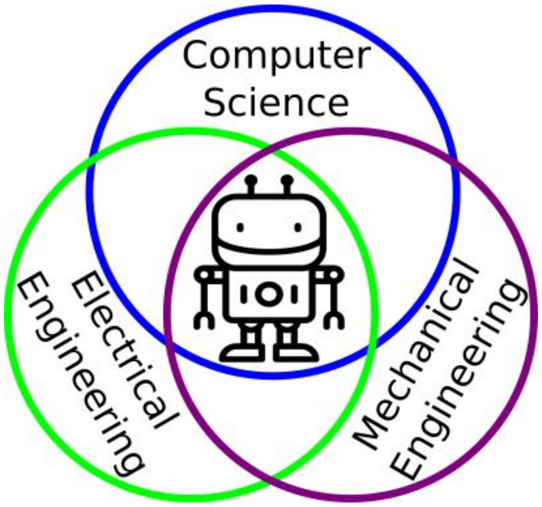
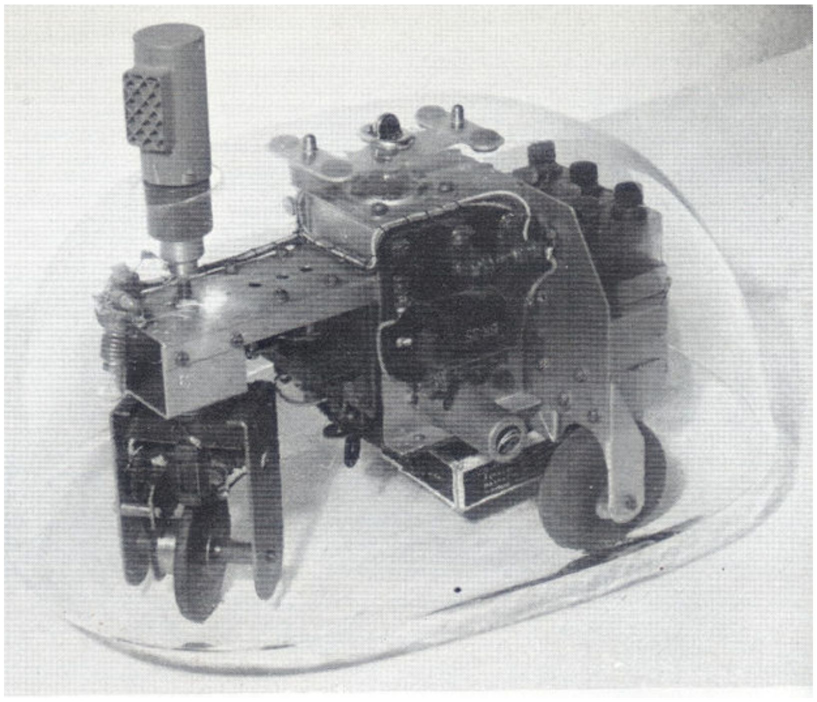
Open Source Turtle Robot (OSTR)

by MakersBox



Robotics is the exciting intersection of a number of engineering fields including mechanical engineering, electrical engineering, and computer science. This project was designed as the basis for a two-day workshop for high school age students to introduce them to engineering principles and giving them a jumping off point for exploring their interests further. There is something for everyone here, even if it is just for the art it creates.



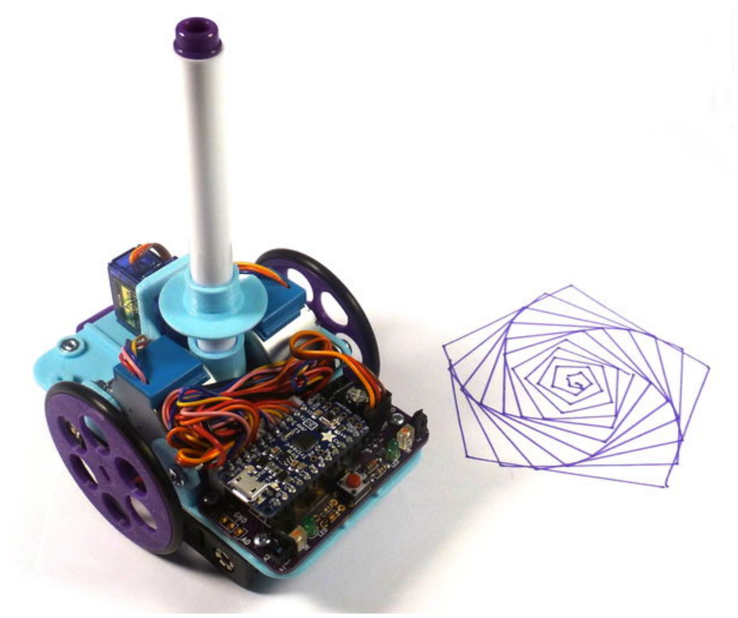
What is Turtle Robot? The concept can be traced back to William Walter’s robotics work in the 1940s which investigated complex behaviors in simple systems. Turtle robots are generally slow moving with tight turning radiuses and can trace a design that shows their behavior over time. They make excellent teaching aides in that their programmed output can be seen visually. A number of programming languages have “Turtle Graphics” built in. Turtle graphics can also be used to investigate advanced topics like L-systems or affine geometry

OSHW US0125 What is Open Source? If you can look past licences and lawyers, Open Source is a philosophy that ideas should be shared, and if you use an idea you attribute the originator, and if you improve on it, you should share your improvement. Every part of this project, from the circuit board to the 3D designs, are available for study, replicating, and improving. Not only that, but it was designed and programmed used Open Source programs like Arduino and KiCad. All you would have to do is use a computer running Linux (an Open Source operating system) and print the parts on a RepRap based 3D printer (an Open Source project), and you would have Open Source Turtles All the Way Down. This project is certified as Open Source Hardware by the OSHWA.

This project is just the latest iteration of my “Low-Cost, Arduino Compatible Robot”, which in turn is based on the work of others (as indicated in the parts section). Improvements include a printed circuit board for ease of construction and robustness, infrared sensors for obstacle detection, and a more compact and stiff chassis for more accurate movement and lower printing cost. Rewiring port assignments and firmware improvements help remove stuttering caused by Arduino’s *digialWrite()* inefficiencies.

In keeping with the engineering fields we are exploring, this project is broken into three distinct sections:

- Electrical (EE: PCB assembly and testing),
- Mechanical (ME: chassis assembly), and
- Firmware (FW: calibration and programming).



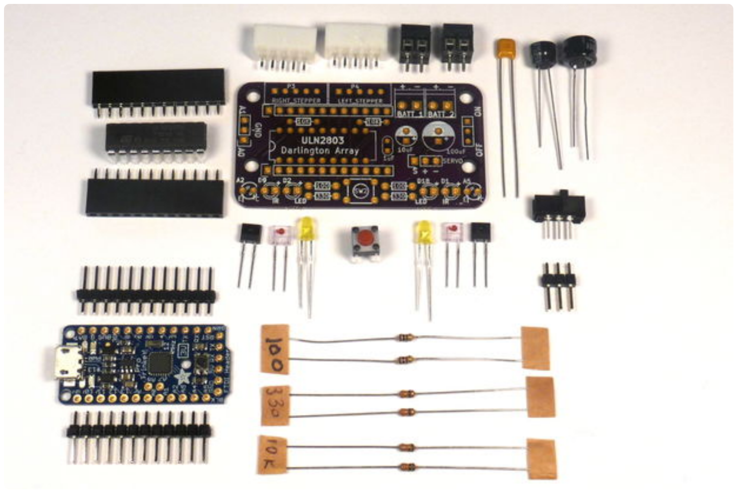
https://youtu.be/ExhSeAjVGYc

Step 1: Electrical Engineering (EE): Parts

The current Bill of Materials is located at: https://github.com/aspro648/OSTR/. A number of components like resistors and capacitors are generic, so substitutions are acceptable. In general, the cost is about $60-80.

A kit for this project is available on Tindie.com.

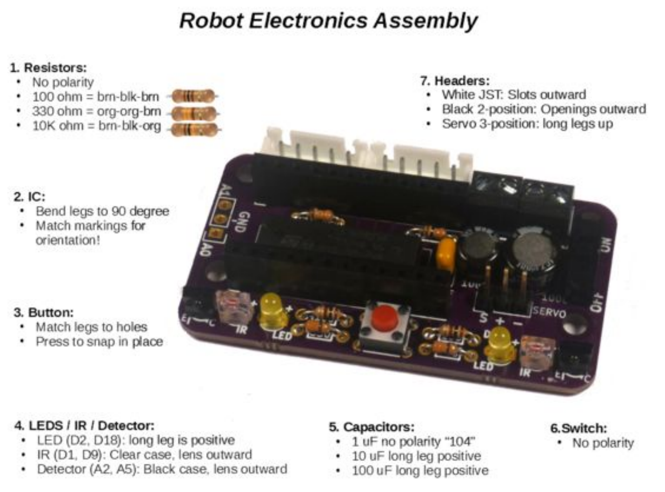
Purchasing the kit will save you the time and expense of ordering from several different vendors and avoid the minimum PCB order premium. You will also be helping me develop and share other projects in my workshops!



Step 2: EE: TLDR;

A well designed circuit should be "build-able" by an experienced hobbiest based on the bill of materials and the PCB silkscreen. For those of you who like to charge ahead, this graphic should be enough to get you past any quirks in the design.

For the rest of us mere mortals, follow along and I'll discuss assembly in detail.



Robot Electronics Assembly

1. Resistors:
- No polarity
- 100 ohm = brn-blk-brn
- 330 ohm = org-org-brn
- 10K ohm = brn-blk-org

2. IC:
- Bend legs to 90 degree
- Match markings for orientation!

3. Button:
- Match legs to holes
- Press to snap in place

4. LEDS / IR / Detector:
- LED (D2, D18): long leg is positive
- IR (D1, D9): Clear case, lens outward
- Detector (A2, A5): Black case, lens outward

5. Capacitors:
- 1 uF no polarity "104"
- 10 uF long leg positive
- 100 uF long leg positive

6.Switch:
- No polarity

7. Headers:
- White JST: Slots outward
- Black 2-position: Openings outward
- Servo 3-position: long legs up

Step 3: EE: Resistance Is Essential.

I like starting with the resistor because:

1. They are relatively heat resistant while you are getting into your soldering groove and the iron is coming up to temp,
2. They have no polarity, so orientation is not critical, and
3. They are the lowest component on the board so sit tight when soldering when the board is flipped on it's back.

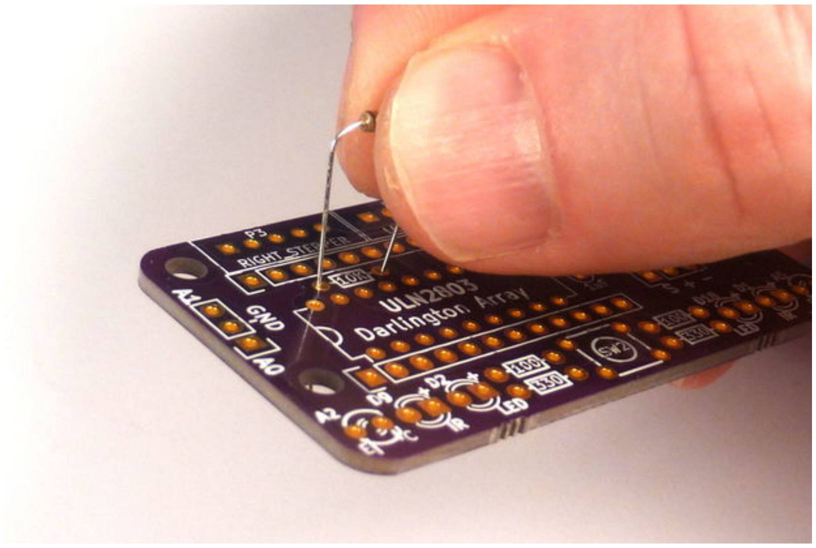
Since these are our first components, we'll cover them a little more in depth starting out. There are three pairs of different values:

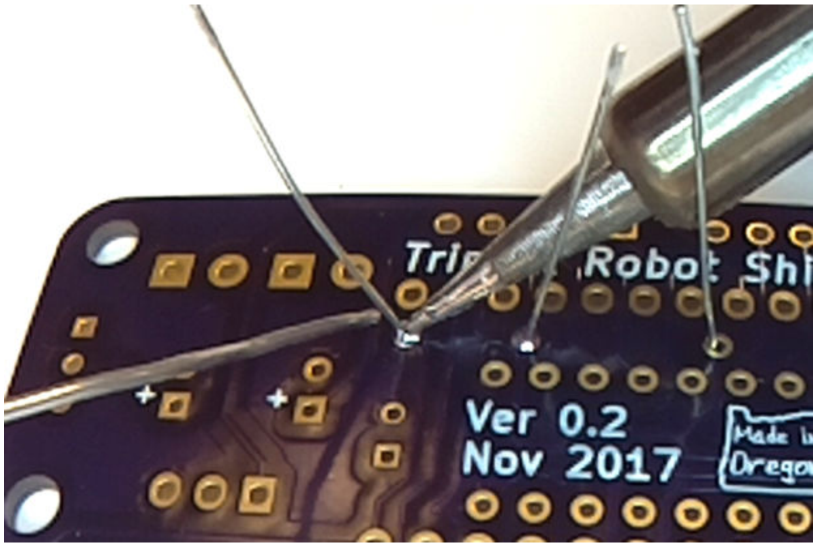
- 10K ohm for sensors [brown - black - orange - gold]
- 330 ohm for LED current limiting [orange - orange -brown -gold]
- 100 ohm for IR current limiting [brown - black - brown - gold]

Check out www.resistorguide.com/resistor-color-code/ if you want to learn more about reading resistor color codes.

You can do one at a time, or all six at once.

