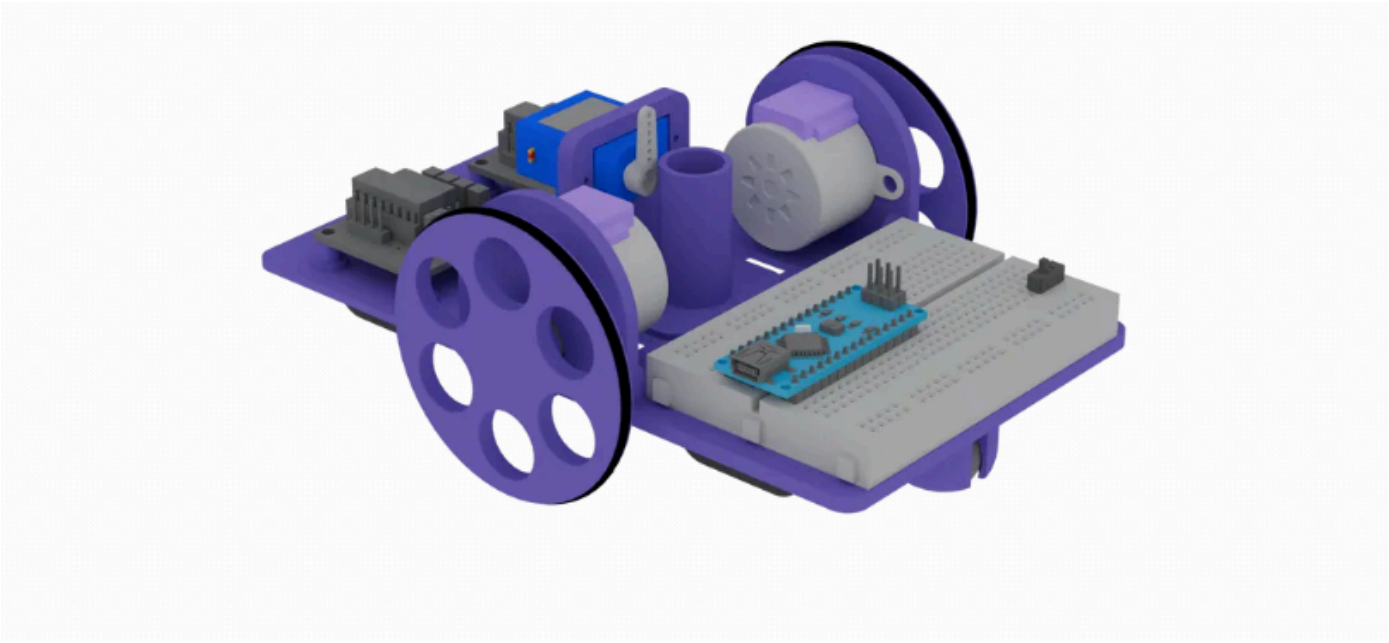


RISE Turtle Robot Instructions

Version 2.0

Your Guide to Assembling and Programming the Turtle Robot

Ages:

9+

Welcome

Thank you for purchasing a RISE kit and supporting our work! We hope you have as much fun building your kit as we did creating it.

- The RISE Team

Warnings


Choking hazard due to small parts. Some parts may have sharp points. Not suitable for children under 4.


Read instructions before proceeding. Failure to follow instructions may lead to part malfunction.


Stop working immediately if you smell anything burning, turn off the robot, and take out the batteries.


Exercise caution when attaching parts. Kit contains wires that may have pointed edges.

Suggestions


Adult supervision recommended.

Contents

Assembly

Programming

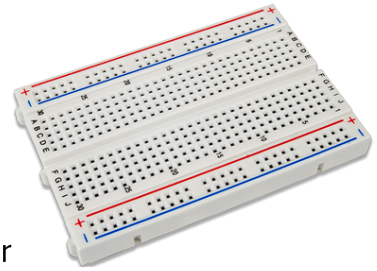
Going Further

Inventory of Parts

2x Stepper Motor & Driver Board

3x AA Battery Holders

Breadboard



Switch

Capicator

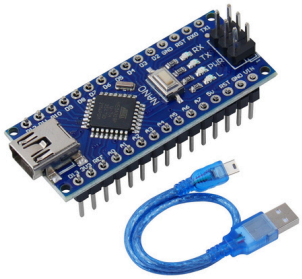
6x Male/Male Jumper Wires

12x Male/Female Jumper Wires

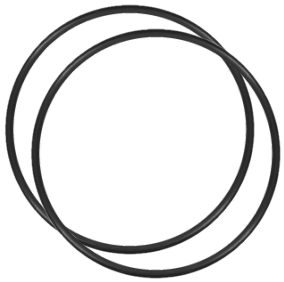
Steel Ball Bearing

Servo Motor

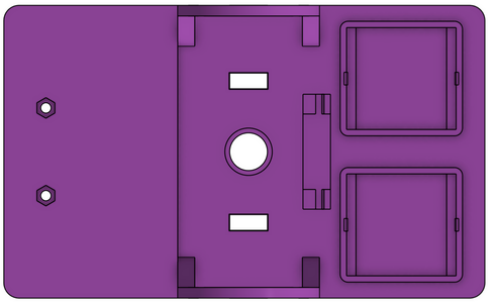
Arduino Nano



2x O-Rings



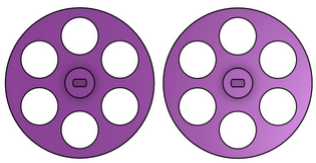
Chassis



Ball Caster

M3 Screws & Nuts

Wheels

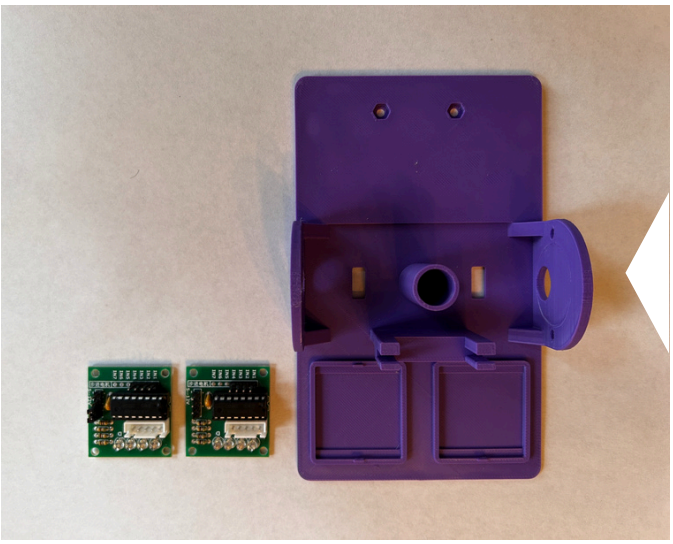


2x Pen Collar

Stepper Motors

Step 1

The first step to building your robot is adding the stepper motors to the chassis. These motors will be what allows your robot to drive!

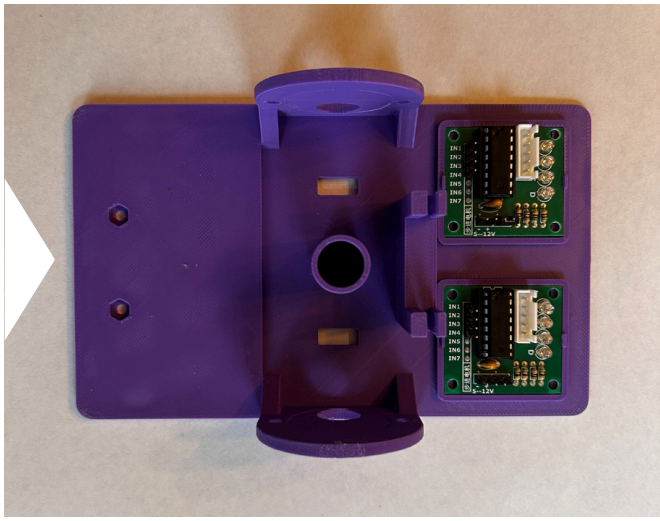


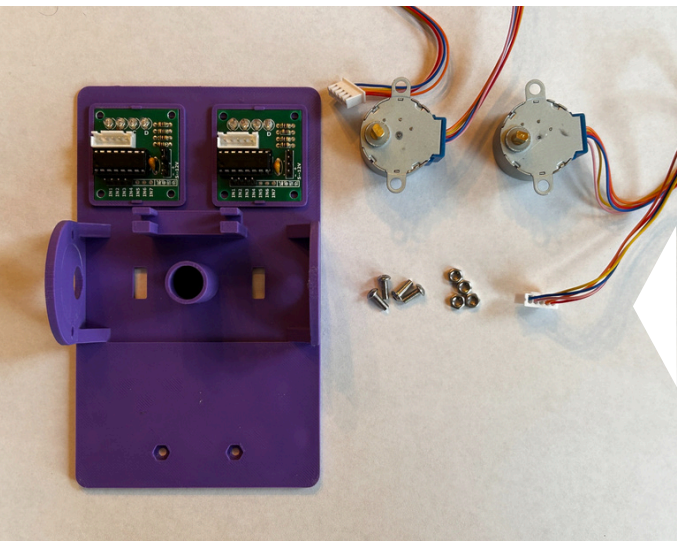
To begin, collect: 3D printed chassis, stepper motor driver boards (x2), s

Position the motor driver boards on top of the square mounts.

Slide one side of the driver board under the clip

Snap the other side into place





Now that the stepper motor driver boards are installed, we can attach the motors.

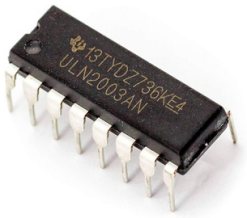
Collect these parts: stepper motors (x2), screwdriver, (4x) 8mm screws, and (4x) hex nuts.

What Does It Do?

The **ULN2003** stepper motor driver board helps the motor turn by sending it the right signals in the right order.

It includes a small control chip (called the **ULN2003**) and a board that makes it easy to plug in wires and the motor.

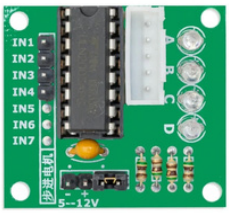
ULN2003



The **ULN2003** is an integrated circuit (IC), which means it's a tiny chip that contains many electronic parts inside to help control things like motors.

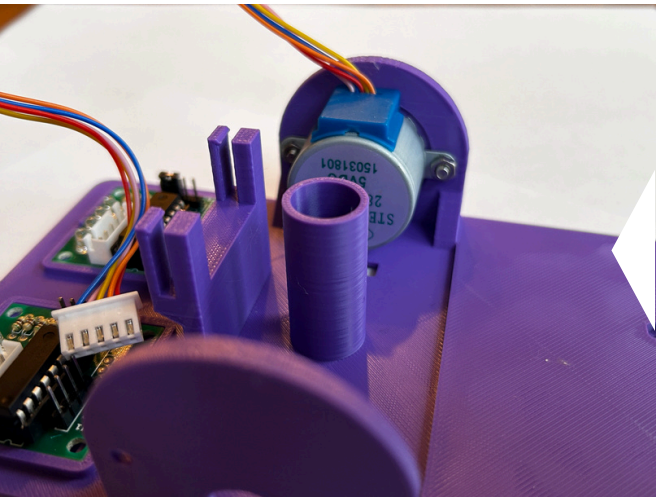
In our case, it will be controlling the motors that spin the wheels on the robot!

Breakout Board

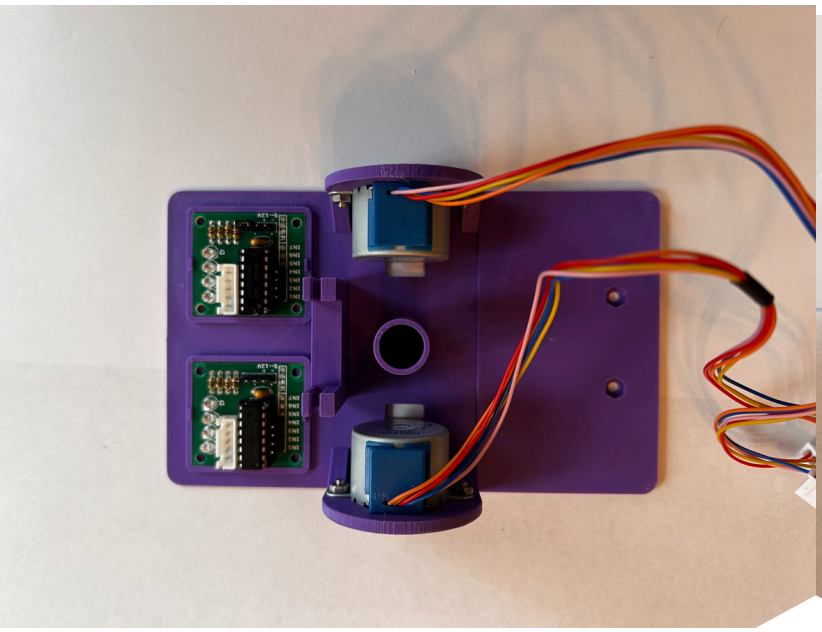


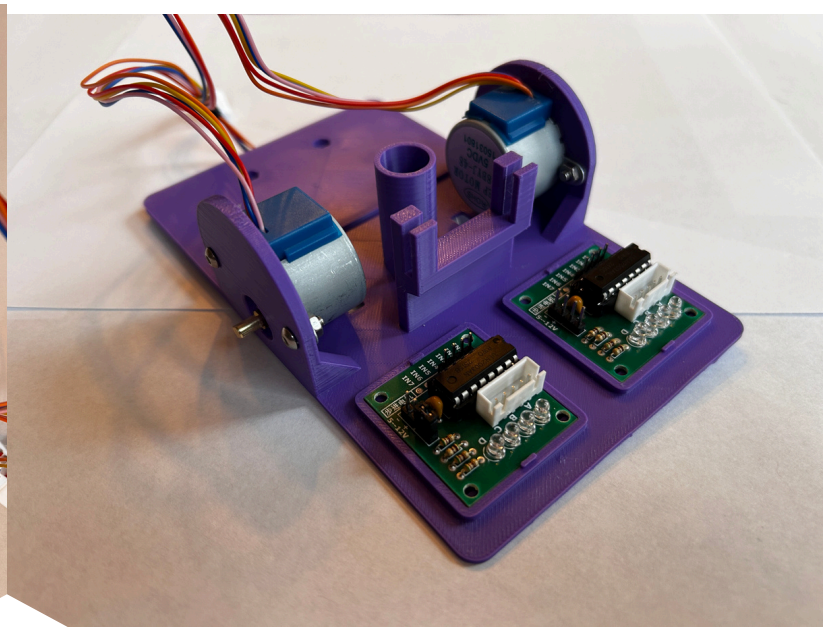
A **breakout board** is a small circuit board that makes a chip easier to use by adding connectors and pins for easy wiring.

