indicator LED.

The robot contains a voltage regulator, which converts battery voltage to regulated 3.3v. Most of the robot electronics, including the microcontrollers and most of the sensors, are powered from 3.3v bus. Motors, servos (if connected), buzzer, and reflectance array are powered directly from the battery.

Note that NiMh batteries are not designed for high currents. Depending on the battery, you can expect 4-5A maximum; this would be enough for all on-board electronics and motors, and leave 1-2A for any electronics you want to add. This should be OK for micro servos and a couple of sensors, but if you want to use standard size servos or power-hungry devices such as AI cameras, you might have issues.

Connecting the ItsyBitsy microcontroller to a computer by USB cable provides power to 3.3v bus, even if the main battery is off. This would activate the microcontrollers and some of the electronics, but not the motors or servos.

You can check the battery voltage in software, using `battery()` function as described in *Yozh Library Guide*. Fully charged NiMH batteries should give about 5.5v.



3.3 Chassis and motors

Yozh robot is using tracked Zumo chassis by Pololu together with two 75:1 High Power 6V microgear motors. The motors are equipped with encoders (rotation counters), for speed control. The motors are controlled by DRV8833 motor driver by TI.

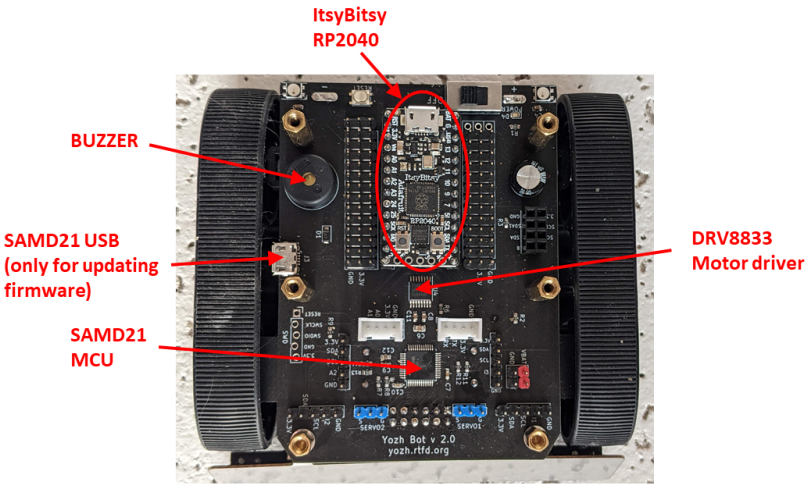
In the front of the robot there is a metal blade protecting it from collisions.

Maximal speed of the robot is FIXME m/s.

3.4 Electronics

The robot is controlled by two microcontrollers (MCU):

- Main (master) MCU: ItsyBitsy RP2040 This MCU is programmed by the user in CircuitPython. Provided CircuitPython library, documented in *Yozh Library Guide*, provides convenient functions for using all features of the robot.
- Secondary (slave) MCU: SAMD21G. This MCU is responsible for all low-level operations, converting high-level commands coming from main MCU into signals sent to motors, servos, NeoPixel LEDs and more, thus freeing pins and other resources of the main MCU for other purposes. Secondary MCU is also responsible for counting the encoder pulses and running the PID control loop maintaining motor speed. This MCU comes preloaded with firmware, written in C++ (using Arduino IDE). Normally, the user shouldn't need to touch this firmware.



The two MCUs talk to each other using I2C communication protocol; main MCU acts as the master on the I2C bus, and the secondary acts as slave.

Some of Yozh hardware is directly controlled by the main MCU, without going through the secondary one:

- OLED display
- Buttons
- Buzzer
- Distance sensors

Everything else – motors, encoders, servos, NeoPixel LEDs, reflectance sensor array, battery voltage monitoring, Inertial Motion Unit – is handled by the secondary MCU.

3.5 Buzzer and LEDs

Yozh contains a buzzer and two addressable RGB LEDs (commonly called NeoPixels), which can be used for showing information to the user.