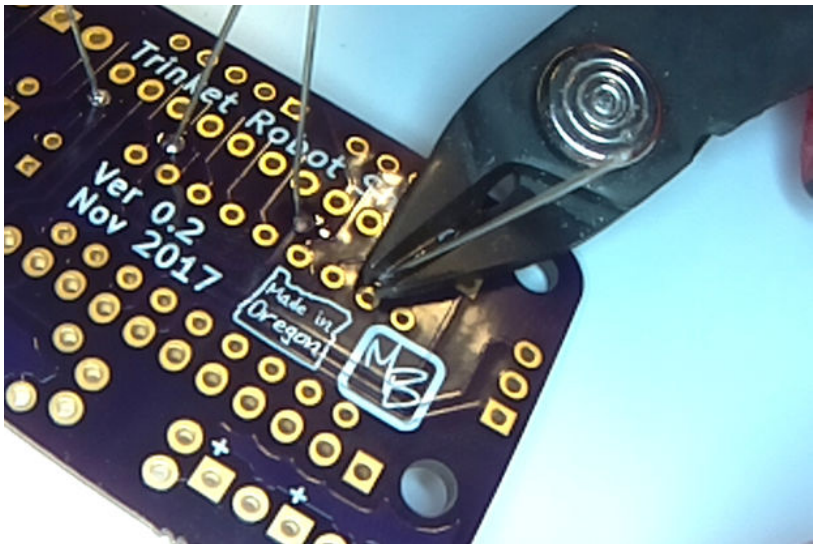
- Bend the leads to the width of the pads and insert the resistor.

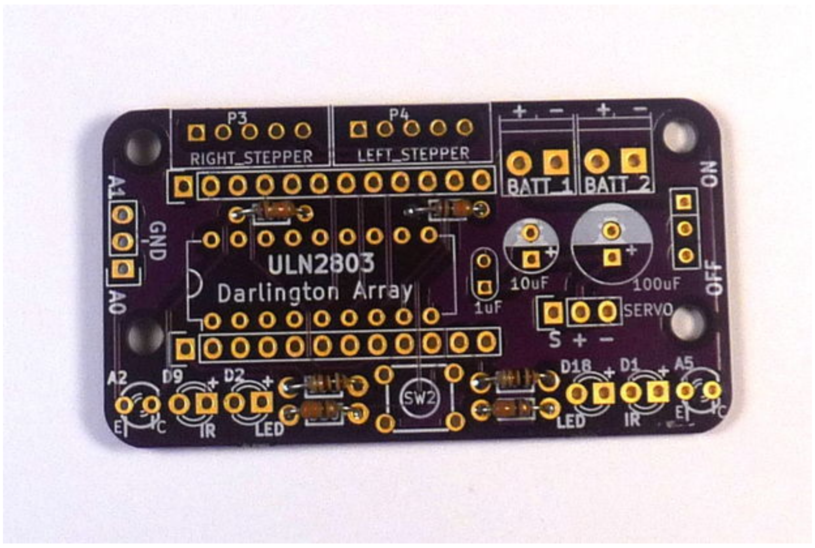


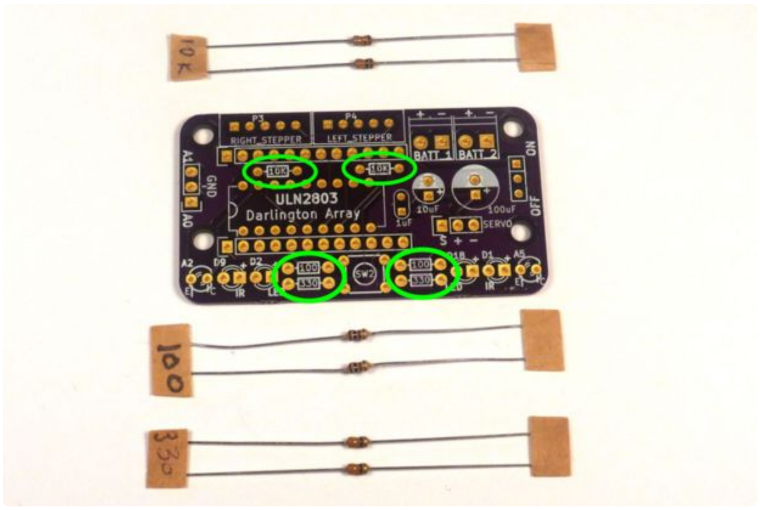


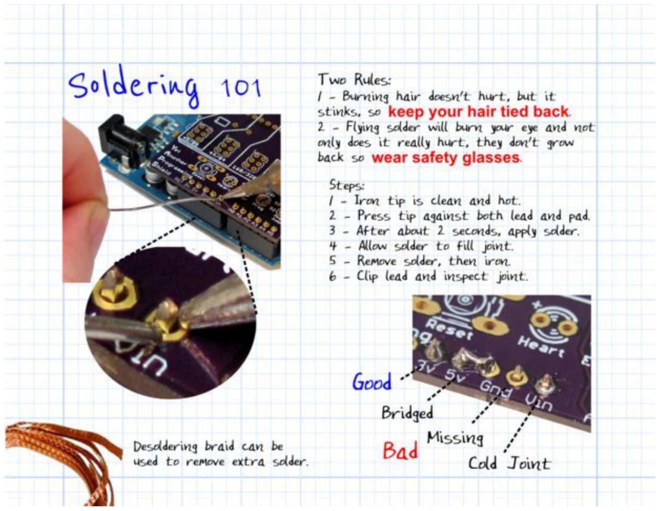
- Flip the board over and heat both the pad and lead for about two seconds.
- Introduce solder, and maintain heat until the solder melts and flows around the pad.
- Remove the solder, and then the heat, in that order.
    - If your solder gets stuck, just re-heat the joint.

- Trim the leads with flush cuts.







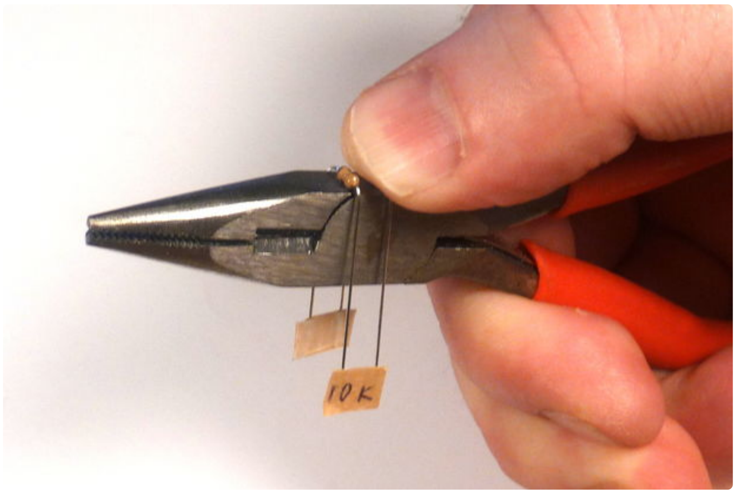


Soldering 101

Two Rules:
1 - Burning hair doesn't hurt, but it stinks, so **keep your hair tied back**.
2 - Flying solder will burn your eye and not only does it really hurt, they don't grow back so **wear safety glasses**.

Steps:
1 - Iron tip is clean and hot.
2 - Press tip against both lead and pad.
3 - After about 2 seconds, apply solder.
4 - Allow solder to fill joint.
5 - Remove solder, then iron.
6 - Clip lead and inspect joint.

Desoldering braid can be used to remove extra solder.



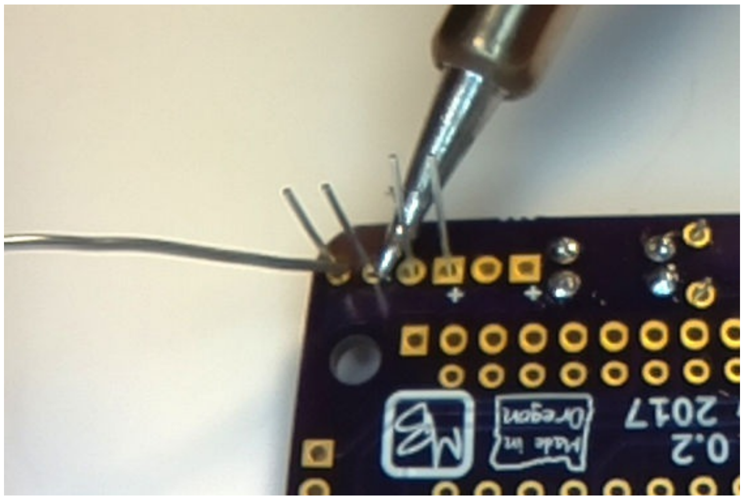
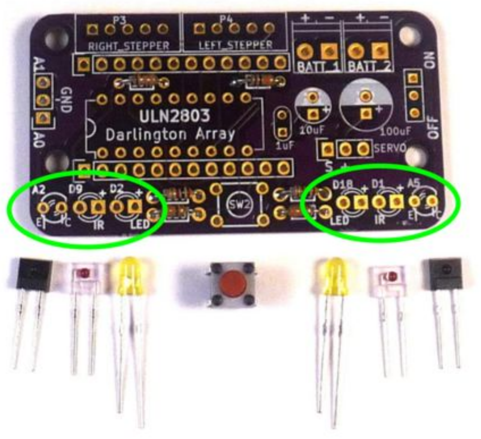
Step 4: EE: Eyes!

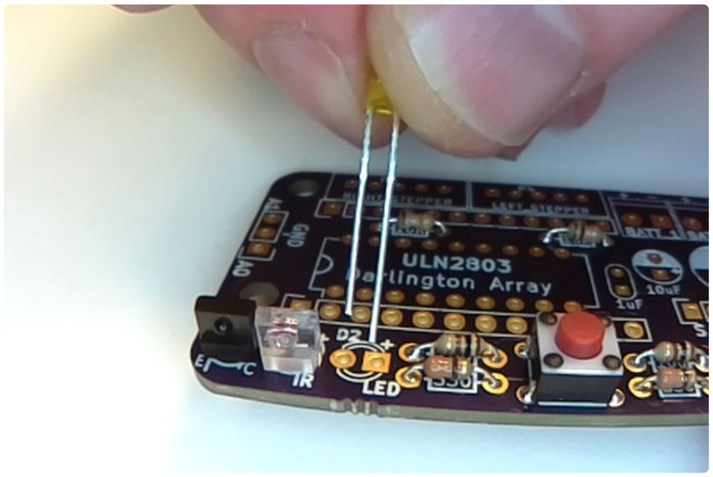
The front end of the robot contains two sets of devices for detecting and indicating obstacles, one for the left side, and one for right. From the outer edge working inward, we have an:

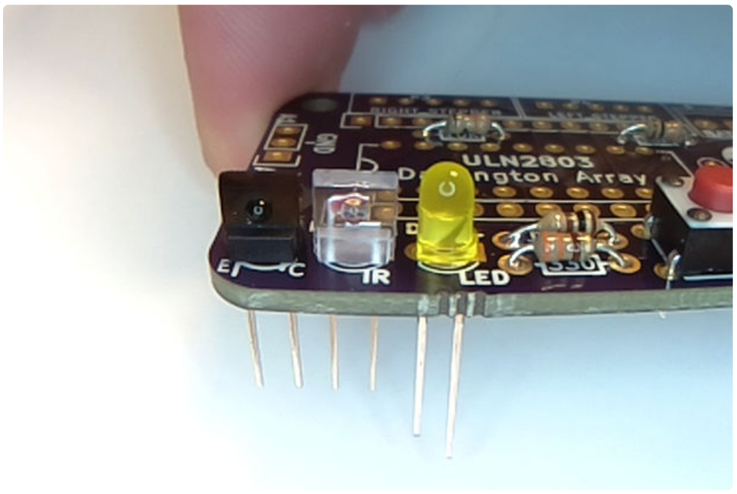
- IR Detector (black),
- IR Emitter (clear), and
- LED (colored).

They each have polarity, so must be put in the correct orientation. The long leg of the LED is the anode (positive), and goes in the pads marked "+". The others go with the lenses facing outward.

- Solder one lead of each component.
- Verify they are oriented and seated correctly.
- Trim the excess lead with flush-cut pliers.

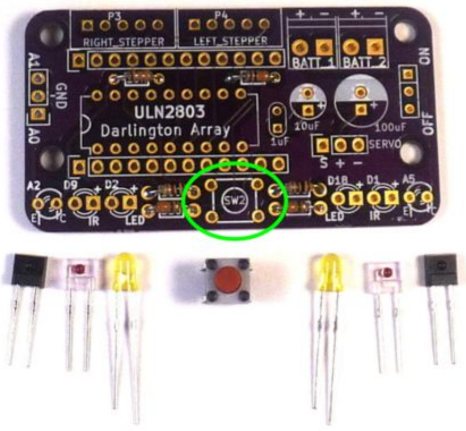
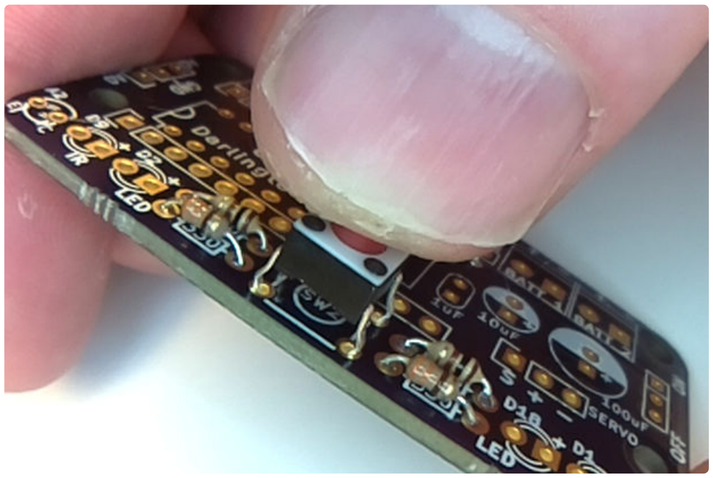
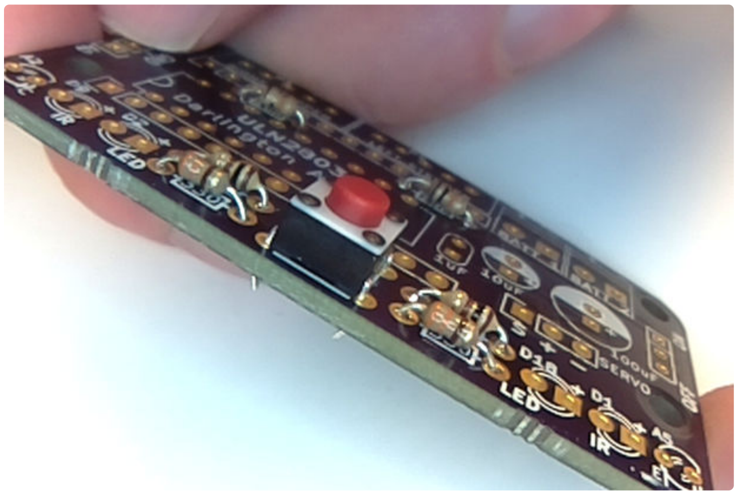







Step 5: EE: Button

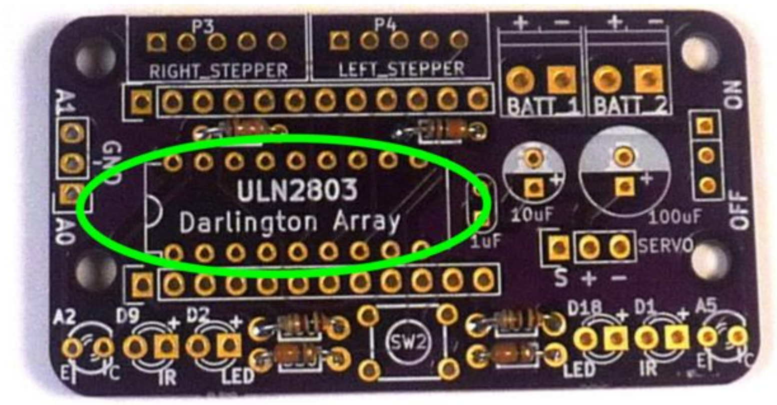
We have a single button to use for input. Match the holes with the leads and gently press them in. Solder them in place and clip the leads.