This one includes motor and wire connectors, plus lights that show the motor is moving.

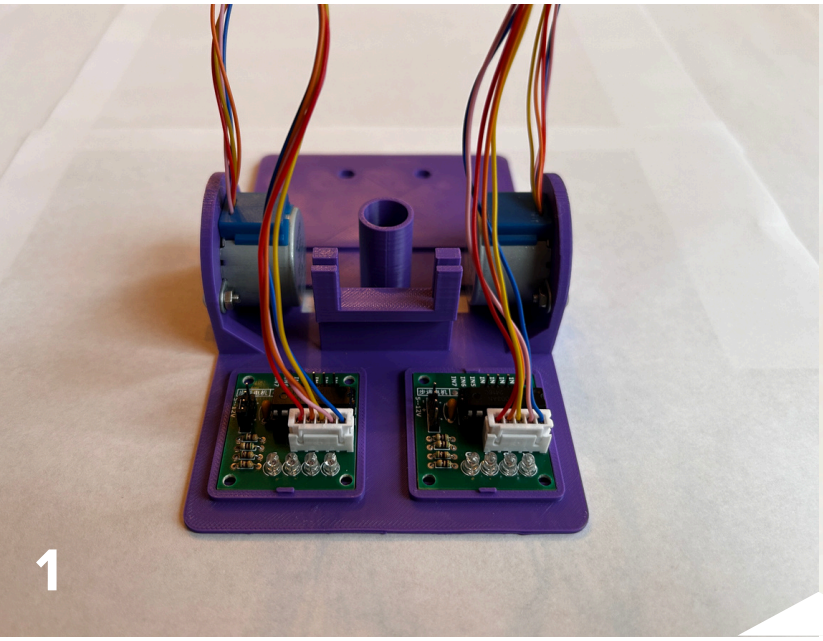


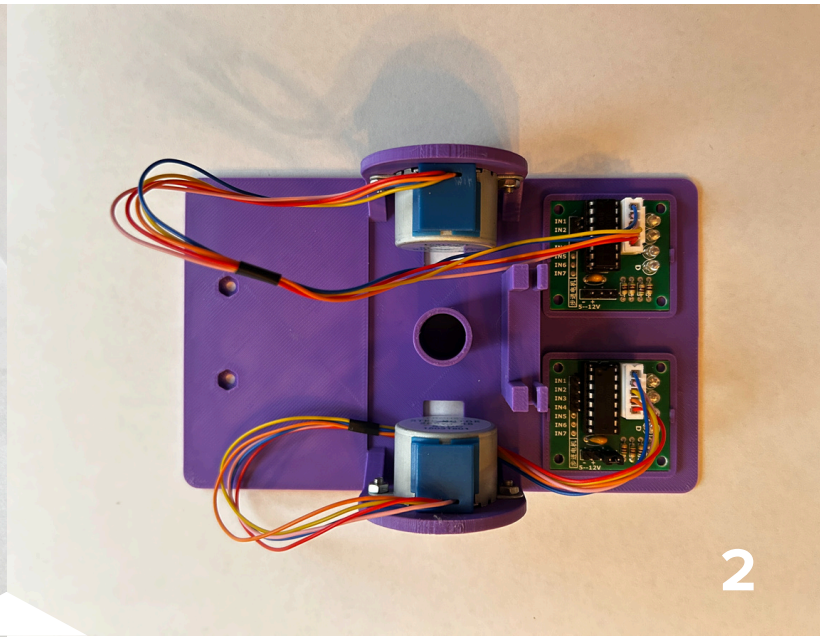
Attach the screws through the rounded motor mounts on the chassis from the outside to the inside, then fit the stepper motor onto the screws and secure it with nuts. Make sure the screw head is pointing out of the robot, and the nuts are pointing toward the inside. Repeat on the other side.





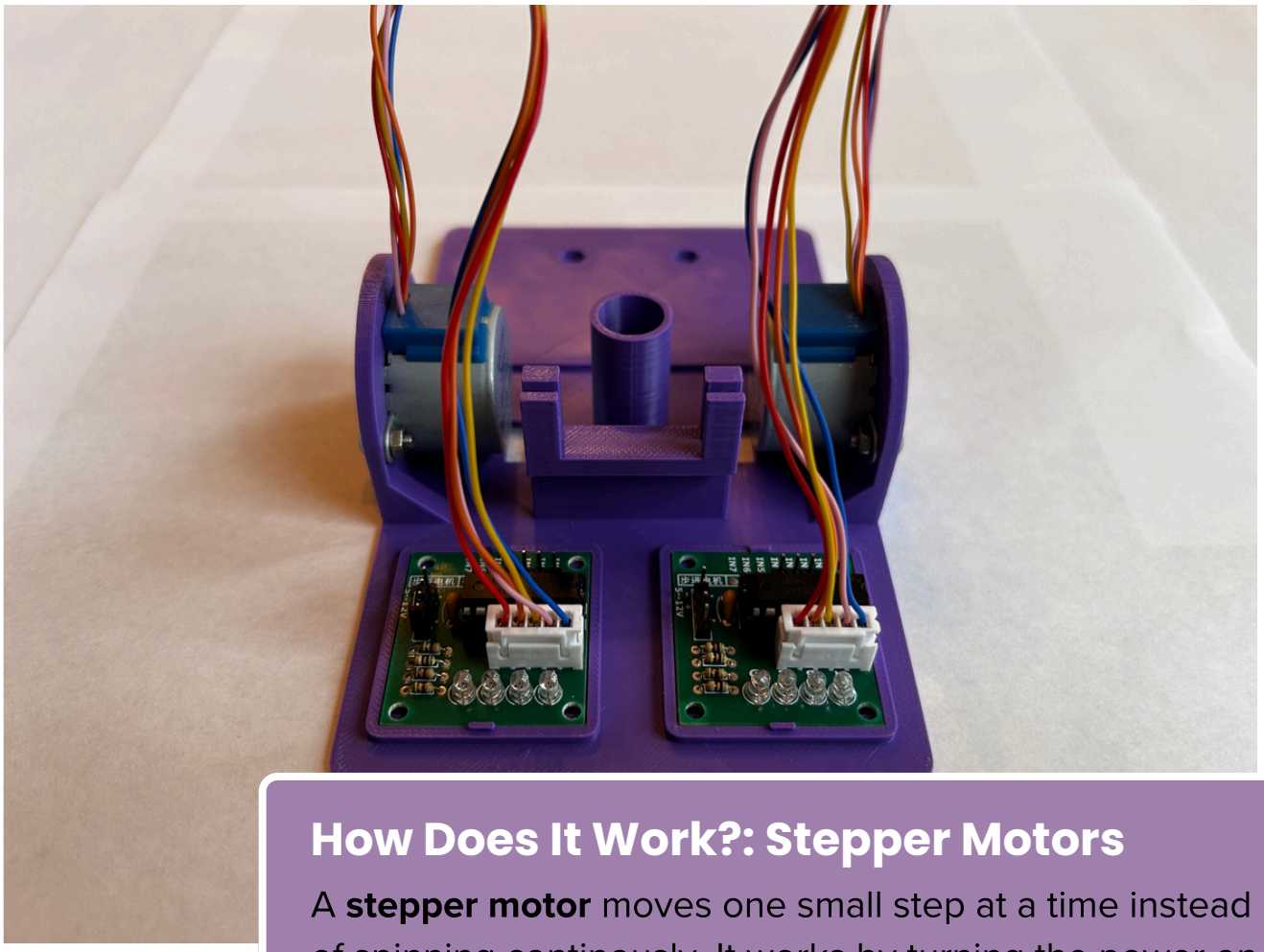
Check that your robot looks like the pictures above





Connect the motor wires to the white connectors on the driver boards. The left motor goes to the left breakout board, and the right motor to the right breakout board. (Picture 1)

Push the wires down firmly so they don't stick out. (Picture 2)



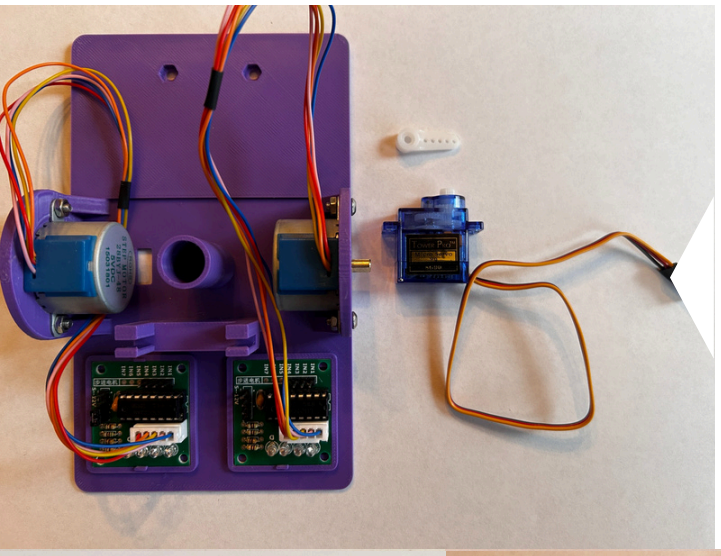
How Does It Work?: Stepper Motors

A **stepper motor** moves one small step at a time instead of spinning continously. It works by turning the power on and off to different parts inside the motor in the correct order. This makes it perfect for our robot because we can control exactly how it moves to draw accurately!

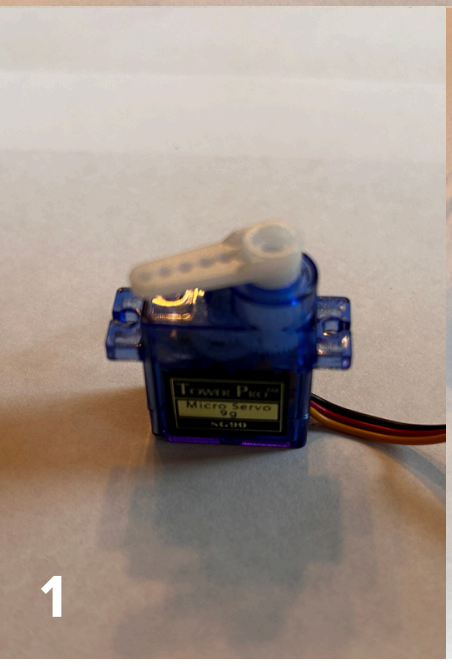
Servo Motor

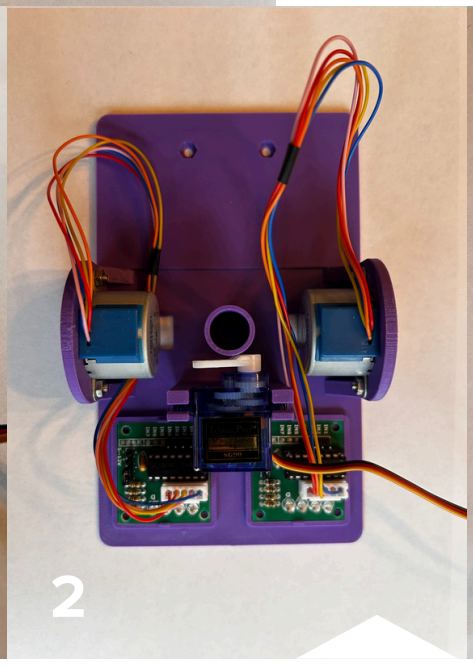
Step 2

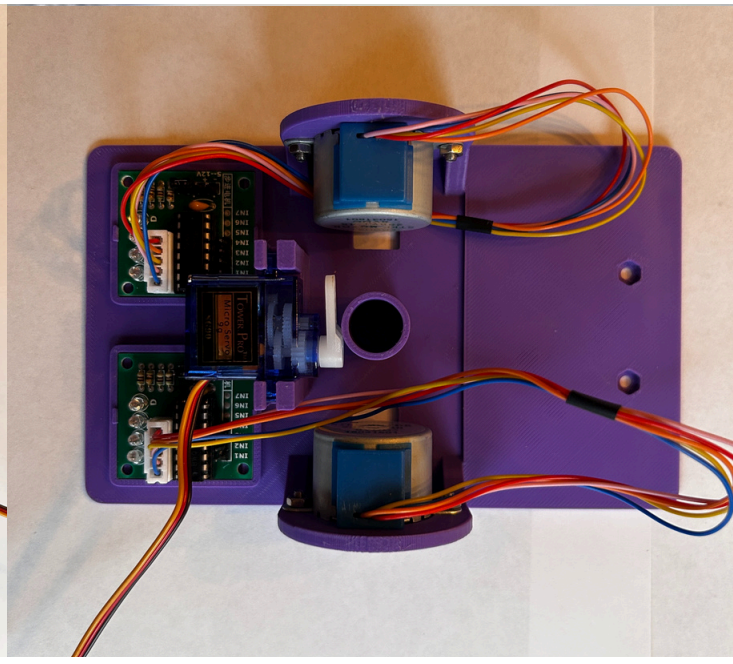
The servo motor is responsible for raising and lowering the marker. Without it your robot would only be able to draw in continous lines, not very useful!