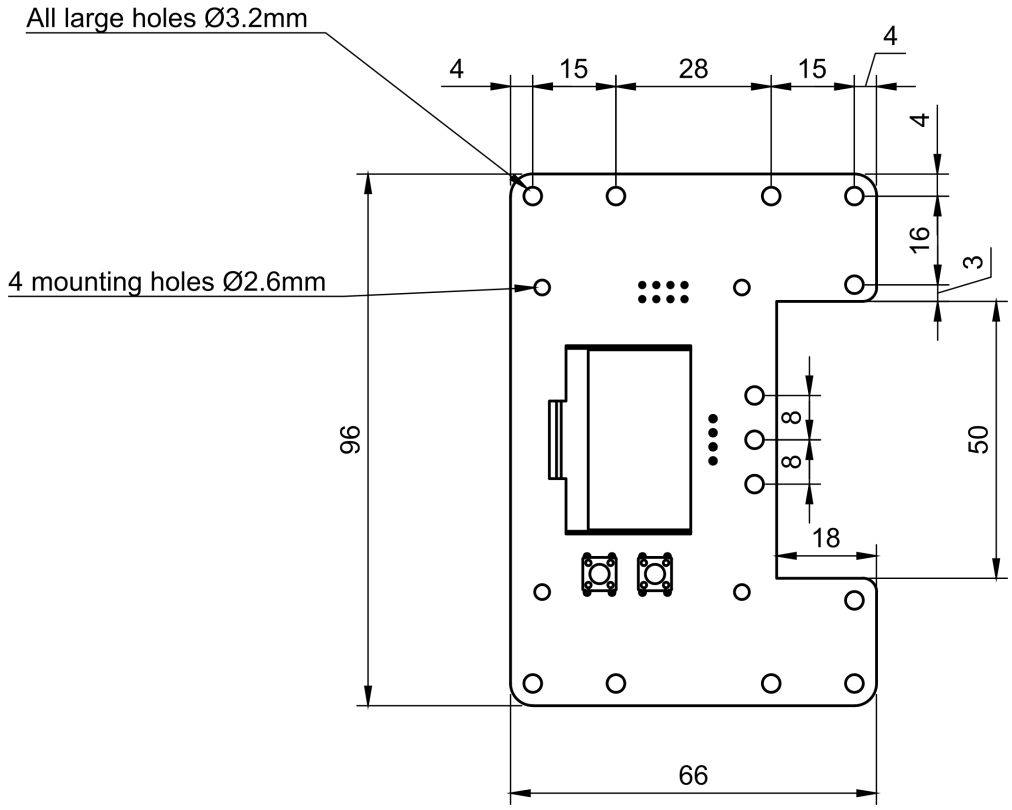
3.6 Top plate

On the top of Yozh robot, there is a top plate containing the following elements:

- an OLED display (1.3” size, 128x64 pixels)
- two buttons
- two Qwiic/Stemma QT I2C connectors (on the back side of the plate)
- a number of 3mm mounting holes for attaching additional electronics

The top plate is mounted on the robot using 22mm long M2.5 standoffs. If necessary, it can be removed – just make sure to modify the code as by default initialization code tries to initialize the OLED display.

The diagram below shows dimensions and hole locations.



3.7 Inertial Motion Unit

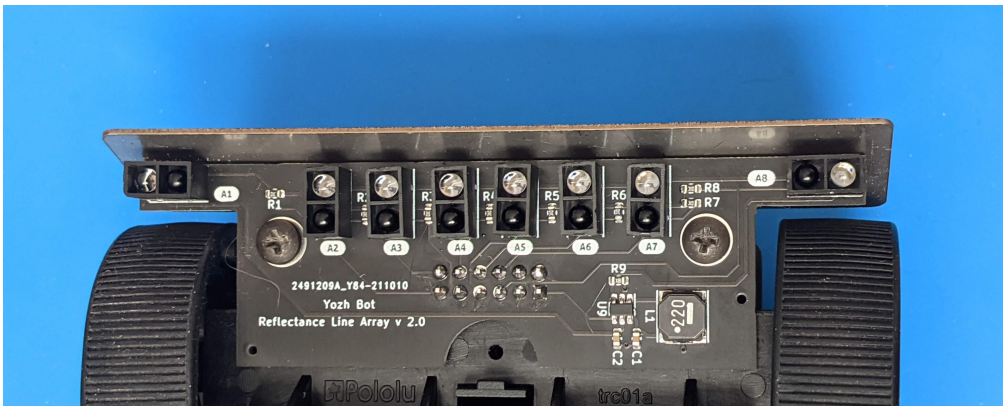
The top plate also contains an inertial motion unit (IMU): LSM6DSL by ST Microelectronics. This chip contains an accelerometer and a gyroscope, allowing the user to measure acceleration and rotation velocity. In addition to raw readings, the secondary MCU also runs a sensor fusion algorithm which uses the accelerometer and gyro data to constantly compute robot orientation in space, giving yaw, pitch, and roll angles. This can be used for precise navigation.

3.8 Distance and reflectance sensors

Yozh robot has several built-in sensors.