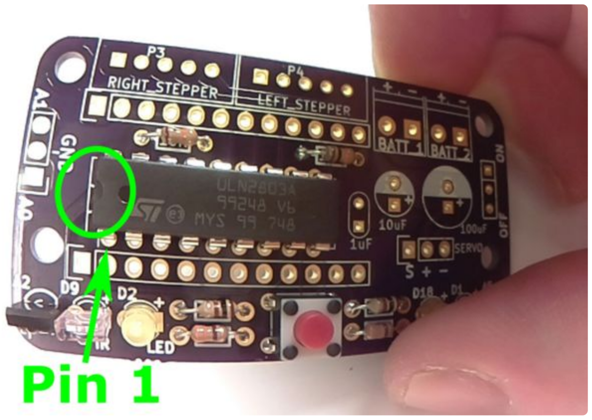




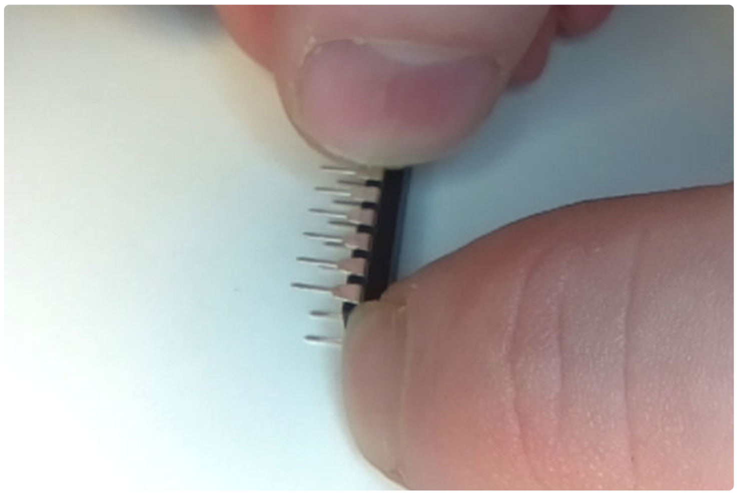
Step 6: EE: Darling Darlington

The single IC on the board is a Darlington array, which is used to control the power going to the motors. From the manufacture, IC pins are splayed slightly outward and need to be bent to 90 degrees to fit in the holes.

- Place the IC on a flat surface and bend each side inward gently.
- **Match the divot on the end marking the side with pin one** with the silk screen.
- Solder two pins in place and check the alignment. It becomes exponentially harder to fix after more than two pins are soldered.
- If you are ready to commit, solder the remaining pins in place.



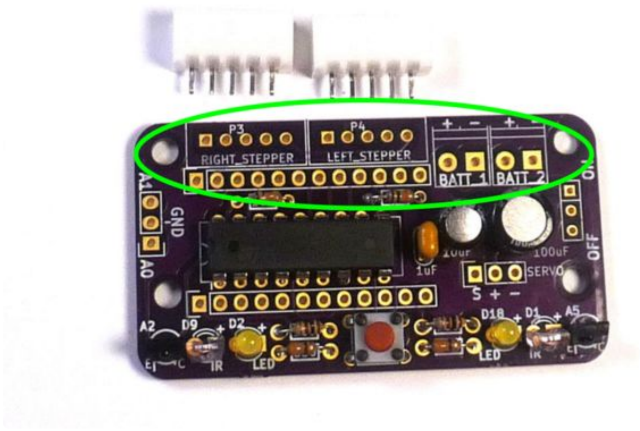
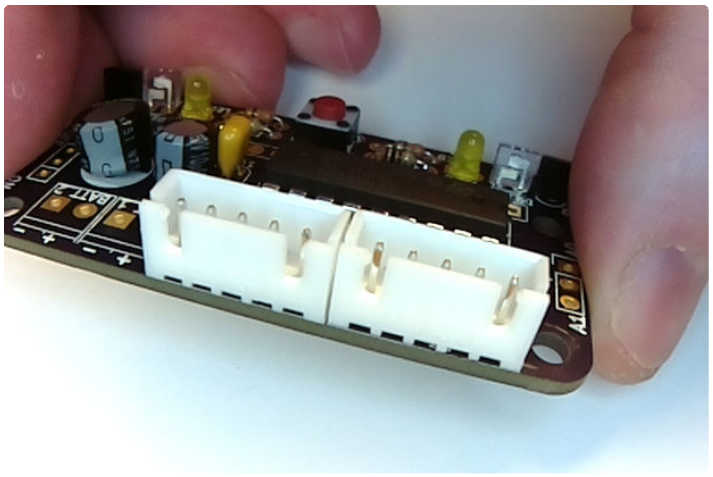
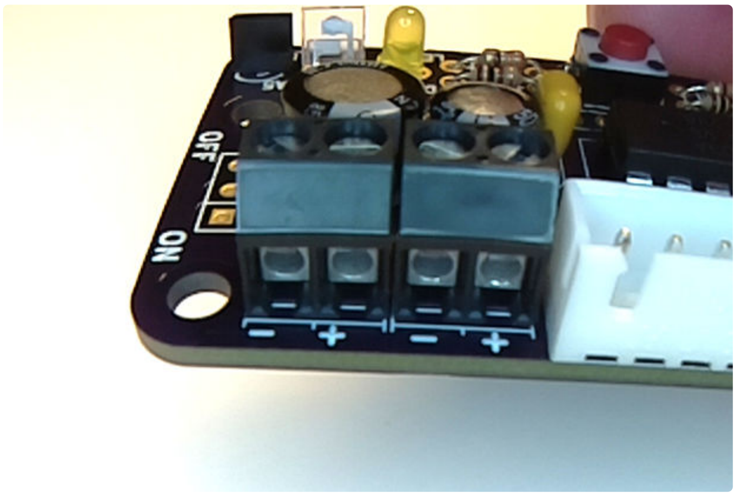




Step 7: EE: Headers & Terminals

The two stepper motors are connected using the white JST headers. They have two slot on one side that face outward.

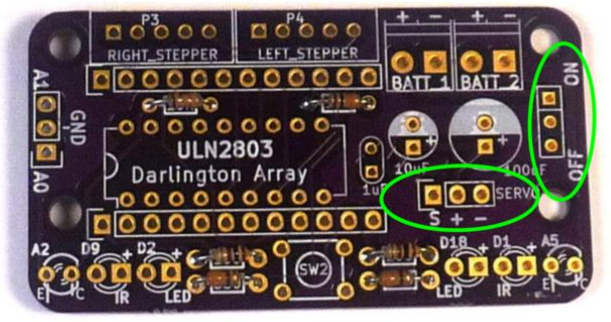
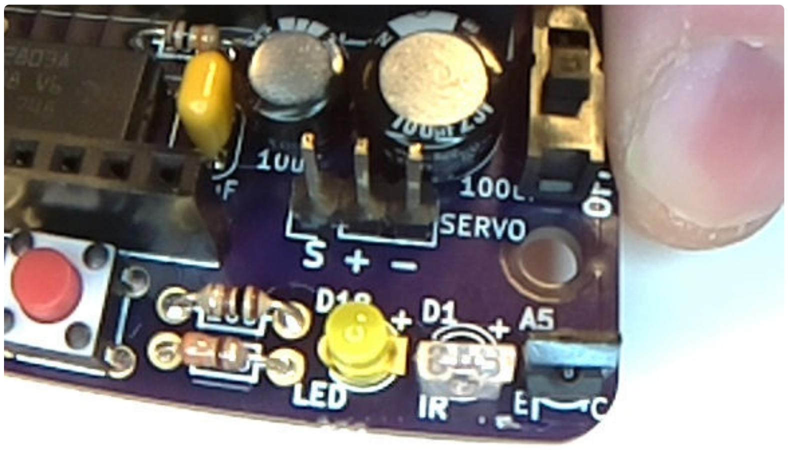
Power from the two battery holders are connected via the two black terminal headers. The openings face outward as well.

Step 8: Servo Header and Power Switch

We need a three pin male header to connect the servo. Insert it with the long legs up (either direction) and solder it in place.

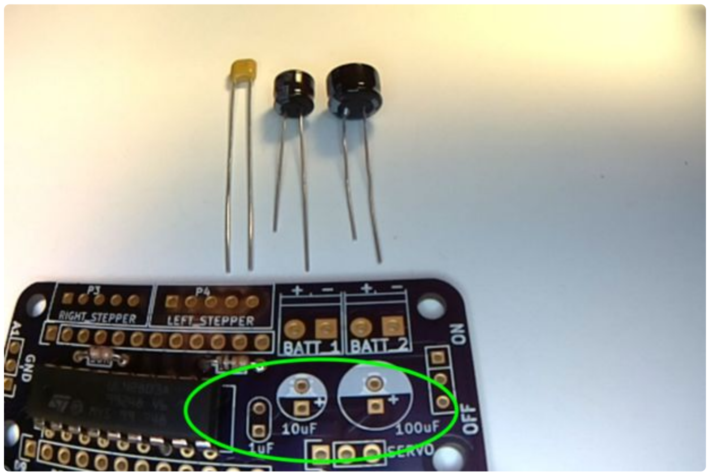
The power switch can installed in either direction.

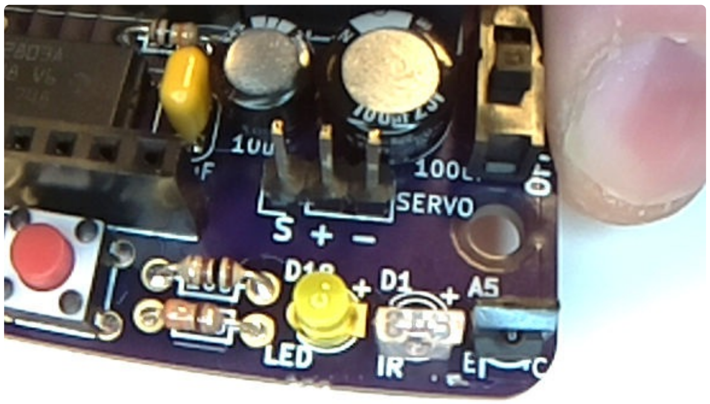



Step 9: EE: Capacitors

There are three capacitors of various capacities to help smooth power transients that occur while the motor signals switch on and off. The two larger black capacitors (1 uF and 10 uF) are electrolytic and have polarity. Like the LEDs, the long leg is positive, and there is a white strip on the body indicating negative.

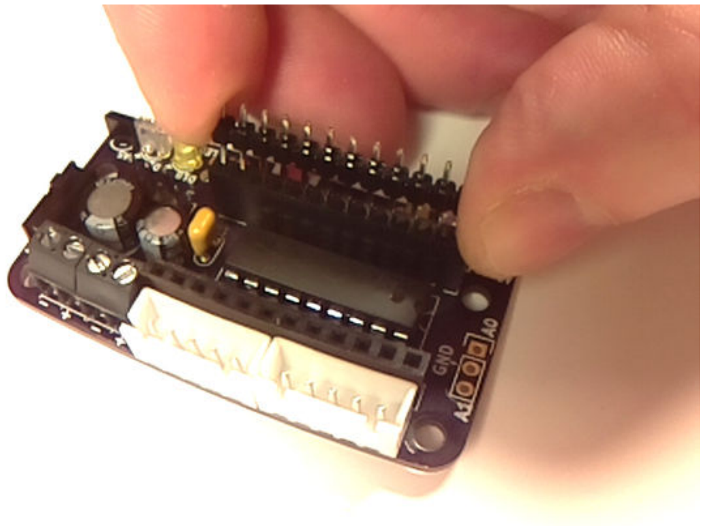
The smaller capacitor (0.1 uF) is ceramic and has no polarity.