Collect these parts: servo motor, smallest servo motor horn. If it's not already attached, connect the servo horn to the top of the servo (Picture 1).

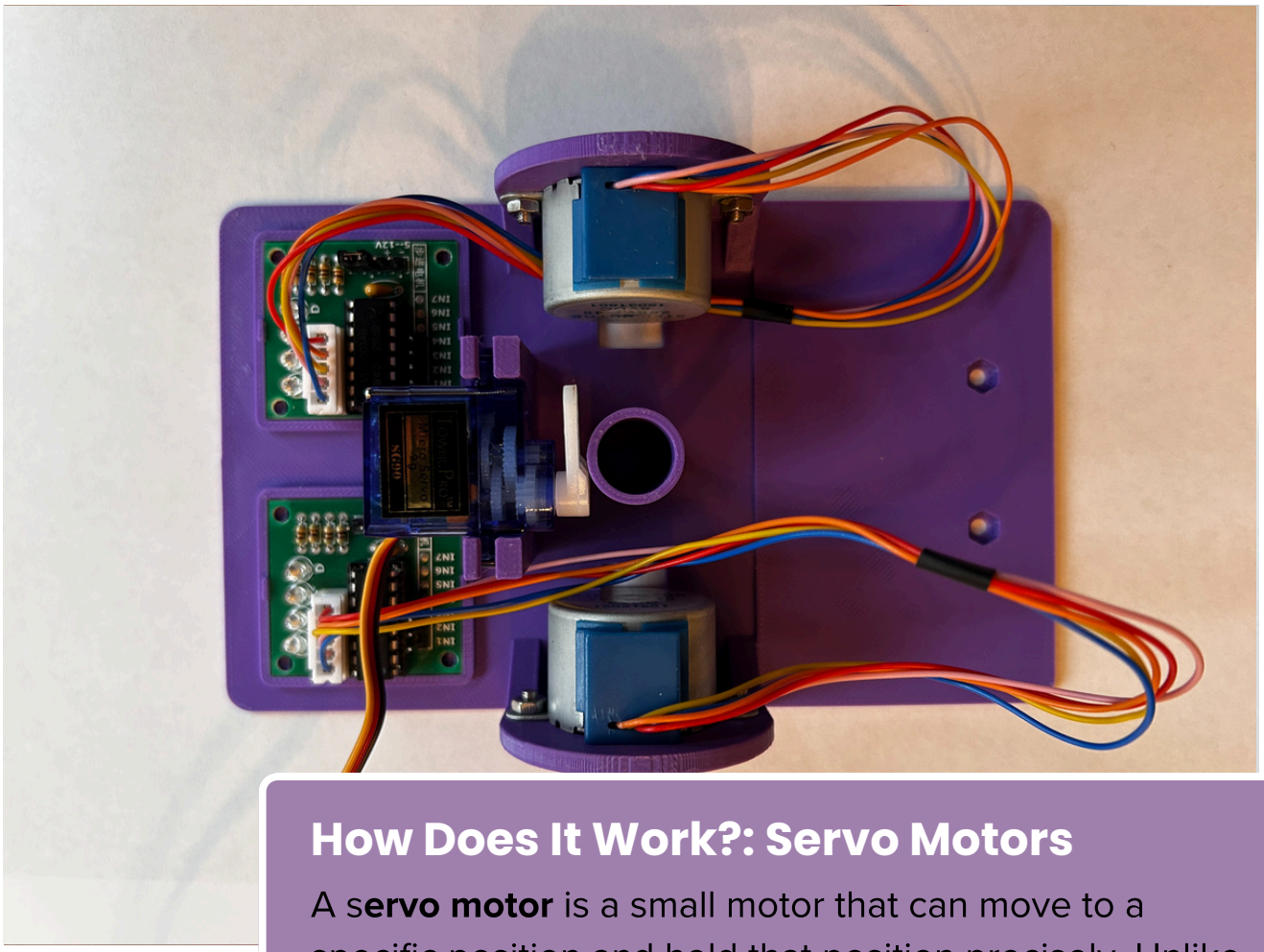






Position the servo in the servo bracket as seen in picture 2.

Push the servo into the clips.



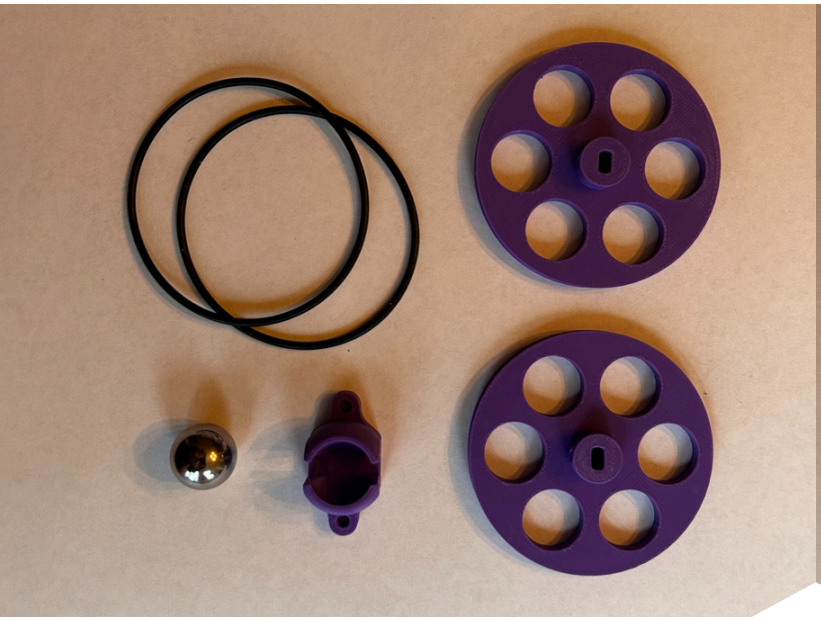
How Does It Work?: Servo Motors

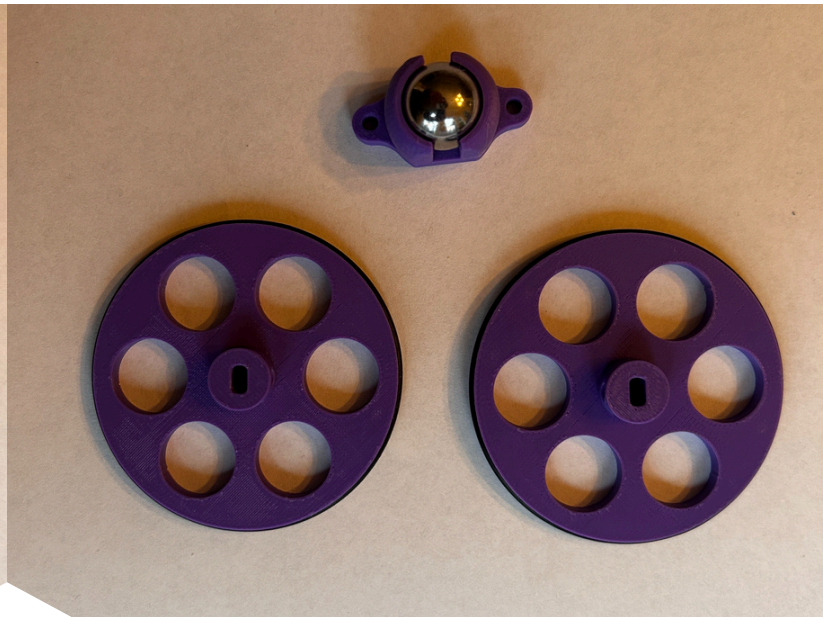
A s**ervo motor** is a small motor that can move to a specific position and hold that position precisely. Unlike regular motors that just spin continuously, a servo motor can turn to an exact angle you tell it to — like pointing a robot's arm or steering a car's wheels. It's great for making precise movements in robots, toys, or gadgets.

Getting Moving!

Step 3

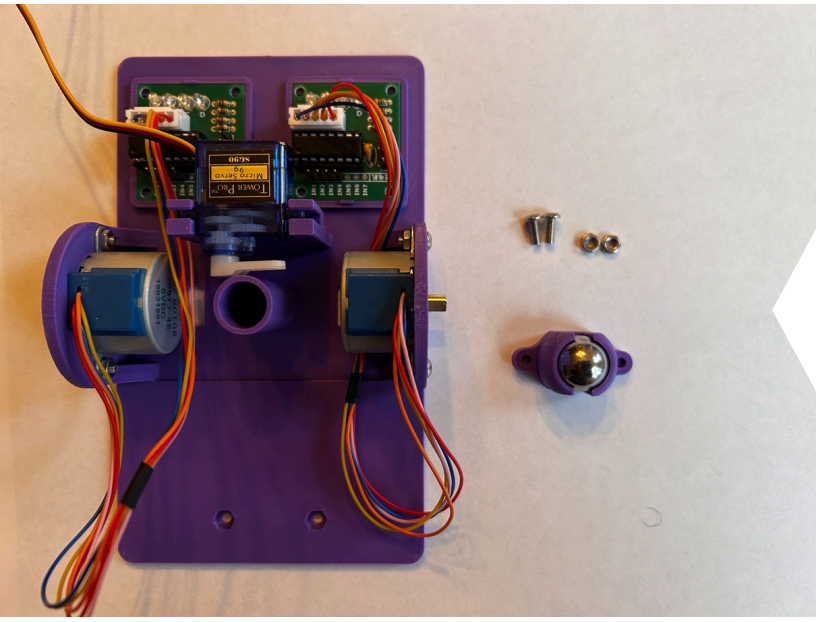
Now that we have all of the motors attached its time to add wheels and get this robot moving!



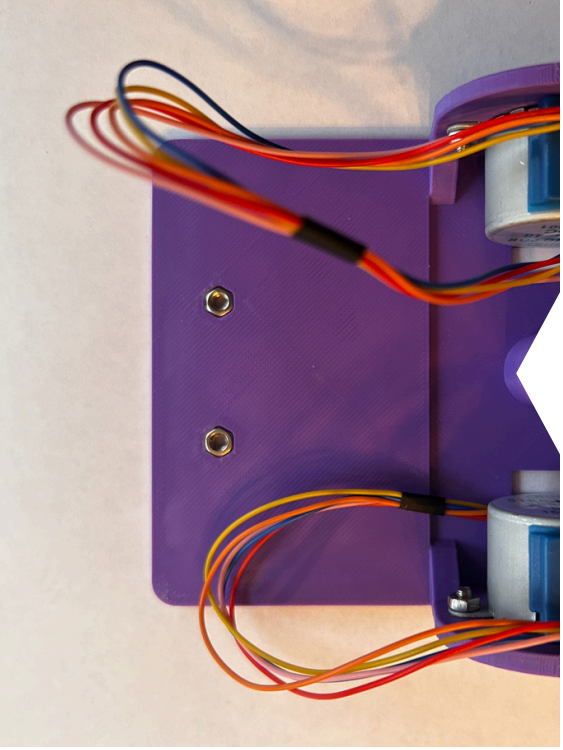


Collect these parts: wheels, o-rings, ball caster, steel ball bearing.

Push the steel ball bearing into the ball caster. Take the o-rings and wrap them around the wheels in the groove.