3.8.1 Reflectance array

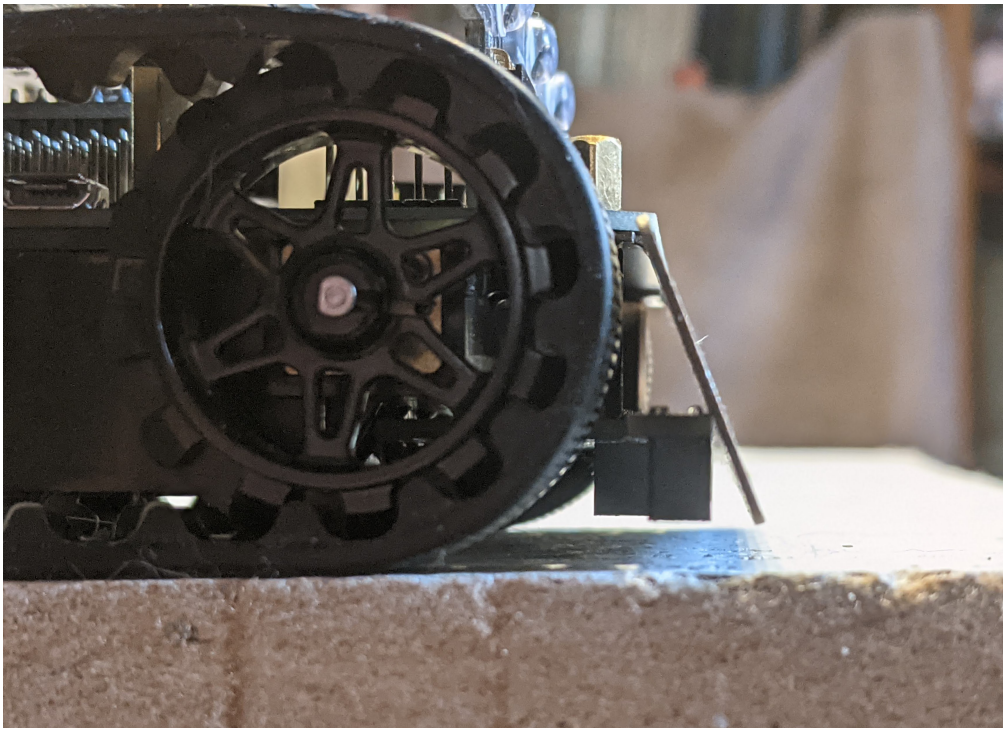
In the front of the robot, there is an array of 8 down-facing reflectance sensors for detecting field borders, following the line, and other similar tasks. It uses ITR9909 sensors by Everlight. The sensors are labeled A1 (rightmost) through A8 (leftmost).

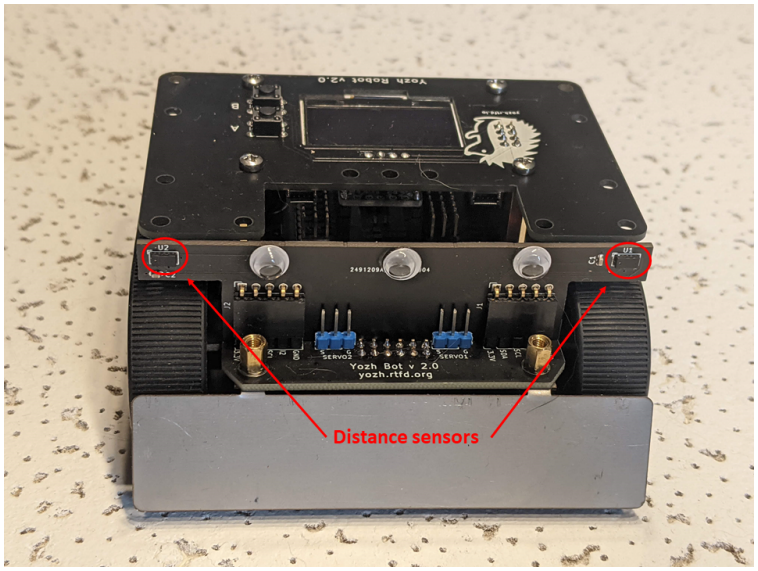


3.8.2 Distance sensors

Yozh also contains a removable board with two front-facing VL53L0X Time-of-Flight laser distance sensors by ST Microelectronics. These sensors have maximal distance of 2m; reliable sensing distance is closer to 1.5m. Each sensor has 25 degree field of view; this leaves a very small "blind spot" immediately in front of the robot, but provides complete coverage enabling the robot to detect any obstacle placed between 15-150 cm away.

These sensors can be used for obstacle avoidance, object tracking, or other similar purposes.




Yozh Robot v2.0
Yozh Bot v 2.0
yozh.rtfd.org
Distance sensors

3.9 Connecting additional sensors

Yozh uses some of the ItsyBitsy pins for controlling built-in electronics as shown in the table below. All other pins are available for connecting additional sensors or other electronics.