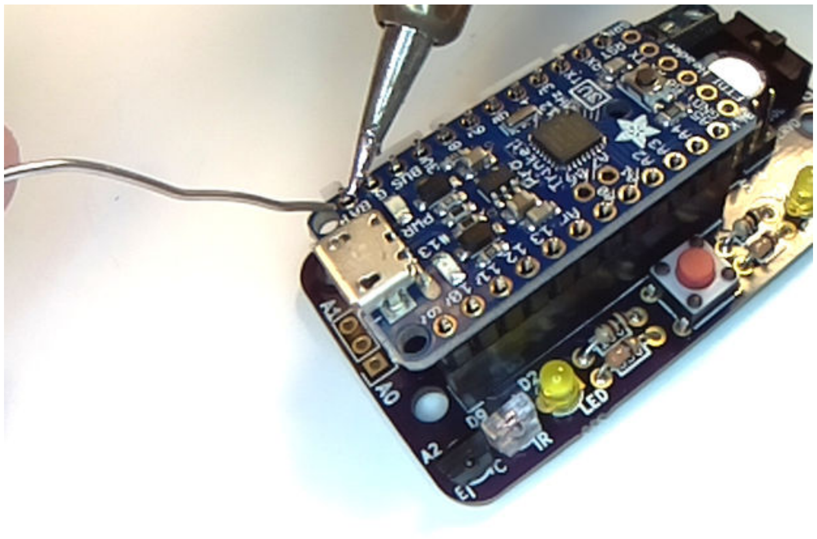




Step 10: EE: Brains

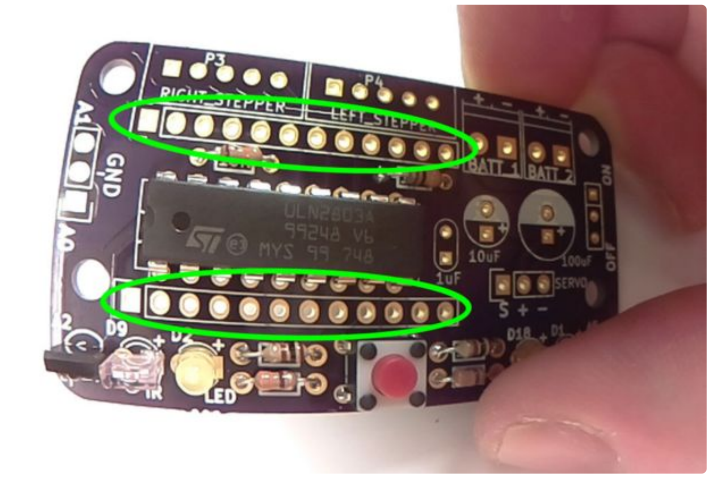
The Adafruit Trinket connects to the motor board via two 12-pin female headers. Solder them in place with the opening facing upward.

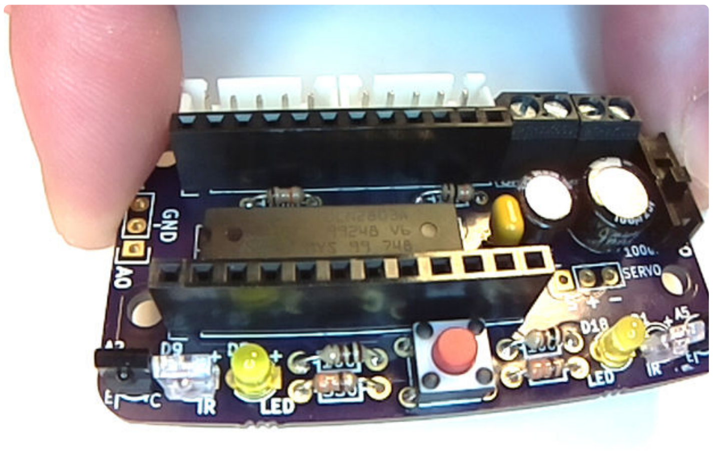


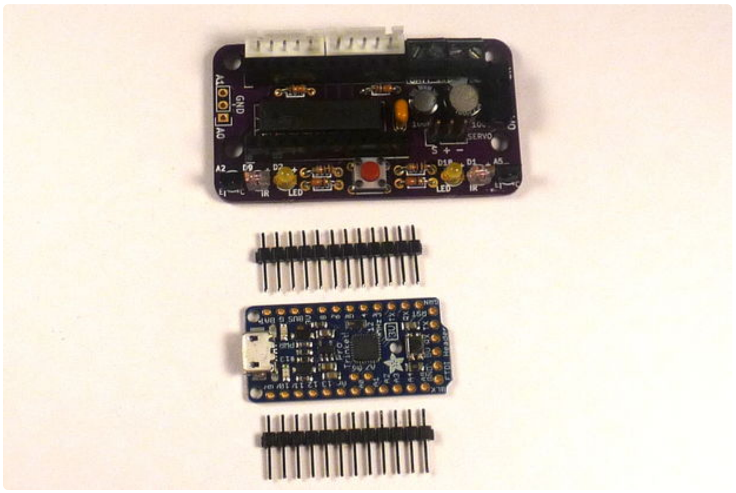


The Trinket comes with pin headers which are longer than needed:

- Cut both of them down to 12 positions.
- Place them in the female headers with the short side facing up.
- Place the Trinket on top of the headers with the USB facing outward, and solder the pin headers into the pads.







Step 11: Mechanical Engineering (ME): Parts

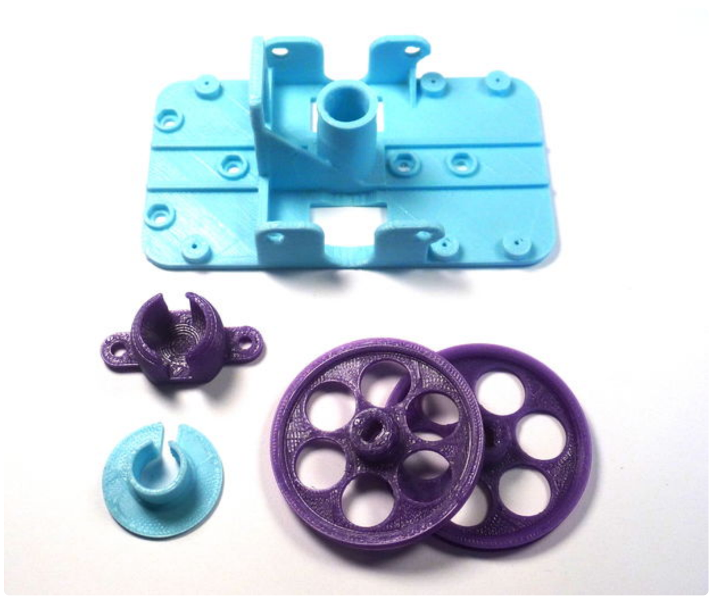
Time to switch gears from Electrical Engineering to Mechanical Engineering.

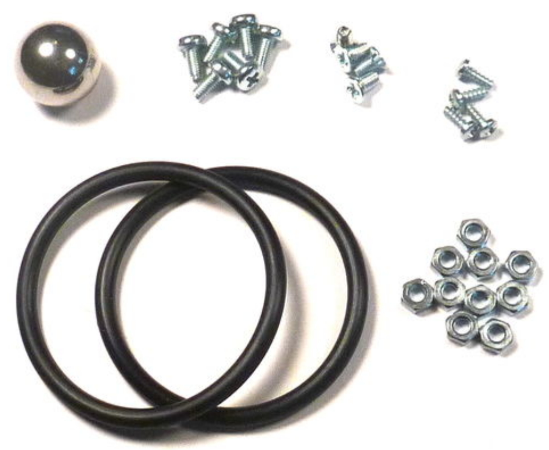
The required hardware is on the bill of materials (BOM): https://github.com/aspro648/OSTR/blob/master/BOM.md.

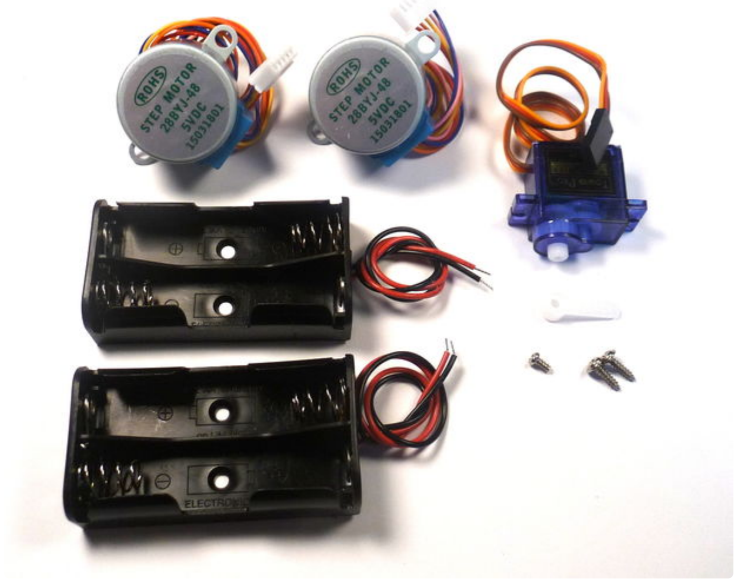
A current copy of printable STLs are located on Thingaverse. You will need:

- 1 Chassis
- 2 Wheels
- 1 Caster
- 1 Pen Collar
- 1 Nameplate (optional).

Again, a kit for this project is available on Tindie.com if you don't have access to 3D printing.






ROHS
STEP MOTOR
28BYJ-48
5VDC
15031801
ROHS
STEP MOTOR
28BYJ-48
5VDC
15031801

Step 12: ME: Chassis Prep

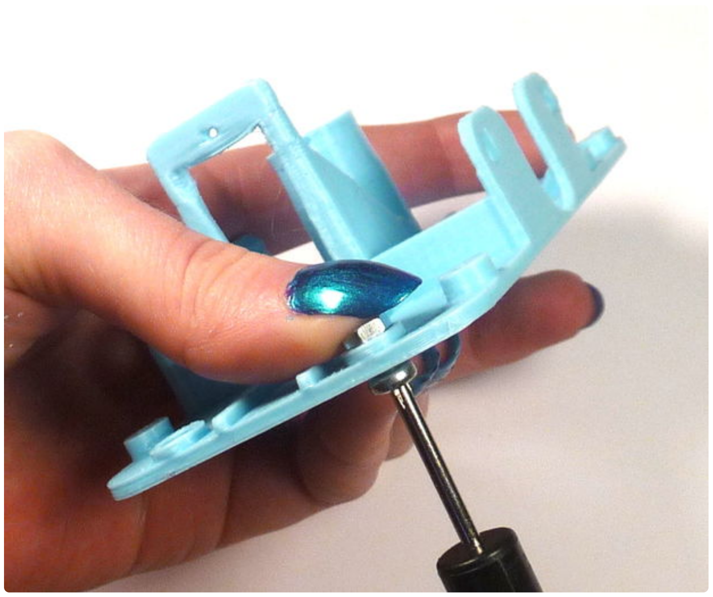
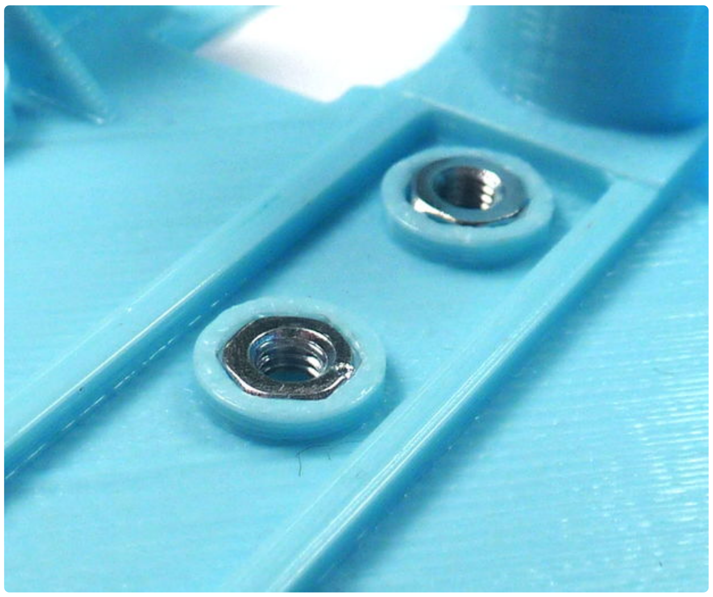
A "chassis" is the fame of a motor vehicle or other wheeled conveyance, and in this case the robot's chassis forms the platform for the electronics, motors, and servos.

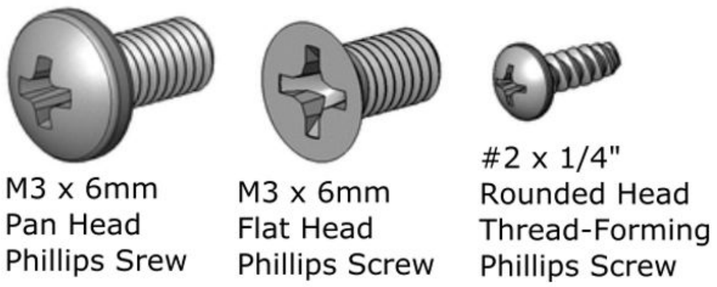
Time to get familiar with your hardware. We will be using three different types of Phillips screws you will have to be able to identify:

- Pan Head Screw (M3 x 6 mm) -> Stepper motors to Chassis
- Flat Head Screw (M3 x 6 mm) -> Battery Holders to Chassis
- Round Head Thread-Forming Screw (#2 x 1/4") -> PCB to Chassis

In order to attach parts, we are going to embed M3 nuts into the chassis.

- Use an M3 round-head screw and tighten it to pull the nut into the pocket so it is seated flush.
- Remove the screw.