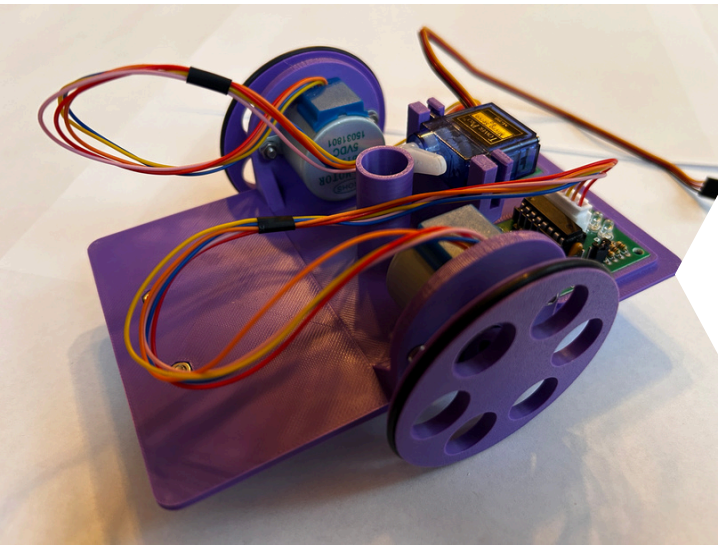


Collect these parts: ball caster and bearing (from last step), 2 8mm screws, 2 nuts



Place the nuts in the hexagonal slots towards the back of the chassis. Push down to ensure they stay in place.

Flip the robot over, line up the ball caster screw holes with the nuts, and screw the ball caster in securely.

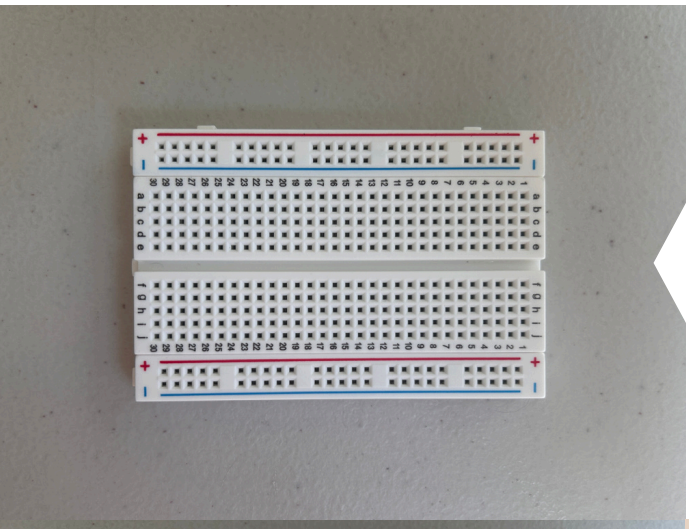


Flip the robot over again and push the wheels onto the motor shafts. Make sure that the motor shaft lines up with the wheels before pushing.

Breadboard

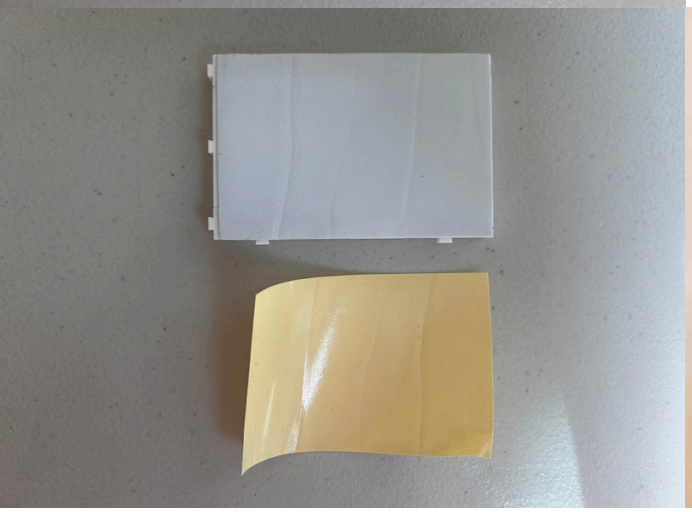
Step 4

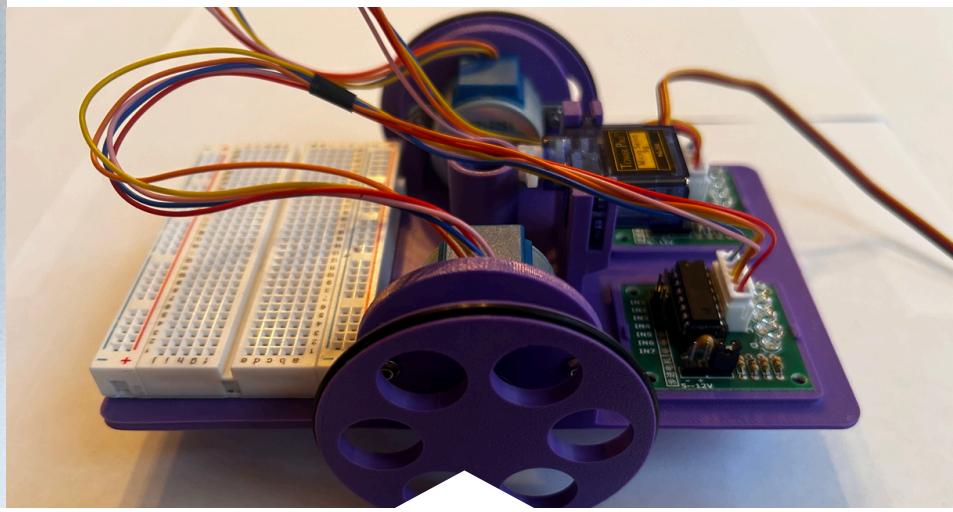
Next, we'll attach the breadboard, which is where all the electronic parts will connect to make the robot work.



Collect these part: breadboard, robot chassis

Peel the yellow protective backing off the breadboard.





Carefully align the breadboard over its space on the back of the chassis, making sure it fits fully on the surface without hanging over the edges. The blue and red breadboard lines should match the picture above.

Breadboard Explained

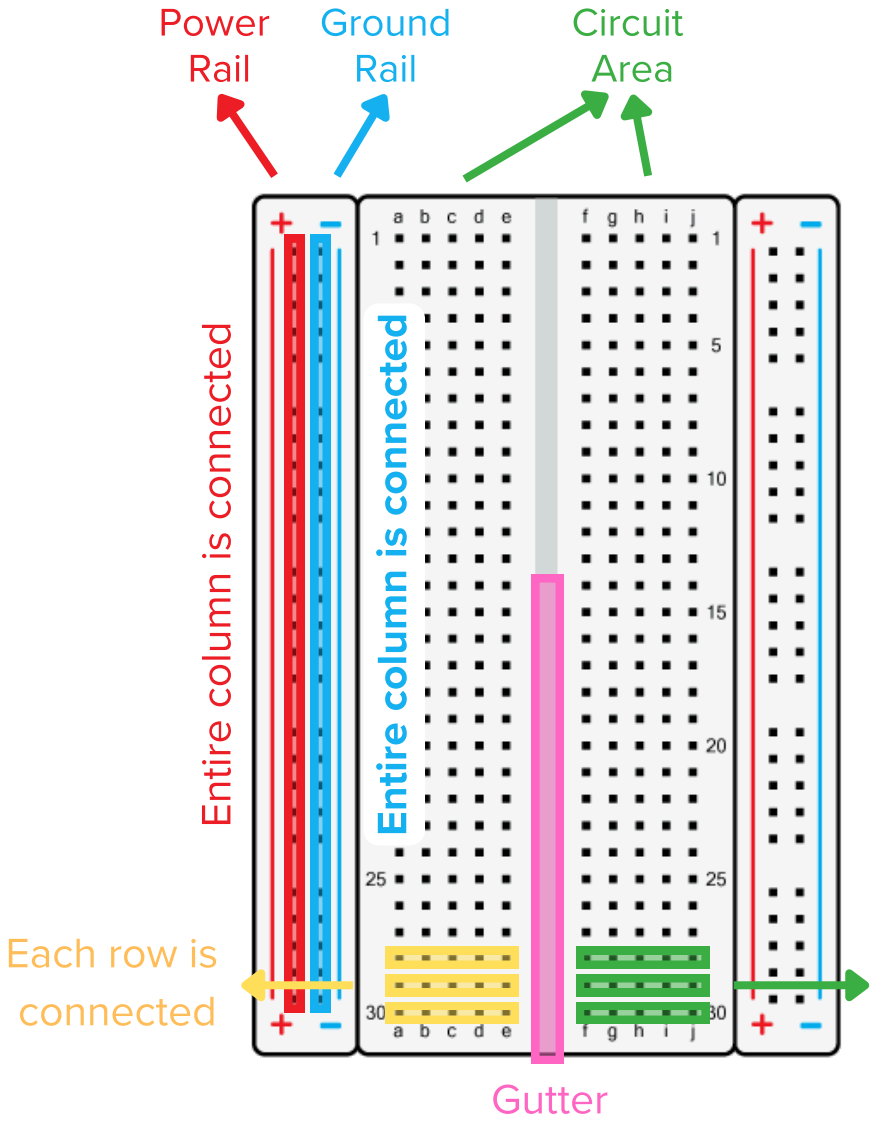
Have you ever wondered how a breadboard works or what it's useful for?

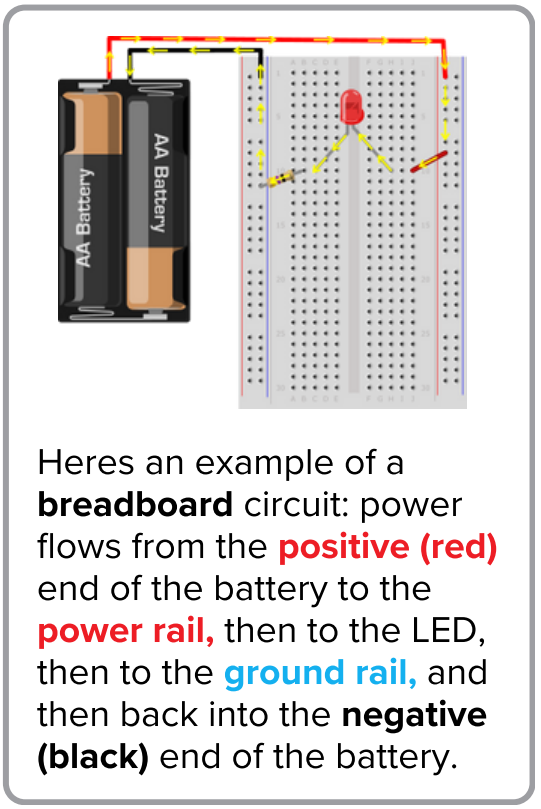
A **breadboard** is a tool used to easily connect electronic components without soldering. It lets you build circuits by simply plugging in wires and parts.

Power rail: A long row on the side of the breadboard that distributes power (+).

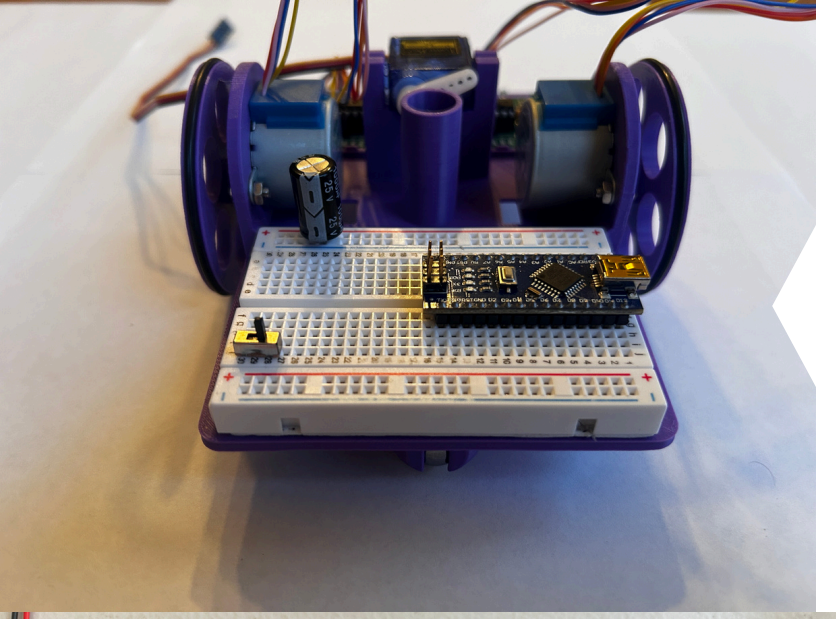
Ground rail: A matching row next to the power rail used to connect all parts to ground (-). Ground is where power flows back to, completing the circuit.

Circuit area: The middle section where you place and connect components to build your circuit.