Pin	Function
SDA	Used by I2C bus.
SCL	
5	Buzzer
7	Used by front distance sensors board
12	Button B
13	Button A
25	Used by front distance sensors board

3.9.1 I2C bus

Pins SDA and SCL of ItsyBitsy are used for I2C communication with the following components of the robot:

- Secondary MCU (I2C address: `0x11`)
- OLED display (I2c address: `0x3c`)
- Front distance sensors (I2C addresses `0x29`, `0x30`)

You can connect additional devices to the same bus as long as they have addresses different from those listed above.

The bus operates at 3.3v; the main board contains pull-up resistors (2.2K) for the I2C bus, so additional pull-ups are not necessary.

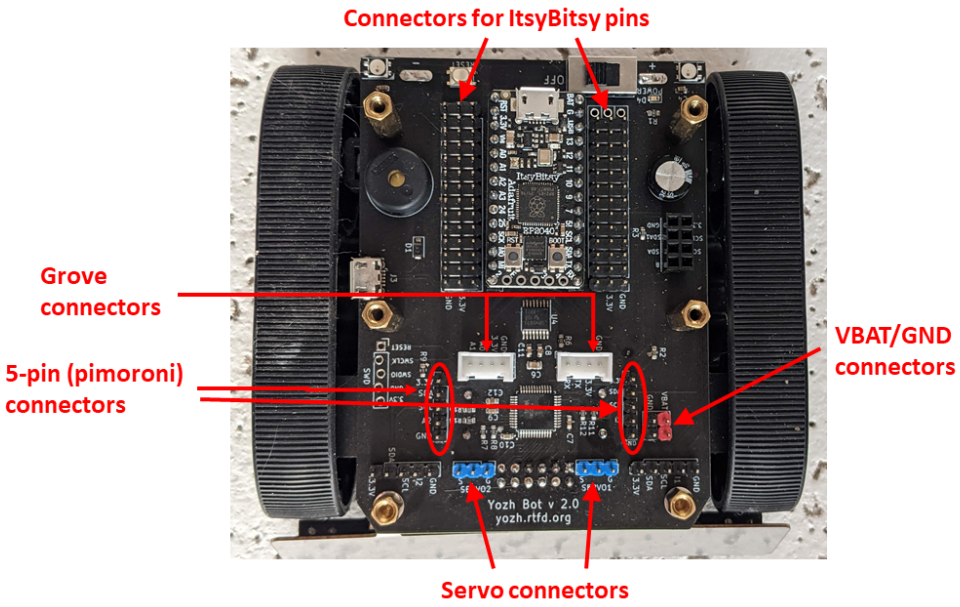
To connect new devices, you can use either the 5-pin connectors at the front of the robot or Qwiic/Stemma QT connectors at the bottom of the top plate.

3.9.2 Connectors

Yozh provides a number of connectors for connecting additional electronics to ItsyBitsy:

- On each side of the ItsyBitsy there are three rows of **male headers** (you need to remove the top plate to access these headers). The outer row is ground, the middle row is 3.3V, and each pin in the row closest to ItsyBitsy is connected to the corresponding pin of ItsyBitsy (except the VBUS pin of ItsyBitsy which is not connected). This allows you to connect to any pin of ItsyBitsy - including those used for other components.
- In the front of the robot, there are two 5-pin male connectors. They follow the pinout convention of Pimoroni breakout garden:
  - pin 1: 3.3v
  - pin 2: SDA
  - pin 3: SCL
  - pin 4: additional GPIO pin
  - pin 5: GND

  Pin 4 of the left 5-pin header (labeled I3) is connected to ItsyBitsy pin 8; pin 4 of the right header is connected to A2.



- In front of the robot, there are also 4-pin Grove connectors. These 2mm pitch locking connectors, designed by Seeed Studio, are commonly used in hobby robotics; a wide variety of sensors and other components using this system are available, see https://wiki.seeedstudio.com/Grove_System/. The pinouts of these connectors are as follows:
  - Left Grove connector: pin 1 - RX, pin 2 - TX, pin 3 - 3.3v, pin 4 - GND
  - Right Grove connector: pin 1 - A0, pin 2 - A1, pin 3 - 3.3v, pin 4 - GND
- On the left side of the robot, there are additional male headers for power connections, connected to GND and battery (VBAT). Depending on the batteries used and their charge level, voltage of VBAT pins can range from 4.5 - 6.5 V.
- Finally, at the bottom of the top plate, there are two Qwiic/Stemma QT I2C connectors.