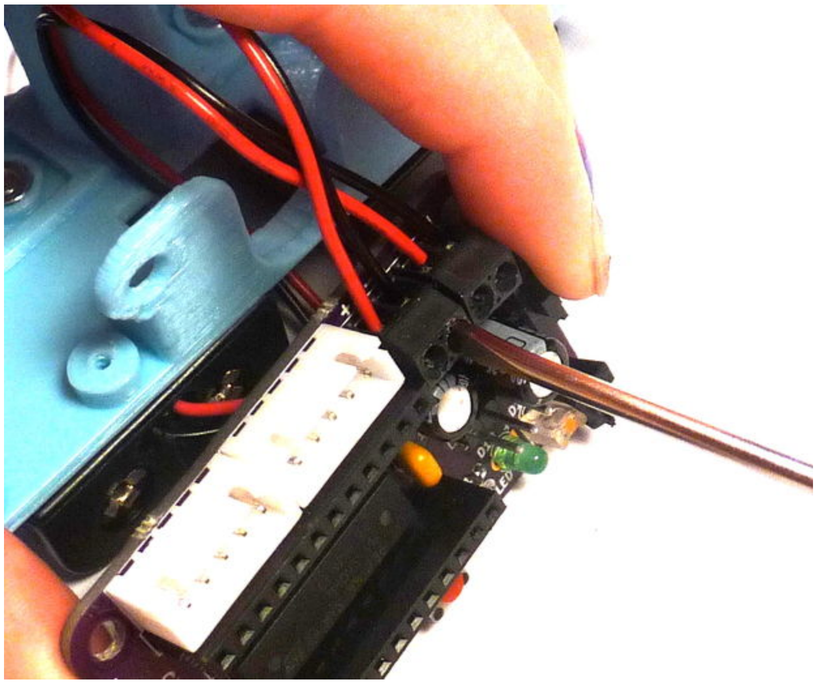



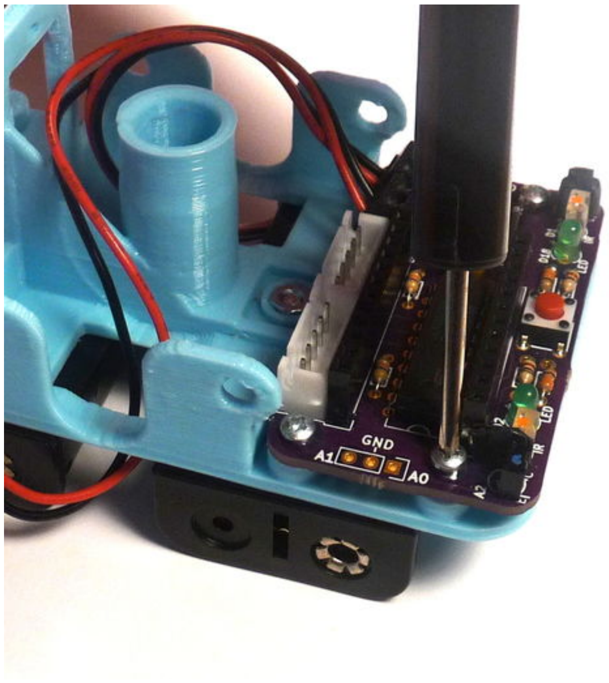


Step 13: ME: Batteries

Integrating the electrical components like motors and batteries with the chassis will fall into the ME's domain.

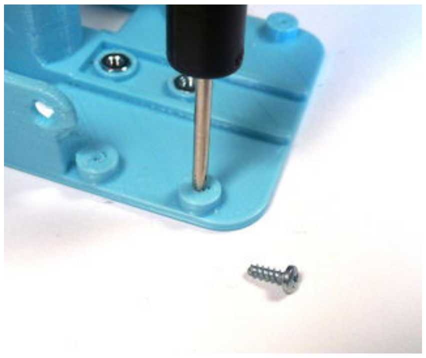
- Attach the battery holders using two M3 flat-head screws and thread the wires through the openings under the motor mounts.

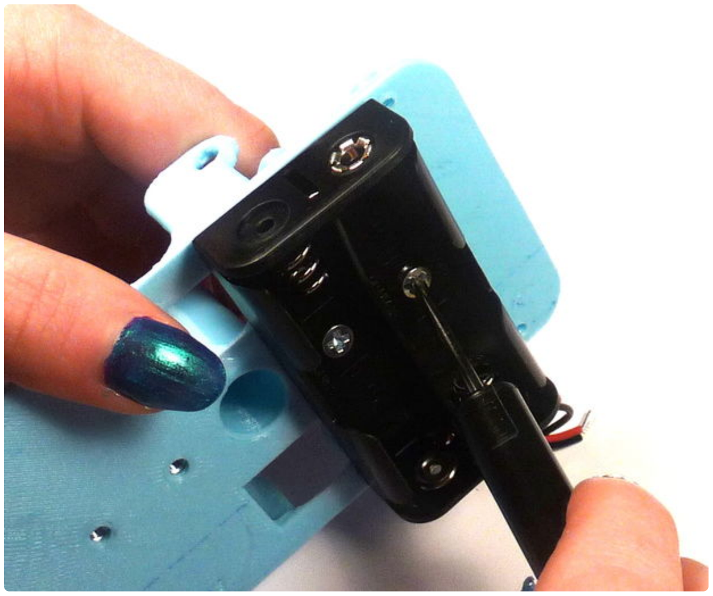
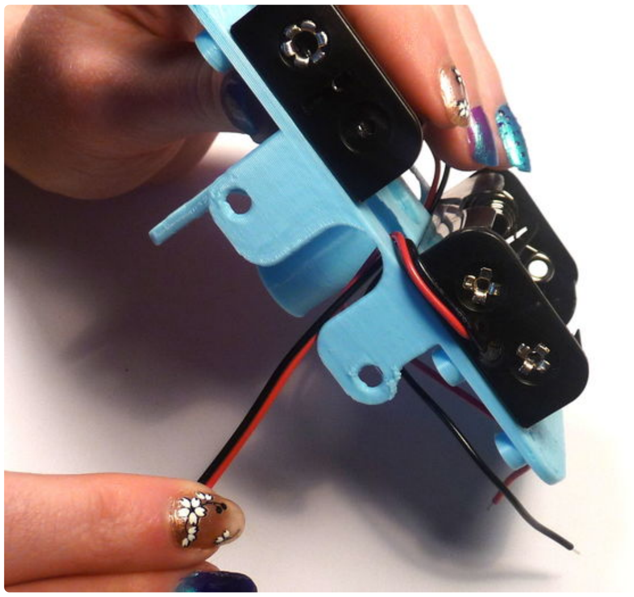




- Put on your "EE" hat and attach the positive and negative wires to the terminals marked "+" and "-", one set of batteries for each terminal.

- Attach the circuit board to the chassis using #2x1/4 thread-forming screws.
    - If the screws wont start, remove the PCB and use the tip of the screw driver to open them up a bit.

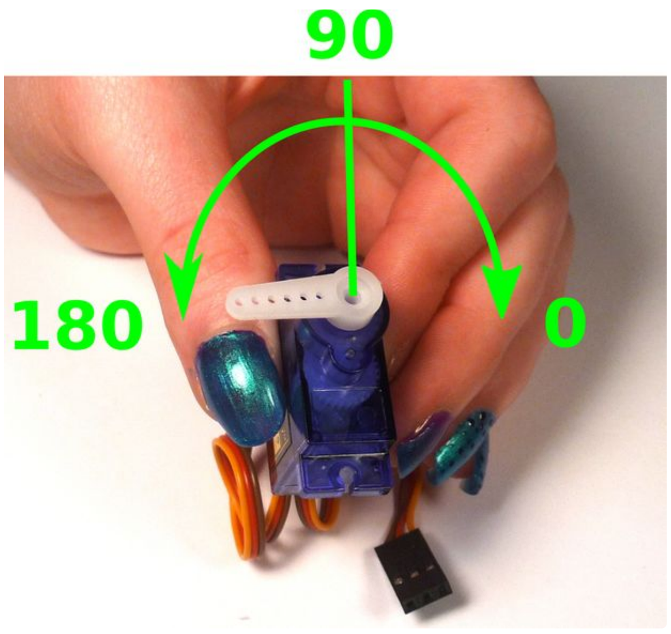


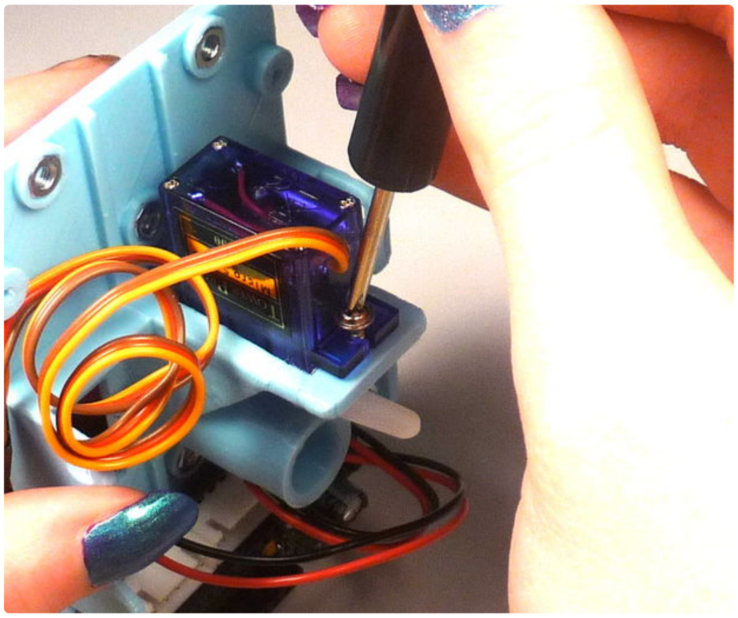



Step 14: ME: Servo

The servo is used to raise and lower the pen for drawing.

- Place the arm on the hub and gently rotate the stepper counter-clockwise looking down on it until it reaches the stop.
- Remove the arm and position it facing left (this will be the down position).
- Insert the small thread-forming screw and tighten.
- Insert the servo in the mount with the hub end upward and attach using two larger thread-forming screws.





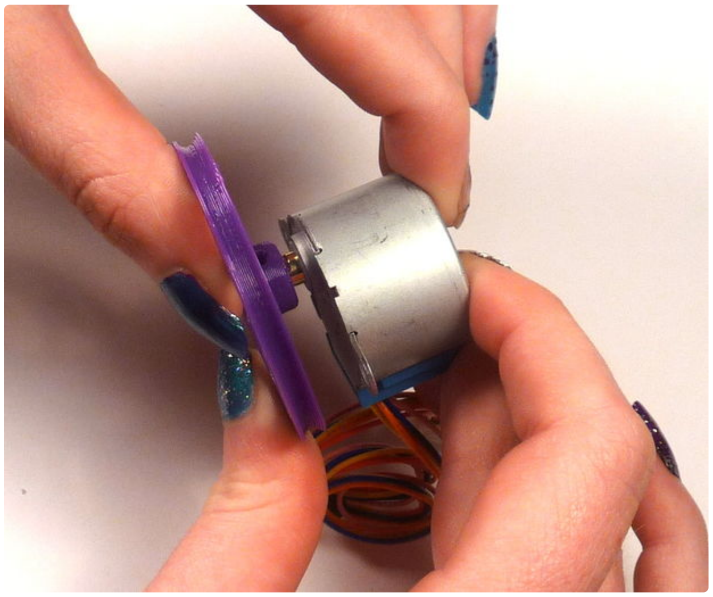
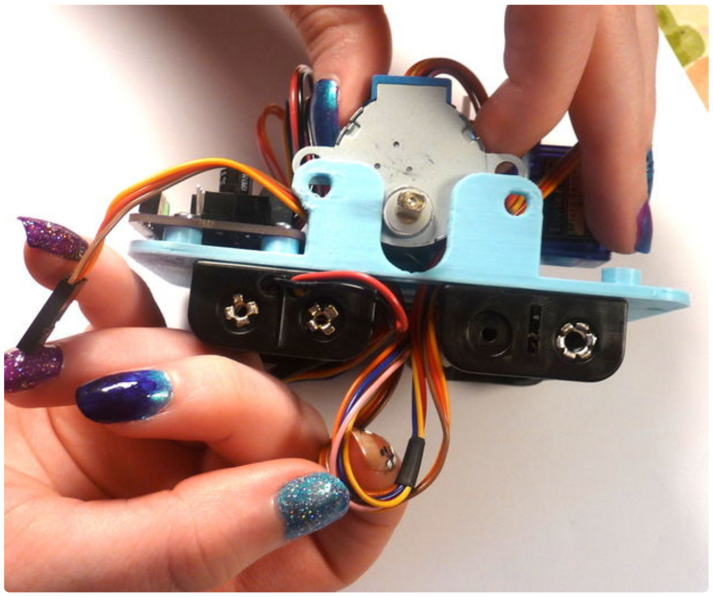
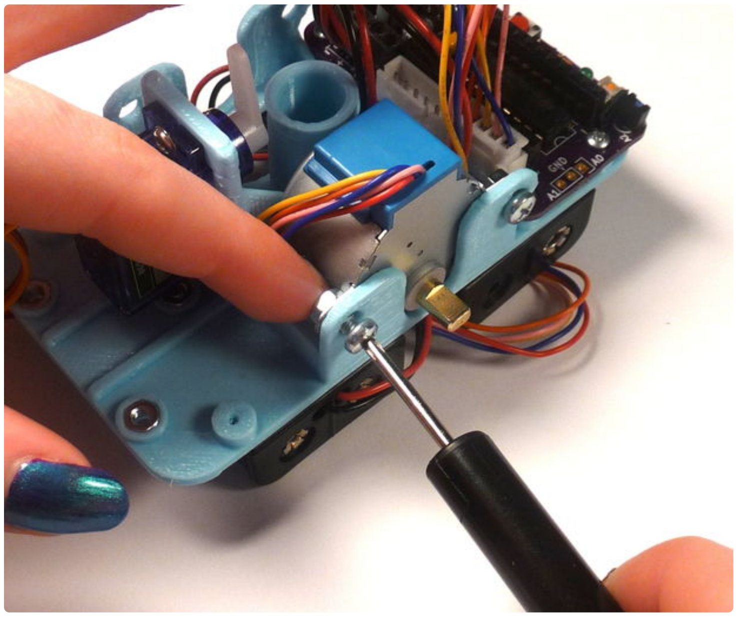
Step 15: ME: Stepper Motors

Before we mount the stepper motors, check the fit of the wheels on the shaft. These should slid on fairly tightly if your printer is calibrated correctly.

- If they are too tight, use a jewelers file to open up clearance. You can also heat the hub with a hair drier, but do so sparingly to avoid warping the rim.
- If they are too loose, you can use an M3 round-head screw to hold the wheel to the shaft (after the motor is mounted).

We are going to pass the wires of the stepper (and servo) under the body to help with wire management.

- Start with the left side and thread the wires for the stepper (and servo) through the opening with wires coming from the stepper following down the back side.
- Insert the motor into the mount and attach with two M3 round-head screws and nuts.
- Gently pull the wires through.