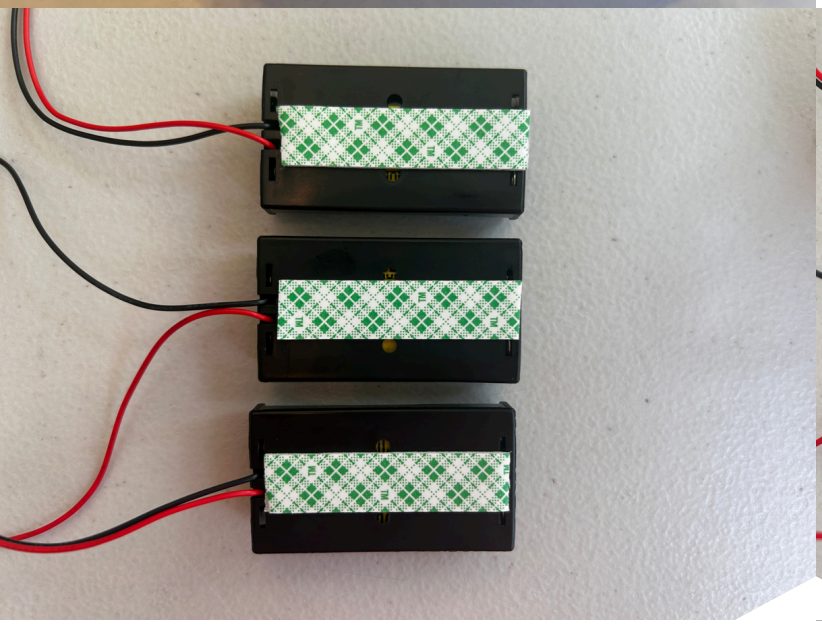


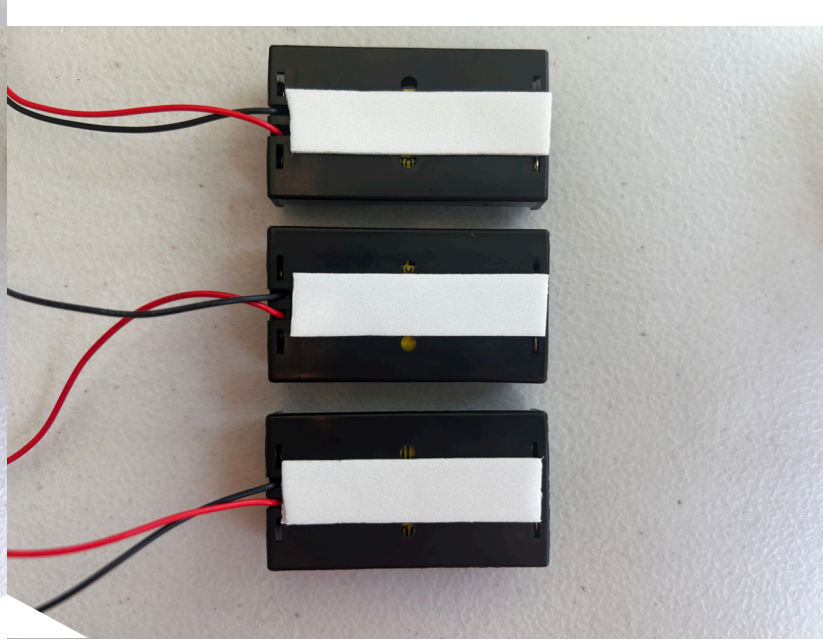


Heres an example of a **breadboard** circuit: power flows from the **positive (red)** end of the battery to the **power rail,** then to the LED, then to the **ground rail,** and then back into the **negative (black)** end of the battery.

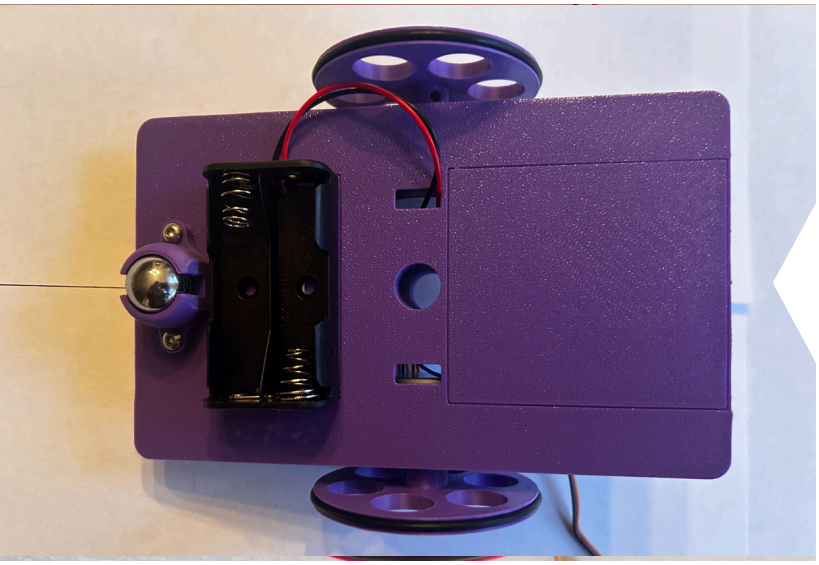


Connect the Arduino Nano, capicator, and switch into the breadboard as shown in the picture.



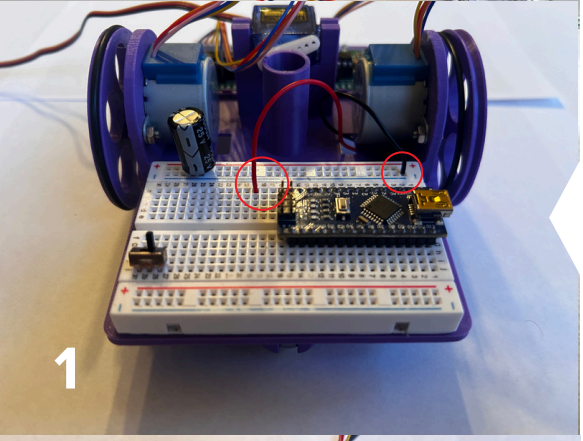


Collect the battery holders. Remove the protective backing off the adhesive foam on the battery holders.



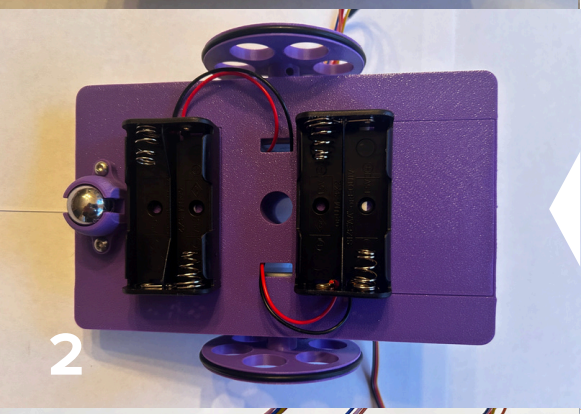
Stick one of the battery packs onto the bottom of the robot chassis as pictured.

Push the red and black battery holder wires through the top rectangular hole.

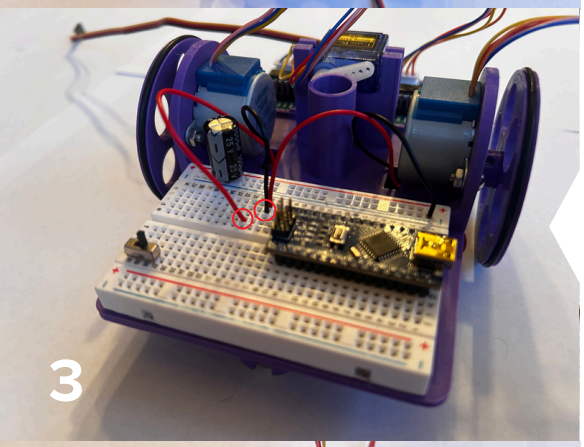


Flip the robot over again. Connect the red and black battery pack wires into the breadboard (Picture 1).

Tip: If your breadboard is not oriented the same way as the picture fix the rotation now before it gets stuck!

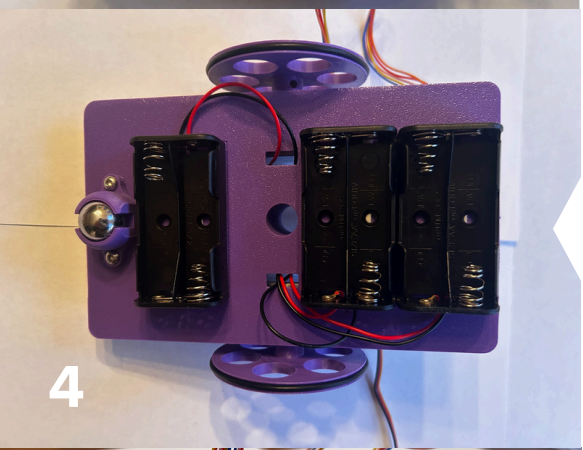


Attach the next battery pack onto the chassis as pictured. Thread the red and black wires through the bottom slot of the chassis (Picture 2).

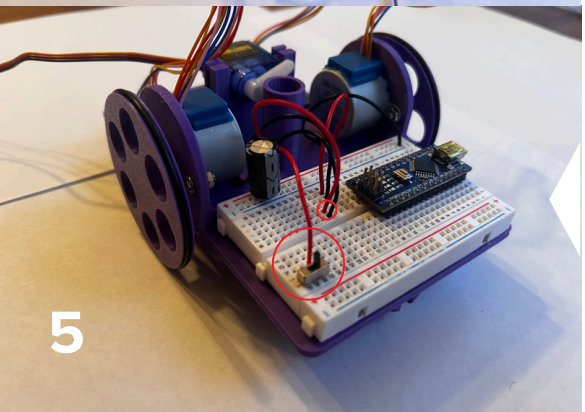


Connect the red and black wires from the battery pack into the breadboard as pictured. Ensure the black wire is connected to the same row as the red wire from the first battery pack (Picture 3).

Attach the final battery pack to the right side of the previous one. Make sure the wires are pointing in the same direction (towards the bottom). Then, thread the wires through the slot at the bottom of the chassis (Picture 4).



Connect the battery pack wires into the breadboard as pictured. The black wire should be connected to the same row as the red wire from the second batter pack. The red wire should be connected to the middle of the switch (Picture 5).



Wiring

Step 5

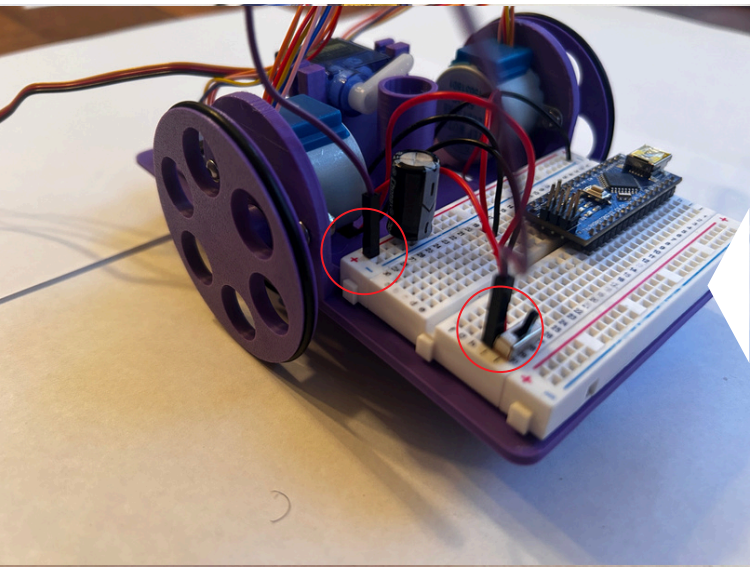
The last step to a completed robot is wiring everything together. Prepare your jumper wires and lets get wiring!

Male-to-Male Wires

Male-to-male wires have metal pins on both ends and are used to connect two things that have holes (sockets).

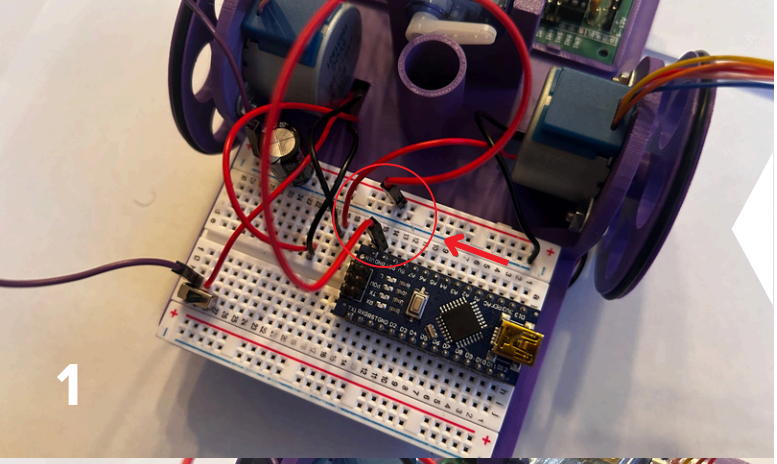
Male-to-Female Wires

Female-to-male wires have a hole on one end and a pin on the other, used to connect something with a pin to something with a socket.

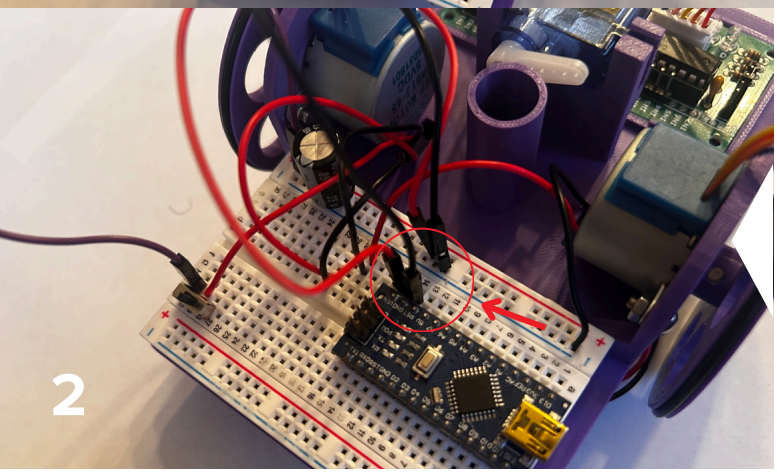


Grab a male-to-male wire. Connect one end to the left side of the switch. Connect the other end to the top power rail of the breadboard.