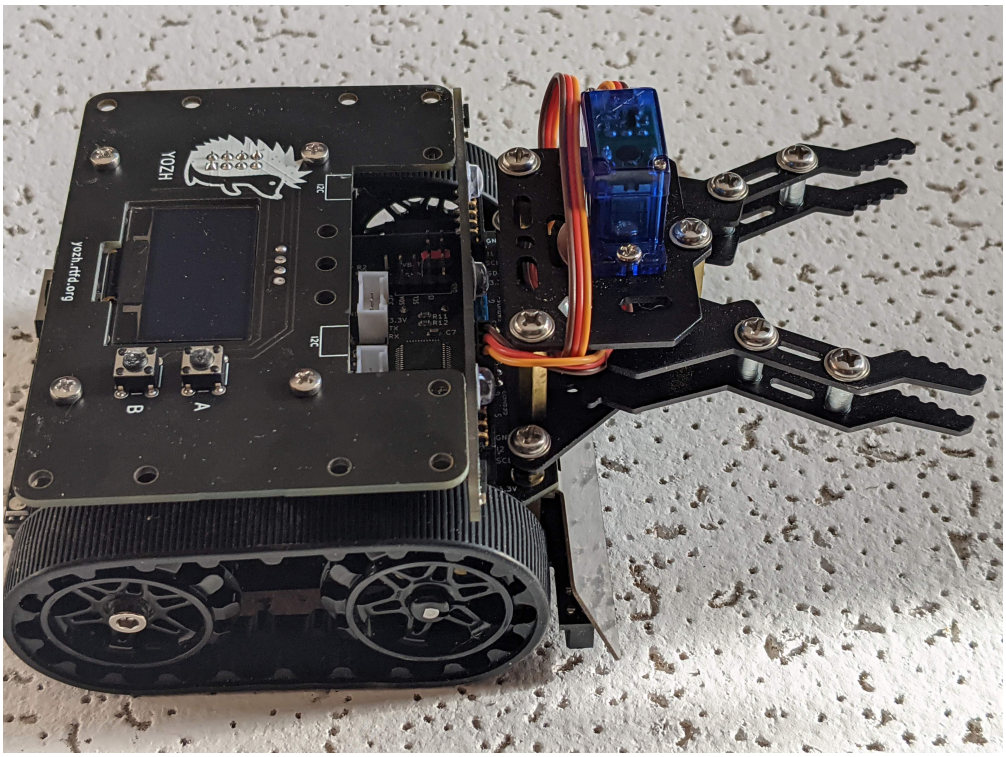
3.10 Servos and attachments

Yozh contains two ports for connecting servos, as shown in the photo below. These ports use standard pin order: GND, VCC, SIGNAL. To help you identify the pins, there are small letters S (Signal) and G (GND) next to corresponding pins. VCC is directly connected to the battery, so voltage ranges between 4.5-6.5V depending on battery charge.

Yozh can also be used with mechanical attachments (grabber, forklift,...) by DFRobot – see photo below for Yozh with the grabber (beetle) attachment. US customers might find it easier to order DFrobot kits from DigiKey:

- Grabber
- Forklift
- Loader

Note that some of these attachments might interfere with the distance sensors.


YOZH
yozh.rtfd.org
B
A

CHAPTER

FOUR

YOZH LIBRARY REFERENCE

In this chapter, we give full list of all commands provided by Yozh Circuit Python library. We assume that the user has already installed Yozh library on the robot, as described in the *Quickstart Guide*.

This document describes version 2.0 of the library. It is intended to be used with CircuitPython 7.

4.1 Initialization and general functions

To begin using the library, you need to put the following in the beginning of your *code.py* file:

```
import yozh
bot = yozh.Yozh()
```

This creates an object with name `bot`, representing your robot. From now on, all commands you give to the robot will be functions and properties of `bot` object. We will not include the name bot in our references below; for example, to use a command `stop_motors()` described below, you would need to write `bot.stop_motors()`.

By default, creating bot object also initializes the OLED display; it will produce errors if the OLED display is not found. If for some reason you are not using OLED, you can initialize the robot using this form of initialization command:

```
import yozh
bot = yozh.Yozh(oled = None)
```

Here are some basic functions:

fw_version()

Returns firmware version as a string. e.g. *2.1*.

battery()

Returns battery voltage, in volts. For normal operation it should be at least 4.5 V.

4.2 Display, buttons, LEDs

Yozh contains a buzzer, two NeoPixel LEDs in the back and an 128x64 OLED screen and two buttons on the top plate, for interaction with the user. To control them, use the functions below.