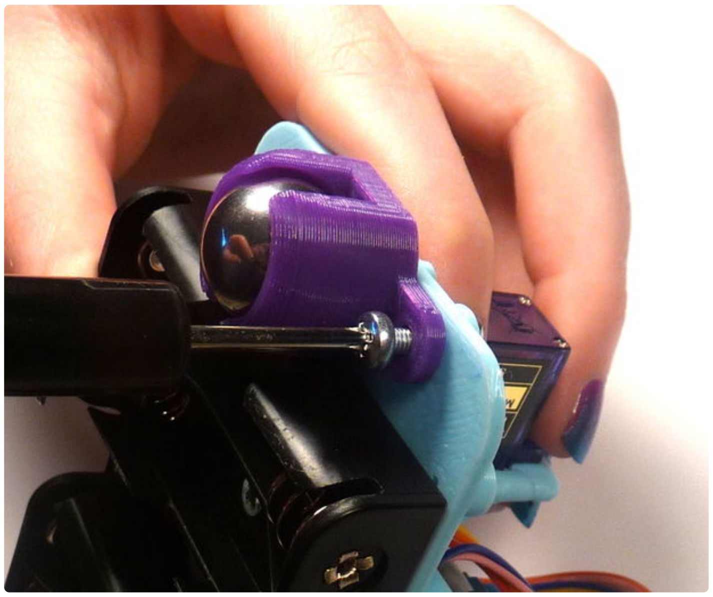




Step 16: ME: Caster

The caster allows the robot to pivot with low friction.

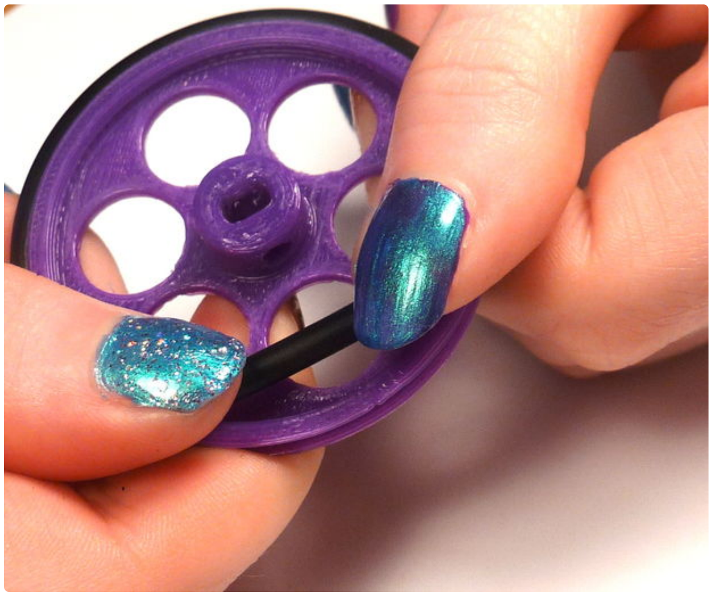
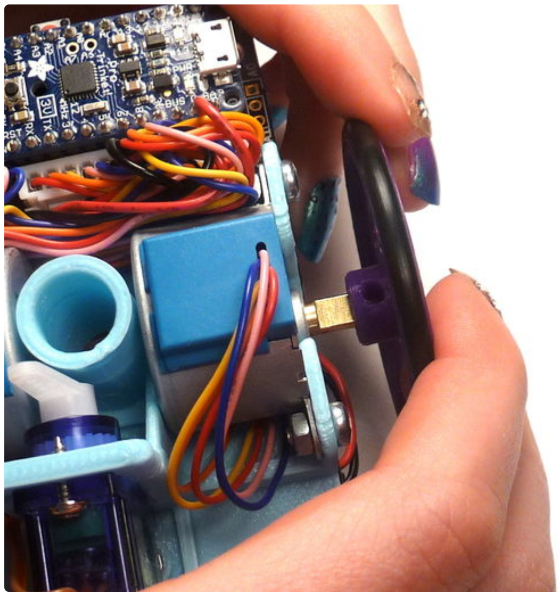
- Gently press the bearing into the holder.
    - If it is too tight, use a hair dryer to soften the plastic.

- Attach the caster to the chassis using M3 round-head screws.



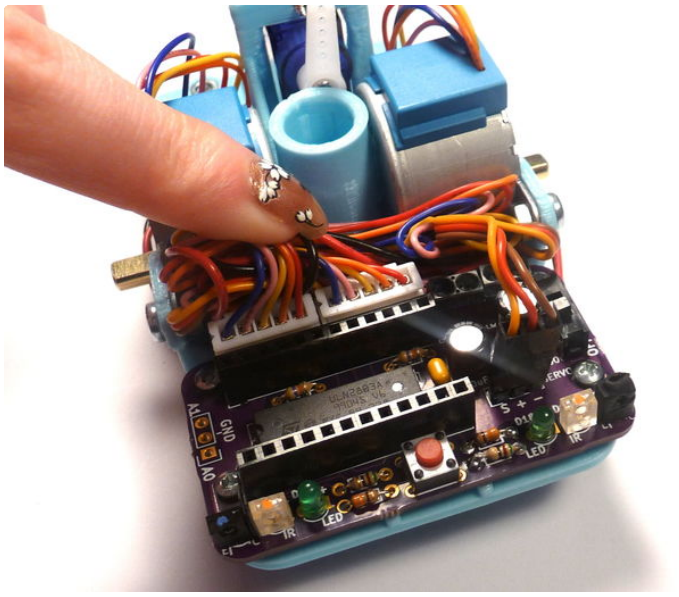
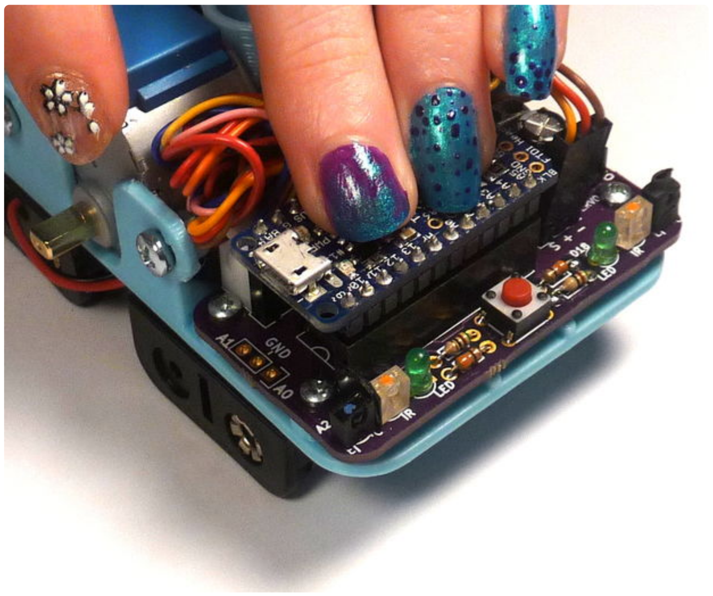
Step 17: ME: Wheels

- Slip the o-ring into the rim groove and stretch it to snap in place.
- Gently press the wheel on to the stepper shaft as far as it will go.
- If the wheel is too loose on the shaft, thread a M3 round-head screw in the hub and tighten on the hub.
    - You will have to ensure the battery wires remain clear of the screw as the wheel rotates.

Step 18: ME: Power

- Attach the stepper connectors to the terminal headers on the PCB.
- Fold and press the wires into the cavities between the headers and the steppers to keep them clear of the servo arm during operation.
- Insert the Adafruit Trinket Pro into the headers with the USB hub facing the edge and press into place.
- Ensure the power switch is set to off.
- Insert the batteries.

Step 19: Firmware (FW): Testing and Blinking

Time to add some code to our robot. We are using an Adafruit Trinket Pro (3V 12Mhz) which has the same microcontroller chip as the Arduino so we can use it's programming environment. In order to have Trinket board support, you need to add it to Arduino, or just download Adafruit's version. Follow the instructions here:

- https://learn.adafruit.com/adafruit-arduino-ide-setup/arduino-1-dot-6-x-ide

If you are using Windows, you will also need to install drivers from:

- https://learn.adafruit.com/adafruit-arduino-ide-setup/windows-driver-installation

Launch your Arduino IDE and set the following:

- [Tools] -> [Programmer] -> "USBtinyISP"
- [Tools] -> [Board] -> "Pro Trinket 3V/12 Mhz (USB)"

Open my favorite test sketch:

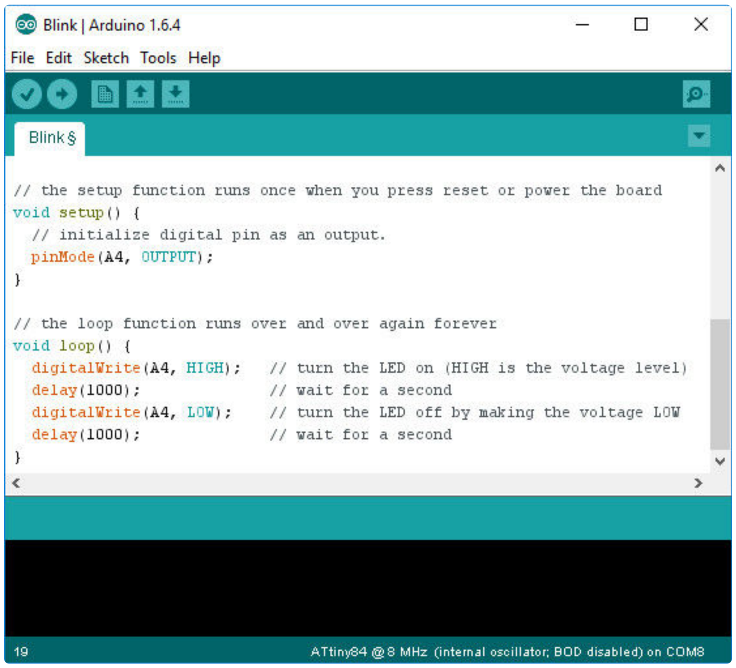
- [File] -> [Examples] -> [01.Basic] -> "Blink".
- Change all pin designations "13" to "A4" (the robot's left LED).
- Use the check mark icon to verify the sketch compiles correctly.
- Attach the USB Micro cable to the Trinket USB port.
- Press the reset button on the Trinket board.
- Immediately press the upload icon on Arduino IDE.

Is your robot's LED blinking? Yes = YEAH! No = DONT PANIC!

- Double check the Arduino IDE settings for Programmer and Board above.
- Try the reset and upload sequence again. (The bootloader only looks for new code for several seconds before moving on to what is currently loaded.)
- Try ideas here: https://learn.adafruit.com/adafruit-arduino-ide-setup/troubleshooting
- Ask for help, either in the comments here or on the Adafruit forums.

Investigations:

- The LED is blinking once every two seconds (one second on, one second off).
- What controls that in the blink sketch?
- Can you make it blink at a different rate?
- How small of a delay can you use and still be able to see the blinking?
- If it appears steady at 5 ms, what happens if you carefully shake it back and forth?



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.
  pinMode(A4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(A4, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(A4, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

```
19                ATtiny84 @ 8 MHz (internal oscillator; BOD disabled) on COM8
```

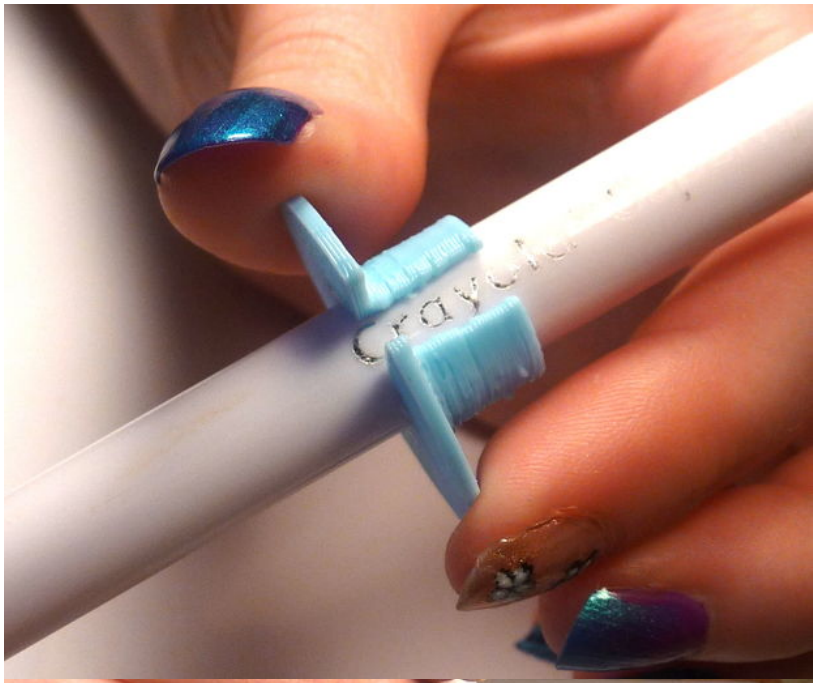
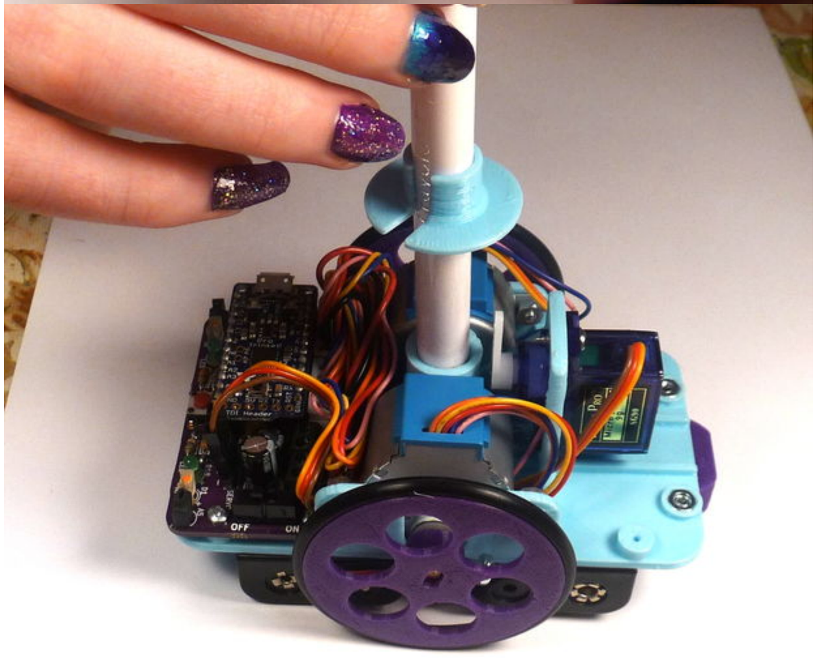
Step 20: FW: Calibration

Because of variations in assembly and alignment, the robot must be calibrated so that it can move precised distances and angles.