Tip: Your wiring does not have to look exactly the same as ours! Breadboards can work in many different ways. If you are struggling to connect a wire see if you can connect it somewhere else.

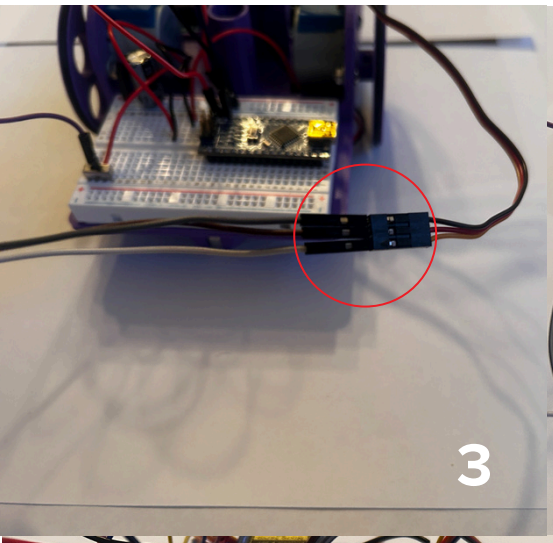


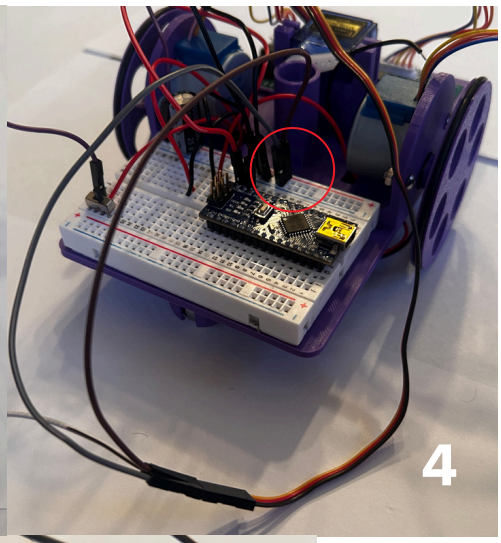
Grab a male-to-male wires. Connect one end to the power rail of the breadboard and the other end to the breadboard hole next to where it says VIN on the arduino (Picture 1).

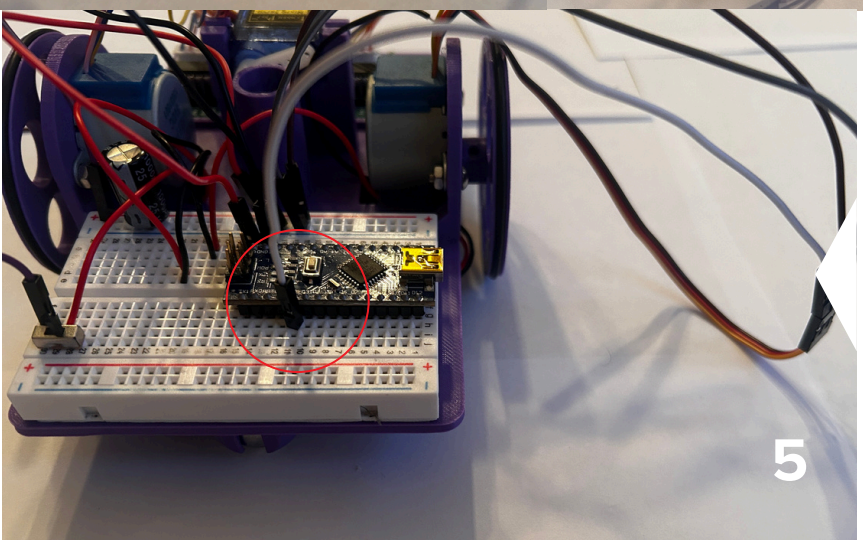


Using another male-male wire, connect one end to the ground rail and the other end to GND on the arduino (Picture 2).

Take three male-to-male wires and connect one end of each wire to the servo wire. Keep track of which color wire connects to each servo wire—for example, green wire to brown, yellow to red, and so on (Picture 3)



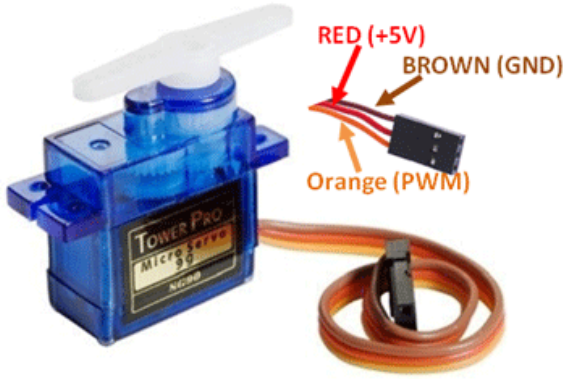




Connect the wire that corresponds to the servo's brown wire to the breadbord ground rail. Connect the wire that corresponds to the red wire to the breadboard power rail. (Picture 4) Finally, connect the wire that corresponds to orange to D2 on the arduino (Picture 5).

Servo Wiring Explained

We have now wired the servo but understanding why we connected wires where we did is very important.

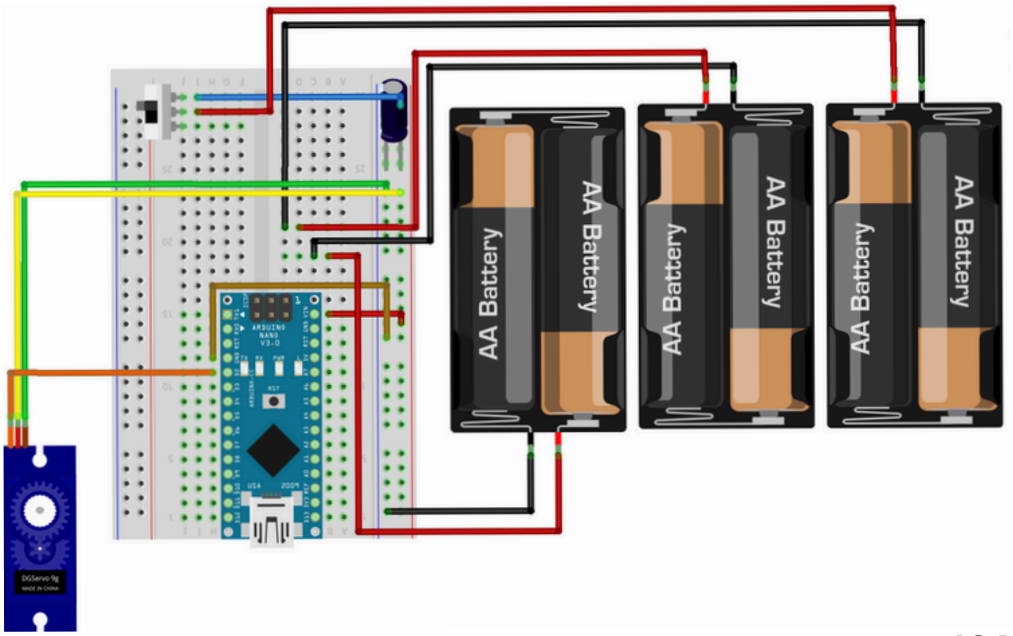


“Pinout” of the SG90 Servo Motor

The red wire on the servo supplies power to the motor, and our motor specifically needs 5 volts to run. The brown wire is the ground, which completes the circuit by letting the power flow back. The orange wire controls the motor’s position using PWM (Pulse Width Modulation) — basically this means it sends rapid pulses that tell the motor exactly what angle to move to.

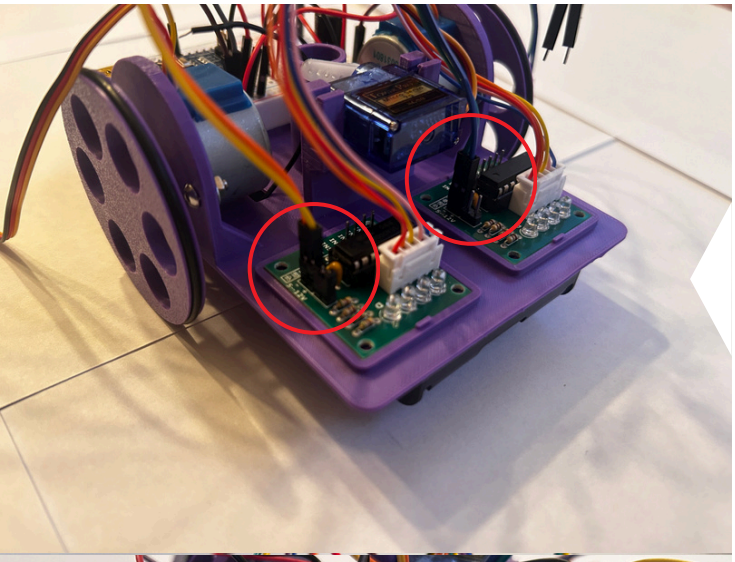
Wiring Check

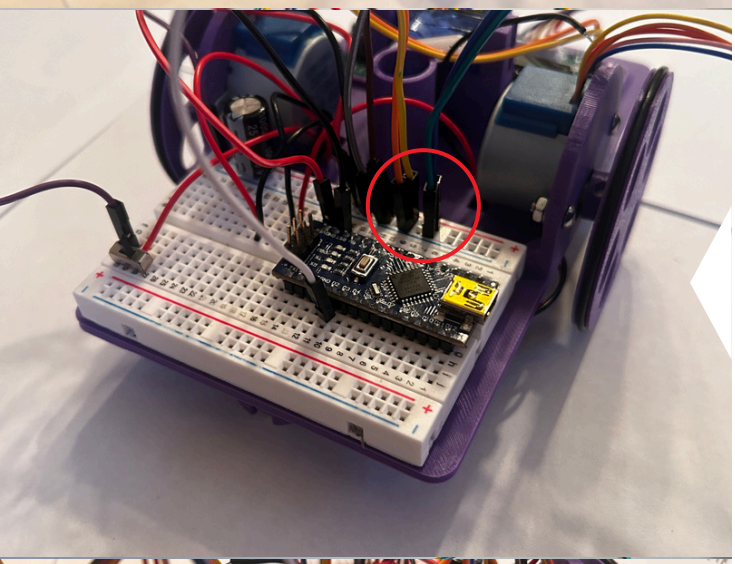
Before we keep going lets make sure your wiring is correct. Checking now will save you from headaches later.

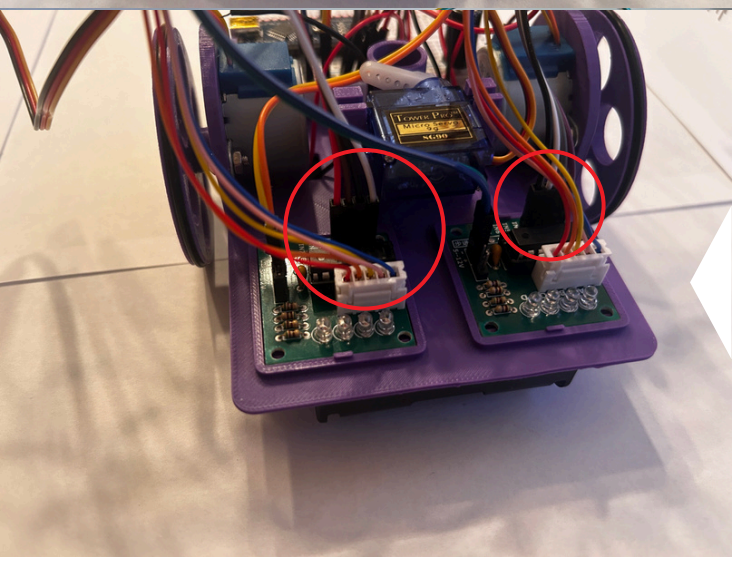


16 Wiring Check

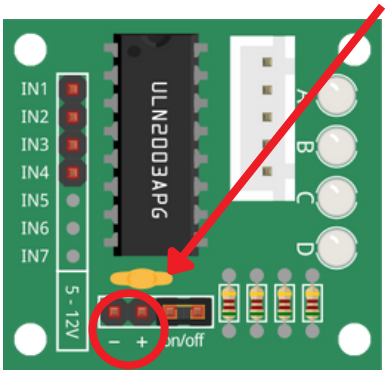
More Wiring





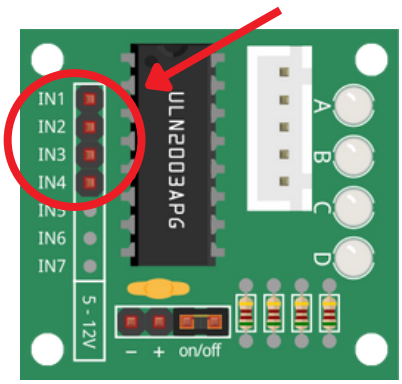


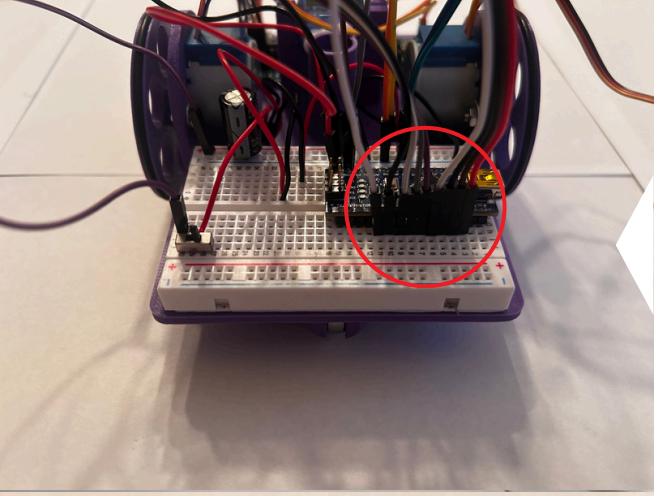
Grab 4 male-to-female wires. Connect the side of each wire with a socket to each one to the pins on the stepper driver boards labled with a “-” or “+”.