4.2.1 LEDs

set_led_L(color)

set_led_R(color)

These commands set the left (respectively, right) LED to given color. Color must be a list of 3 numbers, showing the values of Red, Green, and Blue colors, each ranging between 0–255, e.g. `bot.set_led_L([255,0,0])` to set the left LED red. You can also define named colors for easier use, e.g.

```
BLUE=[0,0,255]

bot.set_led_L(BLUE)
```

set_leds(color_l, color_r)

Set colors of both LEDs at the same time. As before, each color is a list of three values. Parameter `color_r` is optional; if omitted, both LEDs will be set to the same color.

set_led_brightness(value)

Set the maximal brightness of both LEDs to a given value (ranging 0-255). Default value is 64 (i.e., 1/4 of maximal brightness), and it is more than adequate for most purposes, so there is rarely a need to change it. Setting brightness to 255 would produce light bright enough to hurt your eyes (and drain the batteries rather quickly)

4.2.2 Buzzer

buzz(freq, dur=0.5)

Buzz at given frequency (in hertz) for given duration (in seconds). Second parameter is optional; if omitted, duration of 0.5 seconds is used.

4.2.3 Buttons

wait_for(button)

Waits until the user presses the given button. There are two possible pre-defined buttons: `bot.button_A` and `bot.button_B`

is_pressed(button)

Returns `True` if given button is currently pressed and `False` otherwise.

choose_button()

Waits until the user presses one of the two buttons. This function returns string literal `A` or `B` depending on the pressed button:

```
bot.set_text("Press any button", line1)
#wait until user presses one of buttons
if (bot.choose_button()=="A"):
    # do something
else:
    # button B was pressed
    # do something else
```

4.2.4 OLED

The easiest way to interact with OLED display is by using the commands below.

clear_display()

Clears all text and graphics from display

add_textbox()

Add textbox (also known as label) to display. You can enter the actual text when creating the textbox, or replace it later. The command returns index of the textbox, which can be used to update the contents of the textbox later.

The basic use of this command is

```
line1 = bot.add_textbox(text_position=(0,10), text="Yozh is happy!")
```

The command accepts a number of optional parameters, documented below.

add_textbox()

Parameters

- **text_font** – The path to your font file for your data text display.
- **text_position** – The position of text on the display in an (x, y) tuple.
- **text_wrap** – When non-zero, the maximum number of characters on each line before text is wrapped. (for long text data chunks). Defaults to 0, no wrapping.
- **text_maxlen** – The max length of the text. If non-zero, it will be truncated to this length. Defaults to 0, no truncation.
- **text_scale** – The factor to scale the default size of the text by
- **line_spacing** (float) – The factor to space the lines apart
- **text_anchor_point** ((float, float)) – Values between 0 and 1 to indicate where the text position is relative to the label
- **text** – If this is provided, it will set the initial text of the textbox.

set_text(text, i)

Replaces text in textbox with index `i`, e.g.

```
line1 = bot.add_textbox(text_position=(0,10), text="Yozh is happy!")
time.sleep(1.0)
bot.set_text("Press any button", line1)
```

Writing empty text into a textbox deletes it. Thus, if you want to erase current text but keep the textbox for future use, replace the text with a single space `bot.set_text(" ", i)`

Advanced users may also use any commands from CircuitPython `displayio` module to put text and graphics on the OLED display as described in https://learn.adafruit.com/circuitpython-display-support-using-displayio. The display object of the robot can be accessed as `bot.display`, e.g.

```
 display = bot.display
 # Setup the file as the bitmap data source
 bitmap = displayio.OnDiskBitmap("/purple.bmp")

# Create a TileGrid to hold the bitmap
tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
```

(continues on next page)

(continued from previous page)

```
# Create a Group to hold the TileGrid
group = displayio.Group()

# Add the TileGrid to the Group
group.append(tile_grid)

# Add the Group to the Display
display.show(group)
```

4.3 Motor control

Of course, main use of this robot is to drive around, and for this, we need to control the motors.

4.3.1 Basic control

set_motors(power_L, power_R)

Set power for left and right motors. `power_L` is power to left motor, `power_R` is power to right motor. Each of them should be between 100 (full speed forward) and -100 (full speed backward).

Note that because no two motors are exactly identical, even if you give both motors same power (e.g. `set_motors(60,60)`), their speeds might be slightly different, causing the robot to veer to one side instead of moving straight. To fix that, use PID control as described below.

stop_motors()

Stop both motors.