- Measure the wheel diameter from the outer edges of the rubber o-ring.
- Measure the wheelbase from the center of the o-rings on the bottom of the robot (where it will contact the floor).

Calibration firmware:

- Download from https://github.com/aspro648/OSTR/tree/master/firmware
- Enter your measured parameters.
- Upload the sketch as in the previous step.

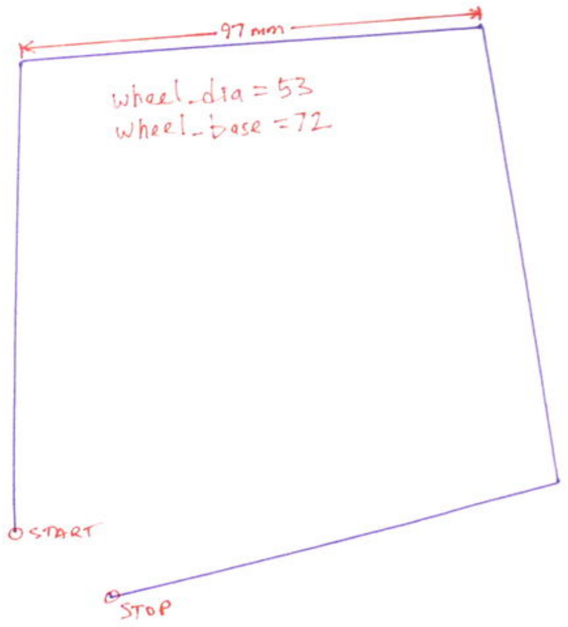



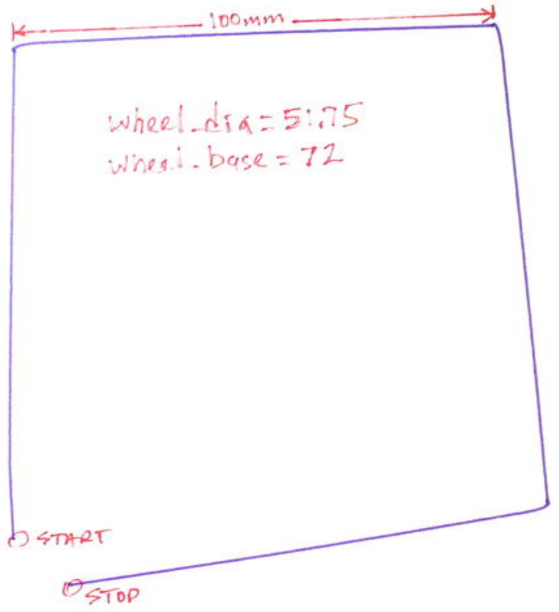
Prepare the pen:

- Remove the cap and slide the pen collar from the tip side.
- Align collar with “C” in “Crayola”.
- Insert pen in holder with servo arm straight up.
- Ensure pen does not touch paper in this position.
- If the pen binds in the shaft, us a file to remove any roughness and increase the bore diameter.

Draw a square:

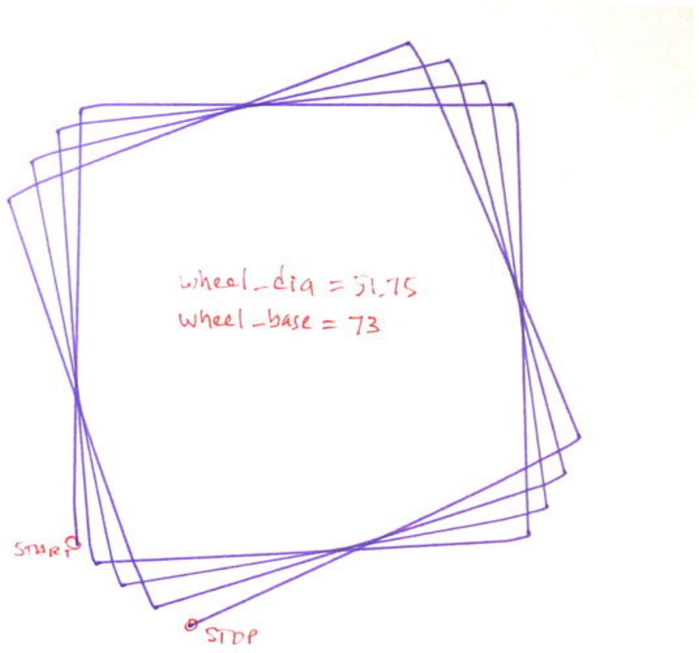
- Slide the power switch to "On".
- Wait several seconds for the bootloader to begin.
- After the robot completes it's first square, remove the pen and turn the robot off.

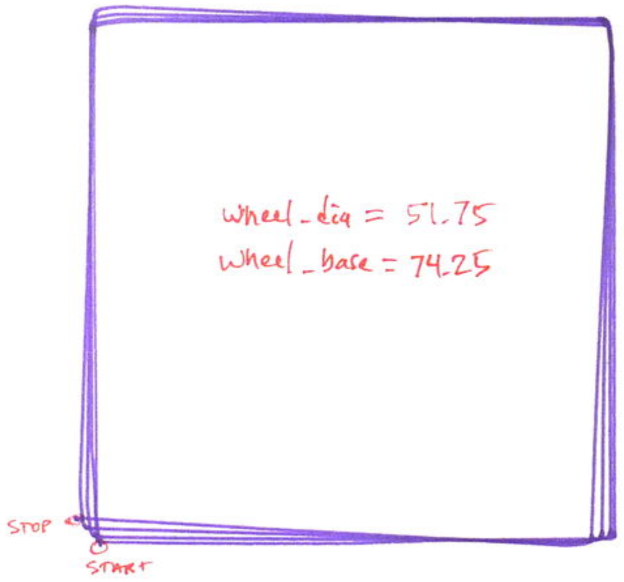




Adjust ***wheel_dia*** parameter first. Measure the length of side of the square. It should be 100 mm:

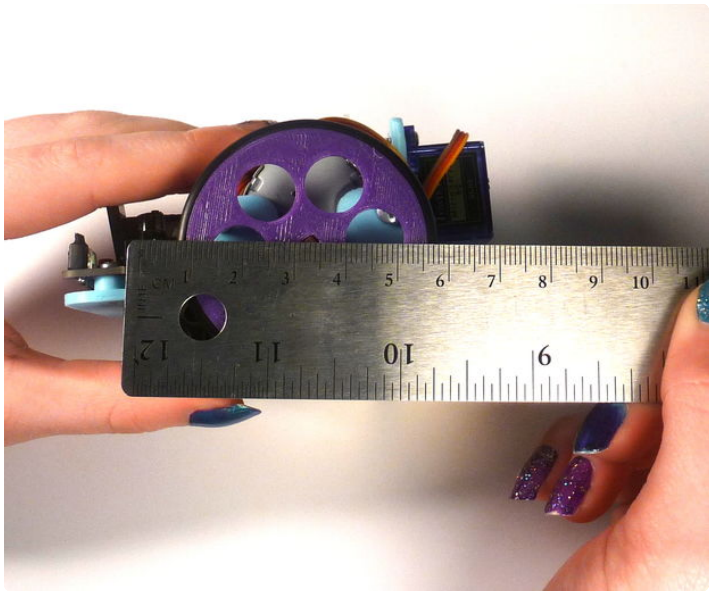
- If the measured distance is too long, increase *wheel_dia*.
- If the measured distance is too short, decrease *wheel_dia*.



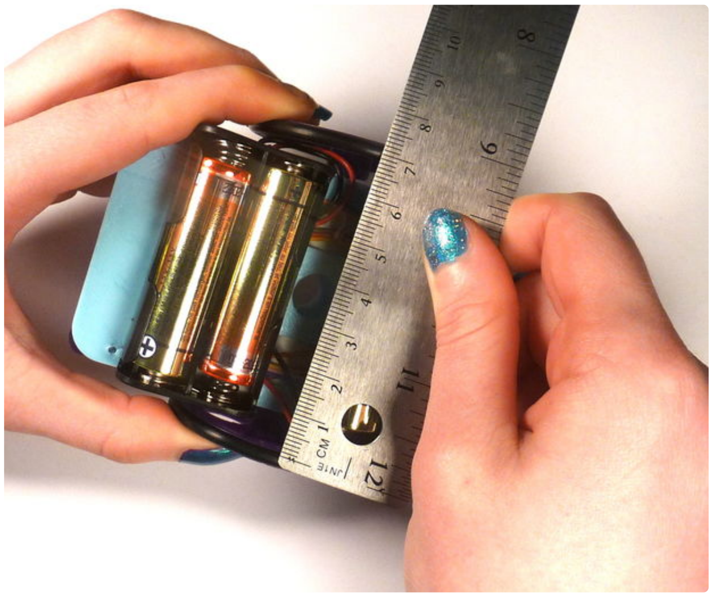


After you have the distance calibration, adjust the ***wheel_base*** parameter which affects the angle of the turn. Place the robot on a fresh sheet of paper, turn it on and let it draw all four squares:

- If the robot is turning too sharply (box is rotating clockwise), decrease *wheel_base* value.
- If robot is not turning sharply enough (box is rotating counter-clockwise), increase *wheel_base* value.
- Because of rounding errors in the stepping code and slop in the gears of the inexpensive steppers, you will never get it perfect, so don't spend too much effort on it.

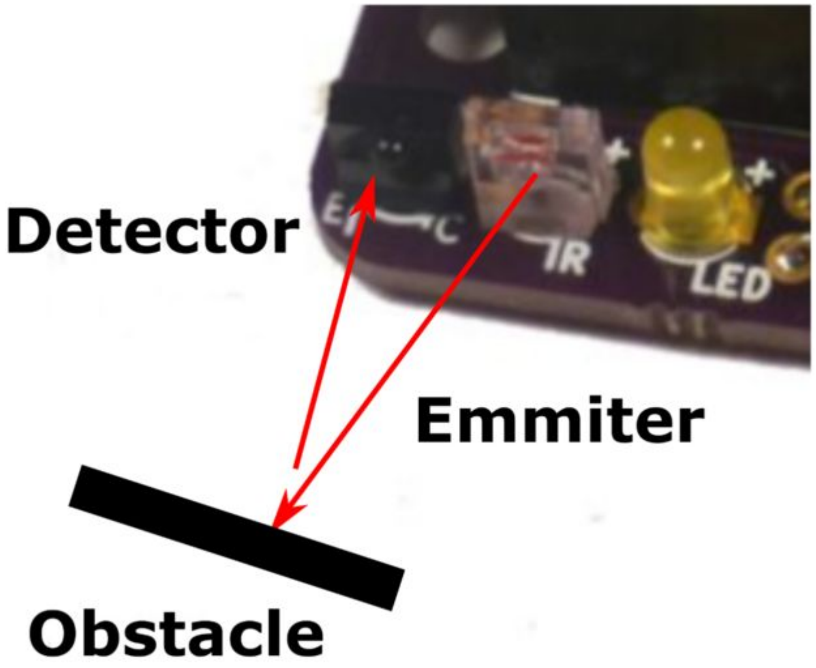


1
2
3
4
5
6
7
8
9
10
12
11
10
9



1
2
3
4
5
6
7
8
9
10
12
11
9
8

Step 21: Vision Check



Now that you have a calibrated turtle, let's look at how it's eyes work. There are lots ways for robots to detect their environment:

- Bump sensors (limit switches).
- IR emitter/detector pair
- Sonar
- Laser range finding
- Beacon detection
- GPS

We are using IR because it is easy and inexpensive. The downside is that detection distance is short (several inches), varies by surface reflectance, and is subject to interference from other IR sources like the sun.

- Download and install the OSTR_eyes sketch from Github.
- Verify the left and right LED light when you hold your hand in front of the detector.

If you are not getting the desired response:

- Verify the "blink" sketch works or that the LEDs blink at the beging of the OSTR_eyes sketch.
- Use a phone or camera to observe the IR emitters are working.

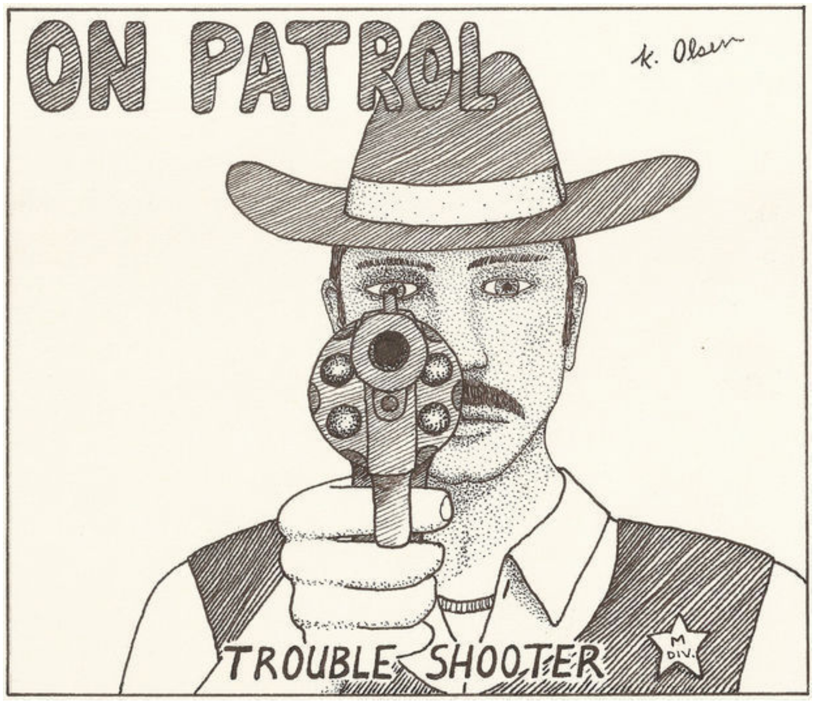
If the indicators eyes remain on with no obstacle in front of them:

- It is possible that IR light is bleeding directly over to the detector. Place a piece of light-blocking material like dark plastic between them. If the light goes out, you have solved them problem. Trim the piece and use some hot glue to hold them in place.