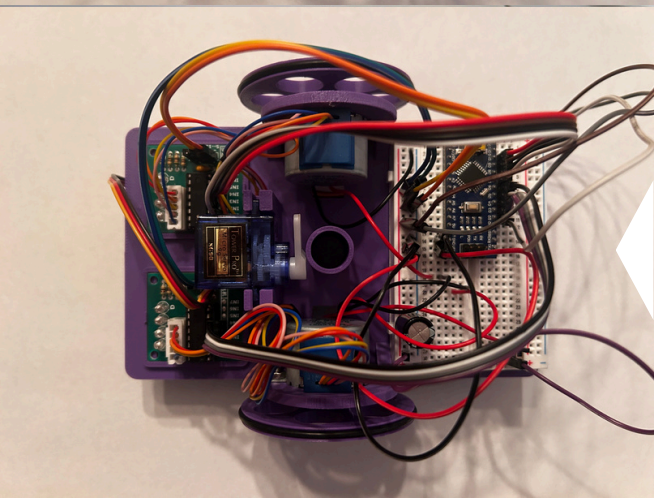


Connect the other end of the wires connected to the “-” pins to the ground rail of the breadboard. Connect the other end of the wires connected to the “+” pins to the power rail of the breadboard.

Grab the remaining 8 male-to-female wires. Connect the side of each wire with a socket to the pins on the stepper driver board labled IN1-4.







Rotate the robot so that the breadboard is closest to you.

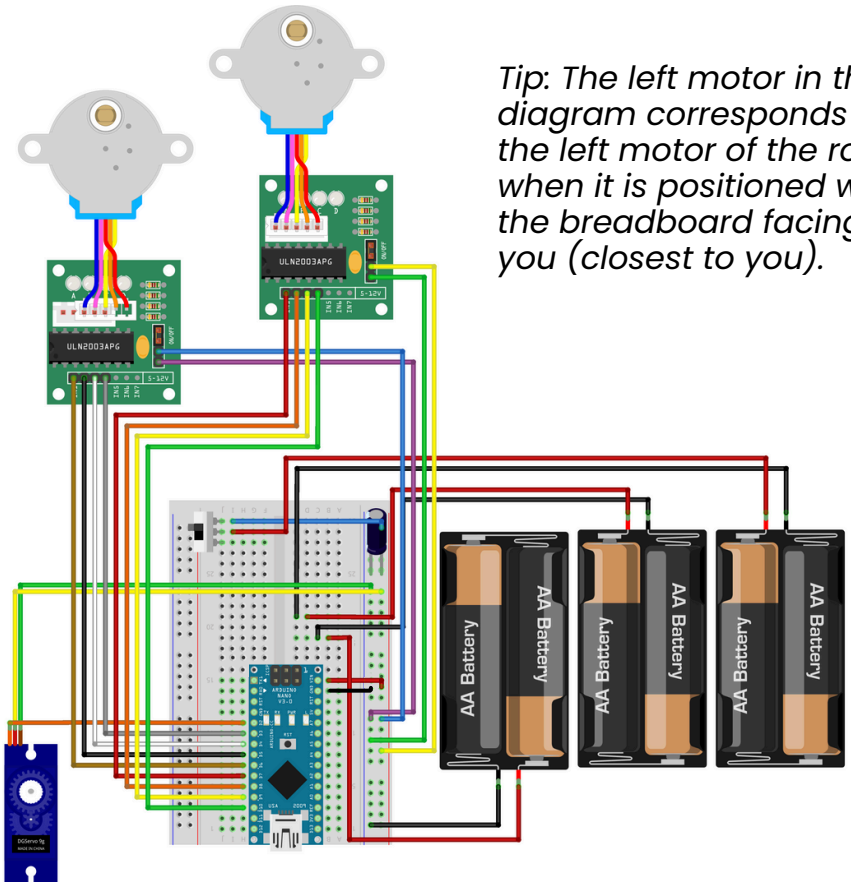
Starting with the left driver board. Connect the wire connected to IN1 to D6 on the arduino, connect IN2 to D5, IN3 to D4, and IN4 to D3. **With** the right driver board. Connect IN1 to D7, IN2 to D8, IN3 to D9, IN4 to D10.

You have completed assembling this RISE kit, congratulations!

Check your wiring before proceeding to the next section.

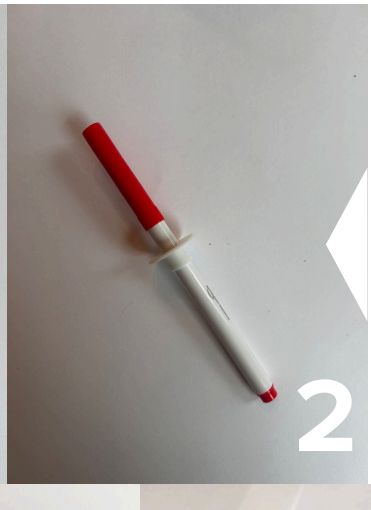
Wiring Check

Before we can start programming lets make sure all the wiring is correct.

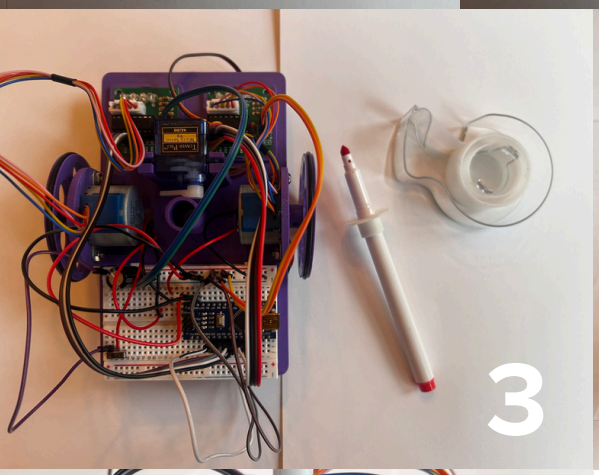


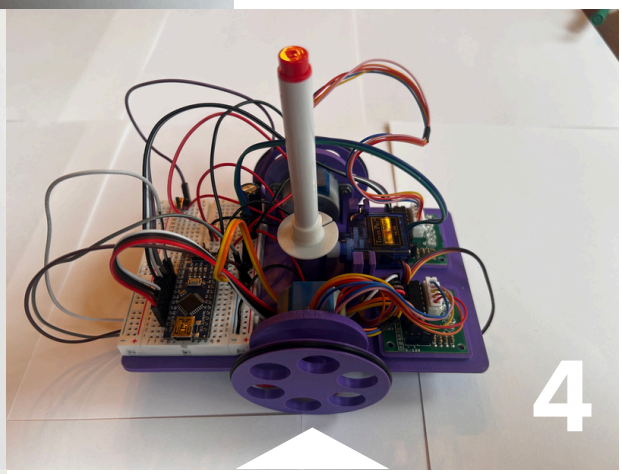
Tip: The left motor in the diagram corresponds to the left motor of the robot when it is positioned with the breadboard facing you (closest to you).

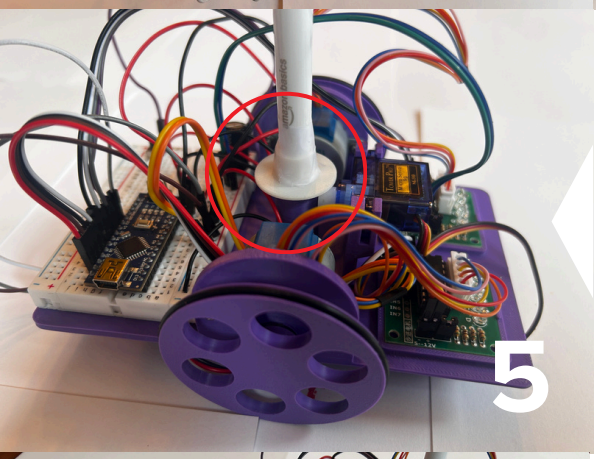
Using the Robot

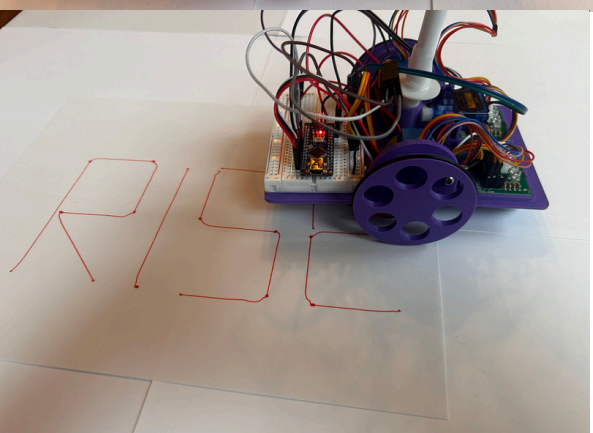


Grab a fine tip marker and a pen collar (Picture 1). If one doesn't fit trying using the other. Fit the collar onto the marker (Picture 2). Removing the cap can help.









Grab the robot, marker, and tape. (Picture 1)

Position the marker in the center marker hole. Move the collar so that the pen contacts the paper and the pen collar sits on the chassis (Picture 2). **Tape** the collar to the marker (Picture 3.

Your robot is now ready to draw. You can remove the marker or put the cap on while storing it.

Programming

Now that we have our robot assembled, let's learn how to program it!

Coding Environment

Step 1

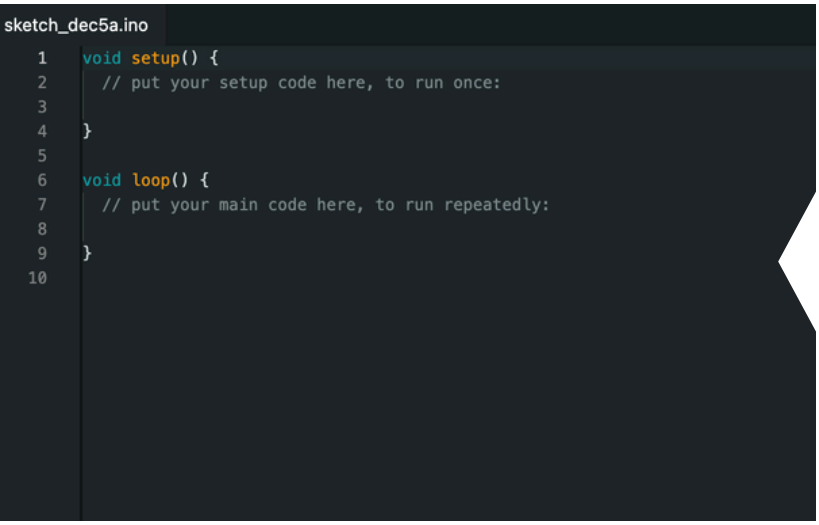
First we need to install the Arduino IDE. This is where we'll write our code. Head over to **https://www.arduino.cc/en/software** and download the IDE. Once you have it downloaded open the application.

Once we open it up, we're going to get something that looks like this.