4.3.2 Encoders

Both motors are equipped with encoders (essentially, rotation counters). For 75:1 HP motors, each motor at full speed produces about 4200 encoder ticks per second.

reset_encoders()

Resets both encoders

get_encoders()

Gets values of both encoders and saves them. These values can be accessed as described below

encoder_L

encoder_R

Value of left and right encoders, in ticks, as fetched at last call of `get_encoders()`. Note that these values are not automatically updated: you need to call `get_encoders()` to update them

get_speeds()

Gets the speeds of both motors and saves them. These values can be accessed as described below

speed_L

speed_R

Speed of left and right motors, in ticks/second, as fetched at last call of `get_speeds()`. Note that these values are not automatically updated: you need to call `get_speeds()` to update them

4.3.3 PID

PID is an abbreviation for Proportional-Integral-Differential control. This is the industry standard way of using feedback (in this case, encoder values) to maintain some parameter (in this case, motor speed) as close as possible to target value.

Yozh bot has PID control built-in; however, it is not enabled by default. To enable/disable PID, use the functions below.

Before enabling PID, you need to provide some information necessary for its proper operation. At the very minimum, you need to provide the speed of the motors when running at maximal power. For 75:1 motors, it is about 4200 ticks/second; for other motors, you can find it by running `motors_test.py` example.

`configure_PID(maxspeed)`

Configures parameters of PID algorithm, using motors maximal speed in encoder ticks/second.

`PID_on()`

`PID_off()`

Enables/disables PID control (for both motors).

Once PID is enabled, you can use same functions as before (`set_motors()`, `stop_motors()`) to control the motors, but now these functions will use encoder feedback to maintain desired motor speed.

4.3.4 Drive control

Yozh python library also provides higher level commands for controlling the robot.

`go_forward (distance, speed=50)`

`go_backward(distance, speed=50)`

Move forward/backward by given distance (in centimeters). Parameter `speed` is optional; if not given, default speed of 50 (i.e. half of maximal) is used.

Note that distance and speed should always be positive, even when moving backward.

`turn(angle, speed=50)`

Turn by given angle, in degrees. Positive values correspond to turning right (clockwise). Parameter `speed` is optional; if not given, default speed of 50 (i.e. half of maximal) is used.

Note that all of these commands use encoder readings to determine how far to drive or turn. Of course, to do this one needs to know how to convert from centimeters or degrees to encoder ticks. This information is stored in properties `bot.CM_TO_TICKS` and `bot.DEG_TO_TICKS`. By default, Yozh library uses `CM_TO_TICKS = 150`, `DEG_TO_TICKS=14`, which should be correct for 75:1 motors. If you find that the robot consistently turns too much (or too little), you can change these values, e.g.

```
bot.DEG_TO_TICKS=15
bot.turn(90)
```

4.4 Servos

Yozh has two ports for connecting servos. To control them, use the commands below.

set_servo1(position)

set_servo2(position)

Sets servo 1/servo 2 to given position. Position ranges between 0 and 1; value of 0.5 corresponds to middle (neutral) position.

Note that these commands expect that the servo is capable of accepting pulsewidths from 500 to 2500 microseconds. Many servos use smaller range; for example, HiTec servos have range of 900 to 2100 microseconds. For such a servo, it will reach maximal turning angle for position value less than one (e.g., for HiTec servo, this value will be 0.8); increasing position value from 0.8 to 1 will have no effect. Similarly, minimal angle will be achieved for `position = 0.2`.

Warning: please remember that if a servo is unable to reach the set position because of some mechanical obstacle (e.g., grabber claws can not fully close because there is an object between them), it will keep trying, drawing significant current. This can lead to servo motor overheating quickly; it can also lead to voltage drop of Yozh battery, interfering with operation of motors or other electronics. Thus, it is best to avoid such situtations.

4.5 Reflectance sensor array

Yozh has a built-in array of reflectance sensors, pointed down. These sensors can be used to detect filed borders, for following the line, and other similar tasks.

4.5.1 Basic usage

linearray_on()

linearray_off()

Turns reflectance array on/off. By default, it is off (to save power).

linearray_raw(i)

Returns raw reading of sensor `i`. One can use either indices 0...7 or (preferred) named values `bot.A1 = 0` ... `bot.A8 = 7`. Readings range 0-1023 depending on amount of reflected light: the more light reflected, the **lower** the value. Typical reading on white paper is about 50-80, and on black painted plywood, 950. Note that black surfaces can be unexpectedly reflective; on some materials which look black to human eye, the reading can be as low as 600.

4.5.2 Calibration

Process of calibration refers to learning the values corresponding to black and white areas of the field and then using these values to rescale the raw readings.

calibrate()

Calibrates the sensors, recording the lowest and highest values. This command should be called when some of the sensors are on the white area and some, on black.

linearray_cal(i)

Returns reading of sensor `i`, rescaled to 0-100: white corresponds to 0 and black to 100. It uses the calibration data, so should only be used after the sensor array has been calibrated.

`sensor_on_white(i)`

Returns `True` if sensor `i` is on white and false otherwise. A sensor is considered to be on white if calibrated value is below 50.

`sensor_on_black(i)`

Returns `True` if sensor `i` is on black and false otherwise.

4.5.3 Line following

A common task for such robots is following the line. To help with that, Yozh library provides the helper function.

`line_position_white()`

Returns a number showing position of the line under the robot, assuming white line on black background. The number ranges between -5 (line far to the left of the robot) to 5 (line far to the right of the robot). 0 is central position: line is exactly under the center of the robot.

Slightly simplifying, this command works by counting how many sensors are to the left of the line, how many are to the right, and then taking the difference. It works best for lines of width 1-2cm; in particular, electric tape or gaffers tape (1/2" or 3/4") works well.

This command only uses the central 6 sensors; rightmost and leftmost sensor (A1 and A8) are not used.

If there is no line under these sensors, the value returned by this command will be close to 0, but can not be relied on.

`line_position_black()`

Same as above, but assuming black line on white background.

4.6 Distance sensors

The robot is equipped with two front-facing distance sensors, using Time-of-Flight laser technology, which can be accessed using the commands below.

`distance_L.range`

`distance_R.range`

Distance reading of left (respectively, right) sensor, in mm. Note that it is a property, not a function - do not use parentheses.

4.7 Inertial Motion Unit

This section describes the functions for using the built-in Inertial Motion Unit (IMU).

Yozh contains a built-in Inertial Motion Unit (IMU), which is based on LSM6DSL chip from ST Microelectronics. This chip combines a 3-axis accelerometer and a 3-axis gyro sensor, which provide information about acceleration and rotational speed. The sensor is placed on the back side of the top plate. Yozh firmware combines the sensor data to provide information about robot's orientation in space, in the form of Yaw, Pitch, and Roll angles. (Yozh firmware is based on the work of Kris Winer and uses data fusion algorithm invented by Sebastian Madgwick.)

Below is the description of functions related to IMU. You can also check sample code in *imu_test* example sketch included with Yozh CircuitPython library.

4.7.1 Initialization

By default, the IMU is inactive. To start/stop it, use the functions below.

void **IMU_start()**

Activate IMU

void **IMU_stop()**

Stop the IMU

bool **IMU_status()**

Returns IMU status. This function can be used to verify that IMU activation was successful. Possible values are:

- 0: IMU is inactive
- 1: IMU is active
- 2: IMU is currently in the process of calibration

4.7.2 Calibration

Before use, the IMU needs to be calibrated. The calibration process determines and then applies corrections (offsets) to the raw data; without these corrections, the data returned by the sensor is very inaccurate.

If you haven't calibrated the sensor before (or want to recalibrate it), use the following function:

void **IMU_calibrate()**

This function will determine and apply the corrections; it will also save these corrections in the flash storage of the Yozh slave microcontroller, where they will be stored for future use. This data is preserved even after you power off the robot (much like the usual USB flash drive).

This function will take about 10 seconds to execute; during this time, the robot must be completely stationary on a flat horizontal surface.

If you had previously calibrated the sensor, you do not need to repeat the calibration process - by default, upon initialization the IMU loads previously saved calibration values.

Note that the IMU is somewhat sensitive to temperature changes, so if the temperature changes (e.g., you moved your robot from indoors to the street for testing), it is advised that you recalibrate the IMU.

4.7.3 Reading Values

Yozh allows you to read both the raw data (accelerometer and gyro readings) and computed orientation, using the following functions:

void **IMU_get_accel()**

Fetches from the sensor raw acceleration data and saves it using member variables `ax`, `ay`, `az`, which give the acceleration in x-, y-, and z- directions respectively in in units of 1g (9.81 m/sec^2) as floats.

void **IMU_get_gyro()**

Fetches from the sensor raw gyro data and saves it using member variables `gx`, `gy`, `gz`, which give the angular rotation velocity around x-, y-, and z- axes respectively, in degree/s (as floats).

float **IMU_yaw()**

float **IMU_pitch()**

float **IMU_roll**()

These functions return yaw, pitch, and roll angles for the robot, in degrees. These three angles determine the robot orientation as described below:

- yaw is the rotation around the vertical axis (positive angle corresponds to clockwise rotation, i.e. right turns), relative to the starting position of the robot
- pitch is the rotation around the horizontal line, running from left to right. Positive pitch angle corresponds to raising the front of the robot and lowering the back
- roll is the rotation around the horizontal line running from front to back. Positive roll angle corresponds to raising the left side of the robot and lowering the right.

For more information about yaw, pitch, and roll angles, please visit https://en.wikipedia.org/wiki/Aircraft_principal_axes

INDEX

I