What Does It Do?: Arduino IDE

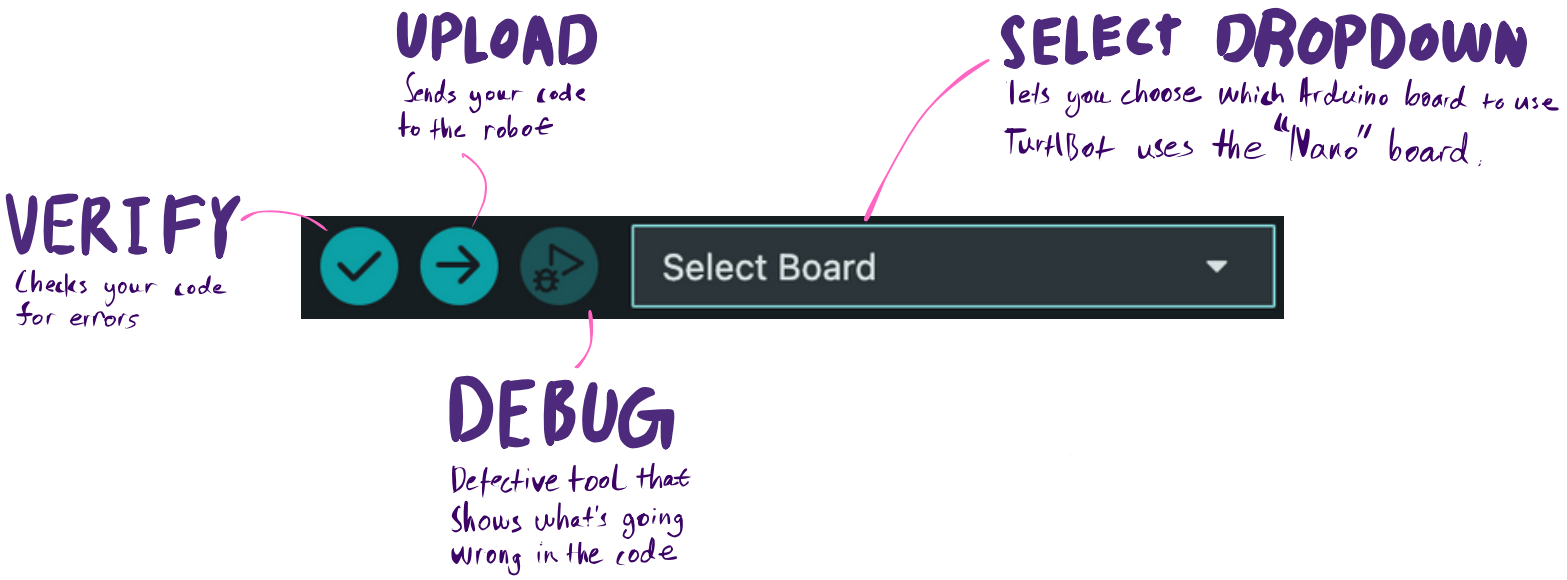
The Arduino IDE is the program we use to write code for our robot in **C++**. With it, we can type our instructions, upload them to the robot's Arduino board, and watch it follow those commands in real time. It makes it easy to test ideas, fix mistakes, and control how our robot moves and reacts.

Let's take a closer look.



This is the **editor.** It's where we write the code that tells our robot what to do. Anything we type here gets sent to the robot so it can run our instructions.

Buttons, Buttons, Buttons!

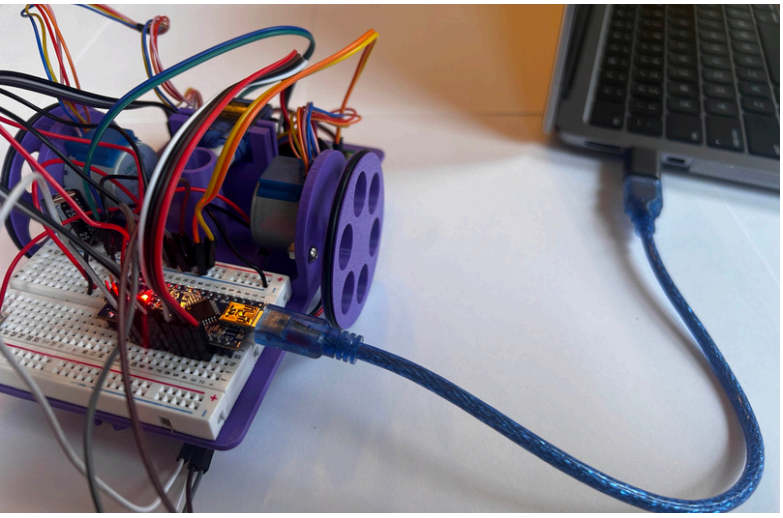


Make sure to familiarize yourself with the coding environment!

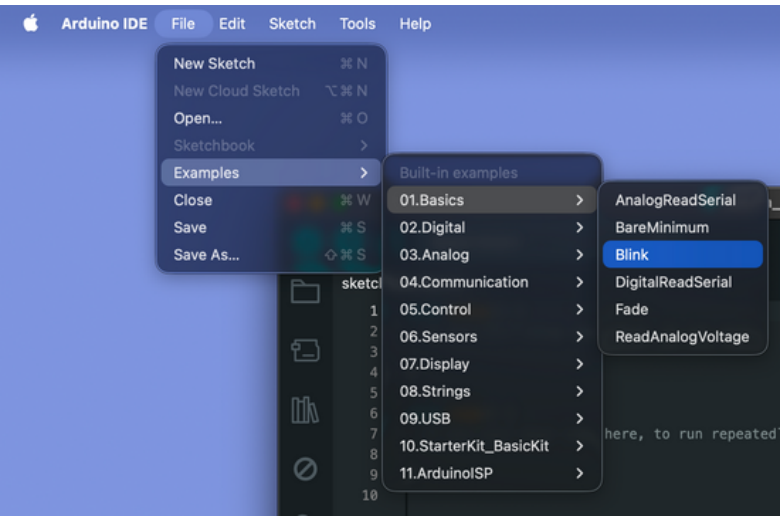
Coding Environment

Step 2

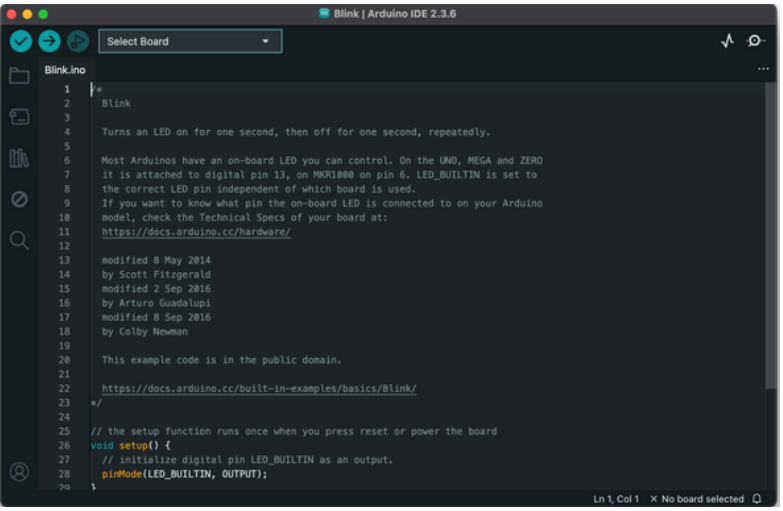
Now, we need to make sure that our coding environment is properly set up! To do that, we're going to run our first piece of code!



Plug the blue USB-A to Mini-B USB cable into TurtlBot and your computer.



If you're on **MacOS** or **Windows**, click **File > Examples > 0.1 Basics > Blink**.



A new window will pop up containing code that has already been written for you.

Click the run button.

If you see **blinking** lights on you're robot, you're good to go!

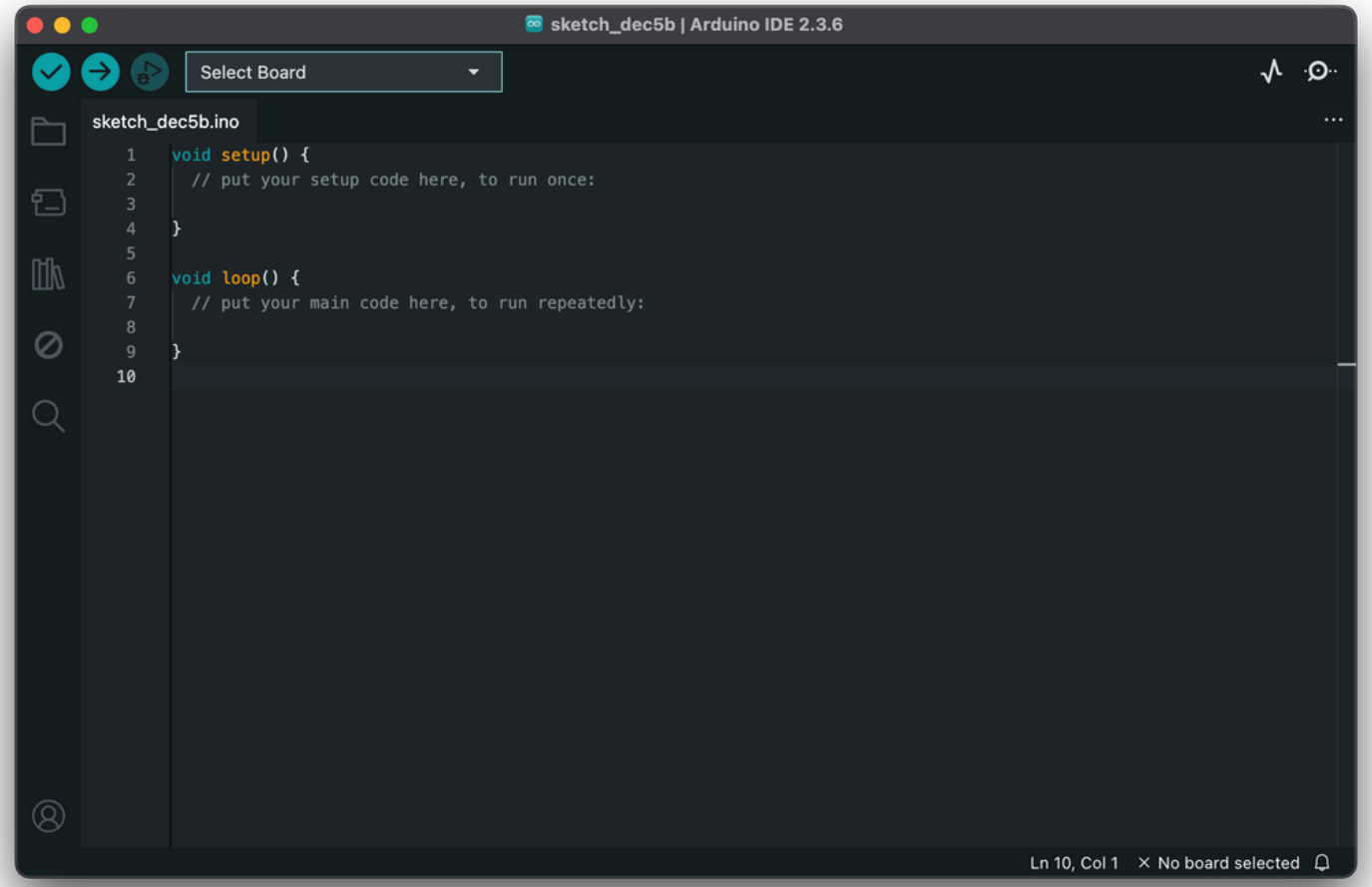
Tech Corner:

If you find that your blink code didn't work, go back to the **Assembly** part of the instructions and check your wiring!

Defining Variables

Step 1

Now that we've finished setting up our coding environment, it's time to write some code! First, open a new file by going to **File > New Sketch**.



In this part, we'll **define** all the **variables**, which is where we tell our robot what parts we're using and how they should behave. Think of it like giving your robot its morning instructions before it starts moving! This is where we connect things like our motor, LEDs, or sensors, and let the robot know which pins they're plugged into. Once the setup is ready, our robot will know exactly what it needs so it can follow the main code we write next.

Step 2

We have now just created a new .ino file to work from. This file will contain all the code that we will need to program our robot. Now, we will walk through the initial variable definitions that are required to make our code work with TurtlBot.

draw_bot.ino

```
1   #include <Stepper.h>
2   #include <Servo.h>
3   #include <math.h>
```

Import Stepper.h, Servo.h, and math.h as shown in the code above. This packages will allow us to communicate with the motors and perform calculations to draw with our robot.

draw_bot.ino

```
4   const int stepsPerRevolution = 2048 ;
5   const float trackCM = 11.0 ;
6   const float wheelDiameterCM = 6.2 ;
7   const float wheelCircumferenceCM = PI *
    wheelDiameterCM;
8   const float stepsPerCM = stepsPerRevolution
    wheelCircumferenceCM;
9   const float turnCorrection = 1.01 ;
```

Assign these variables to match your robot's measurements. This ensures your robot knows its own size so it can drive and turn the way you expect.

Define which pins each stepper is plugged into. Each motor has four wires, and we give each one a number so the robot knows how to move its wheels. The direction numbers help make sure both motors spin the right way!

draw_bot.ino

```
10  #define STEPPER1_DIR 1
11  #define STEPPER2_DIR -1
12
13  #define STEPPER1_IN1 3
14  #define STEPPER1_IN2 4
15  #define STEPPER1_IN3 5
16  #define STEPPER1_IN4 6
17
18  #define STEPPER2_IN1 7
19  #define STEPPER2_IN2 8
20  #define STEPPER2_IN3 9
21  #define STEPPER2_IN4 10
```

draw_bot.ino

```
22  #define SERVO_PIN 2
23
24  Stepper stepper1(stepsPerRevolution,
    STEPPER1_IN1,
    STEPPER1_IN3,
    STEPPER1_IN2,
    STEPPER1_IN4);
25  Stepper stepper2(stepsPerRevolution,
    STEPPER2_IN1,
    STEPPER2_IN3,
    STEPPER2_IN2,
    STEPPER2_IN4);
26
27  Servo myServo;
```

Finish **defining** the pins for the servo motor.

Create the stepper1 and stepper2 **objects**.

Create the myservo **object.**

We have now finished defining all the initial variables needed to code our robot. Next, we will move onto creating functions that tell TurtlBot how to move and draw!