Step 22: Trouble Shooting

This is a pretty complex project, and will likely give you fits at some point. I like to tell workshop participants that they need to have:

- **Patience** - Don't give up at your first failure.
- **Persistence** - You have to keep trying different things to solve problems.
- **A Positive Attitude** - You can do it!



If you are having **trouble uploading sketches** to the Trinket:

- Remove the Trinket from the Robot Driver Board.
- Double check the steps of "Firmware (FW): Testing and Blinking".
- Try ideas at https://learn.adafruit.com/adafruit-arduino-ide-se...
- Ask for help in the comments or Adafruit forums.

No power when on-switch turned on:

- Ensure you have fresh batteries.
- Ensure batteries installed correctly.
- Double-check battery connections to terminal blocks.
- Disconnect servo & steppers.
- Use a volt meter to verify voltage at terminal blocks.

Wheels do not move or stop moving during run for calibration step:

- Ensure you have fresh batteries installed correctly.
- Check white stepper connectors inserted fully.
- Ensure battery wires are not binding the wheels.

- Ensure steppers rotate by hand.

Robot turns the wrong direction:

- You have the left and right steppers plugged in to the opposite headers.

Pen doesn't leave a trace:

- Ensure pen is free in barrel and will drop below the wheel.
    - Use a file to remove roughness and open barrel diameter.

- Ensure pen holder is at the proper level hold the pen up.
- Ensure servo points straight up during *penup()* command. If not, adjust PENUP angle in firmware.

Pen Too Loose:

- Verify your 3D printing parameters.
- Add some tape to the pen to increase the barrel diameter.

Step 23: Now What? Curriculum!

It works and draws nice squares and can detect walls. Now the fun begins.

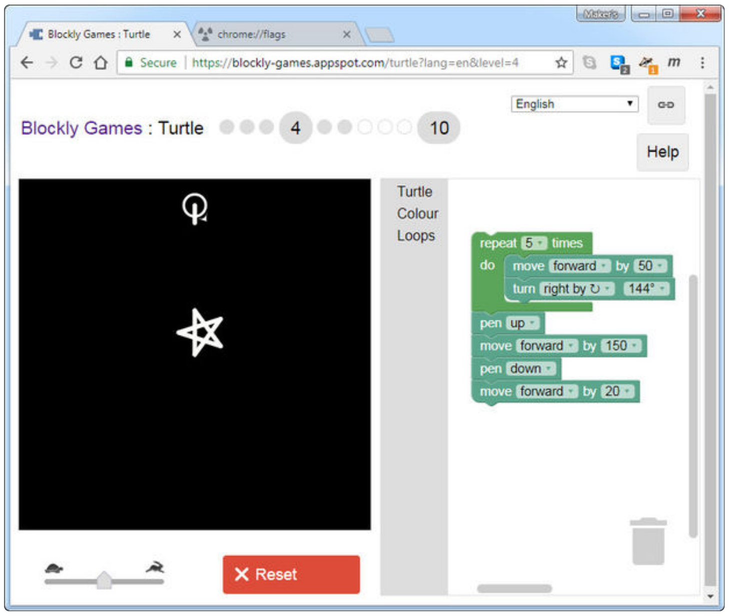
Here are a couple of resources for learning turtle graphics.

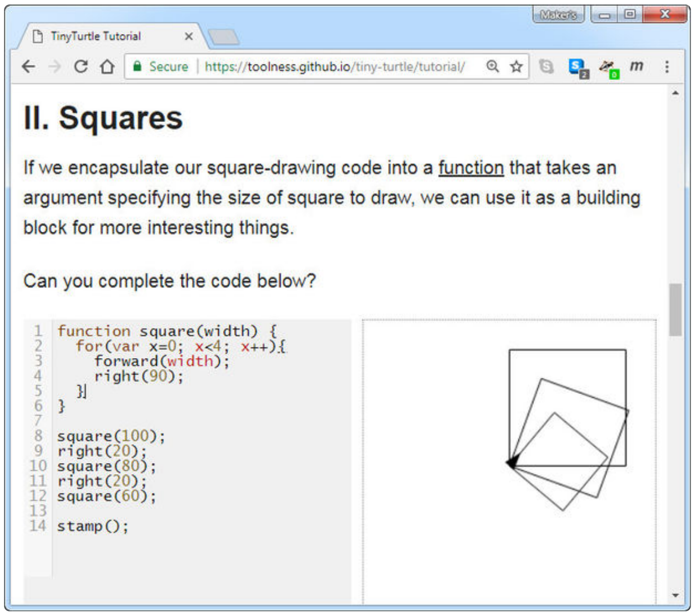
- https://blockly-games.appspot.com/ (block programming)
- TinyTurtle Tutorial (JavaScript)
- Code with Anna and Elsa from Hour of Code

I have also posted an Instructable about using the turtle robot these on-line resources with the Turtle Robot. In general, any Turtle JavaScript code can be pasted and run in the calibration sketch. You can test output online the a computer first and then uploaded it to your turtle to draw out in real life!

For students, here are a couple of project ideas:

- Program your robot to write your name!
- Design and 3D print a nameplate in TinkerCad from a template. It can be attached below your servo motor.
- Give your robot some personality with some hot glue and bling. (Just keep the wheels and eyes clear of obstructions).
- From the OSTR_eyes sketch, design and test an algorithm to navigate a room. What do you do when one eye detects something. Both eyes? Could you incorporate Arduino's random() function.
- Construct a maze on a large sheet of paper on the floor and program your robot to navigate through it.
- Construct a maze with walls and design an algorithm to navigate it automatically.
- The button in between the LEDs hasn't been put to use yet, and is connected to Arduino pin "A3". What could it be used for? Use it to turn an LED on and off to start with.
- If you didn't do the *Investigation* section of the "*Firmware (FW): Testing and Blinking*" step, go back and give it a try.





II. Squares

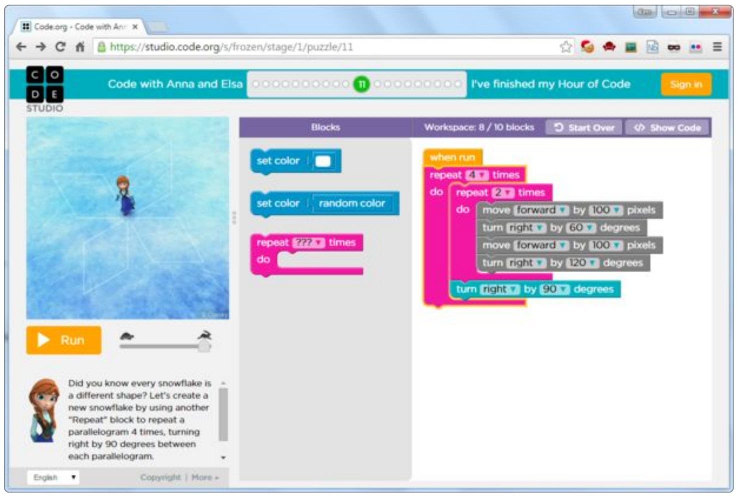
If we encapsulate our square-drawing code into a function that takes an argument specifying the size of square to draw, we can use it as a building block for more interesting things.

Can you complete the code below?

```
function square(width) {
  for(var x=0; x<4; x++){
    forward(width);
    right(90);
  }
}

square(100);
right(20);
square(80);
right(20);
square(60);

stamp();
```



```
when run
repeat 4 times
do  repeat 2 times
    do  move forward by 100 pixels
        turn right by 60 degrees
        move forward by 100 pixels
        turn right by 120 degrees
    turn right by 90 degrees
```

Did you know every snowflake is a different shape? Let's create a new snowflake by using another "Repeat" block to repeat a parallelogram 4 times, turning right by 90 degrees between each parallelogram.

Step 24: Workshop Ideas

Join the Turtle Robots forum to share your robot and teaching ideas!

Resources are always limited when doing workshops, so I've been breaking students into three groups and rotating them through three areas, each with a knowledgeable volunteer.

- Electrical: Soldering the PCB.
    - Soldering irons, one for every two students so they can work in pairs.
    - Flush cuts
    - Safety glasses

- Mechanical: Chassis assembly with steppers and servos.
    - screwdrivers
    - plyers
    - hot glue

- Firmware: Turtle tutorials, LED blinking, uploading firmware, calibration.
    - Laptops/computers, at least one for every two students so they can work in pairs.
    - Rulers for calibration
    - Pre-built robots for testing code if they haven't assembled their robot yet.

Safety with the electrical portion is important and should be discussed before starting. Safety glasses should be worn. I have a squirt bottle of water at the table for cooling burns (and wetting sponges). Here are a couple of the soldering videos I like to show:

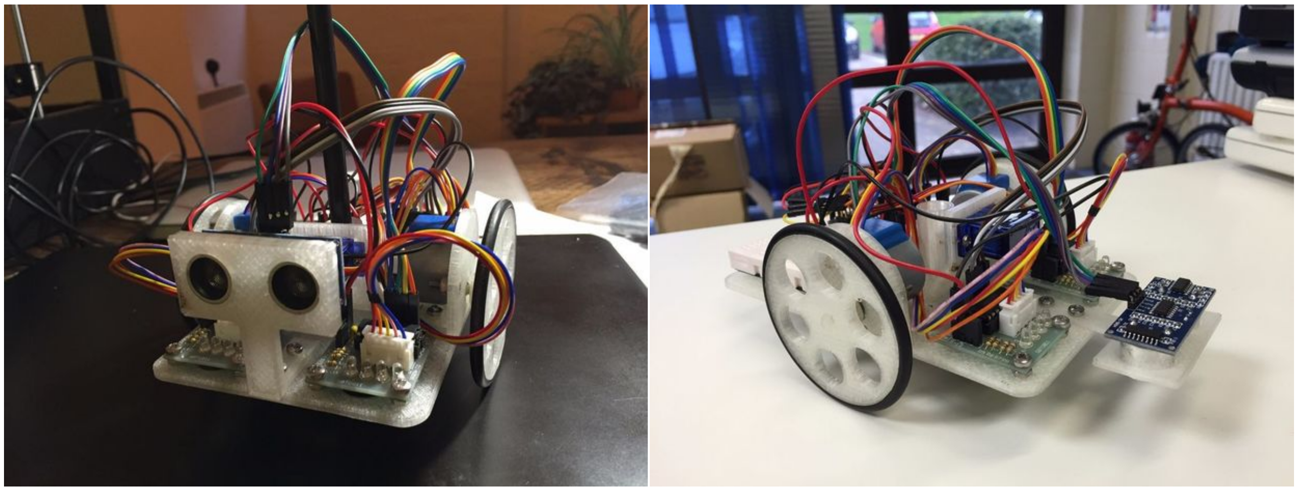
- https://youtu.be/P5L4Gl6Q4Xo
- https://youtu.be/QKbJxytERvg

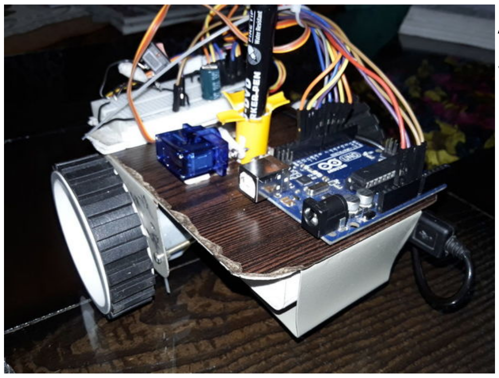
If time allows, give them a challenge to accomplish like a maze (see the "What Now? Curriculum!" step.)

Step 25: Turtles in the Wild

I love seeing these robots out in the wild, especially when they have been modified in ways I would have never have thought. Here are a couple of notable examples, with hopefully more to come:

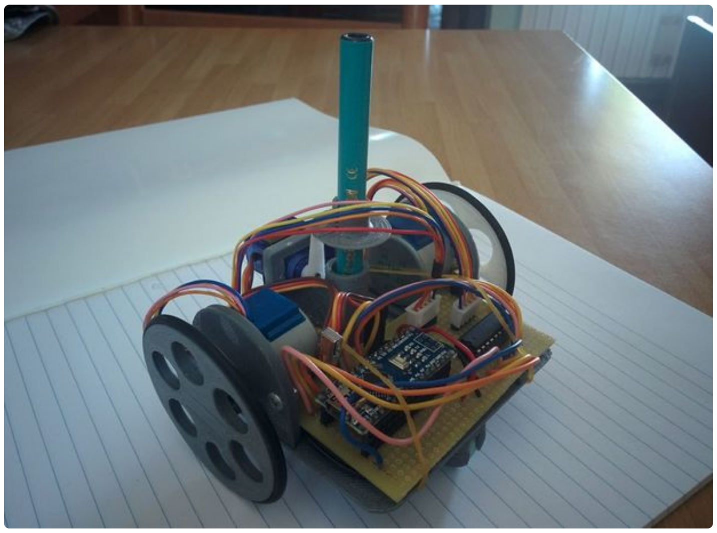
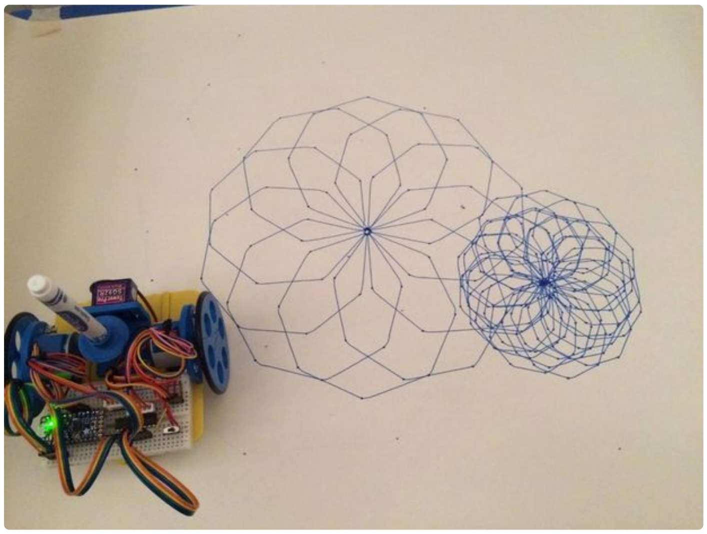
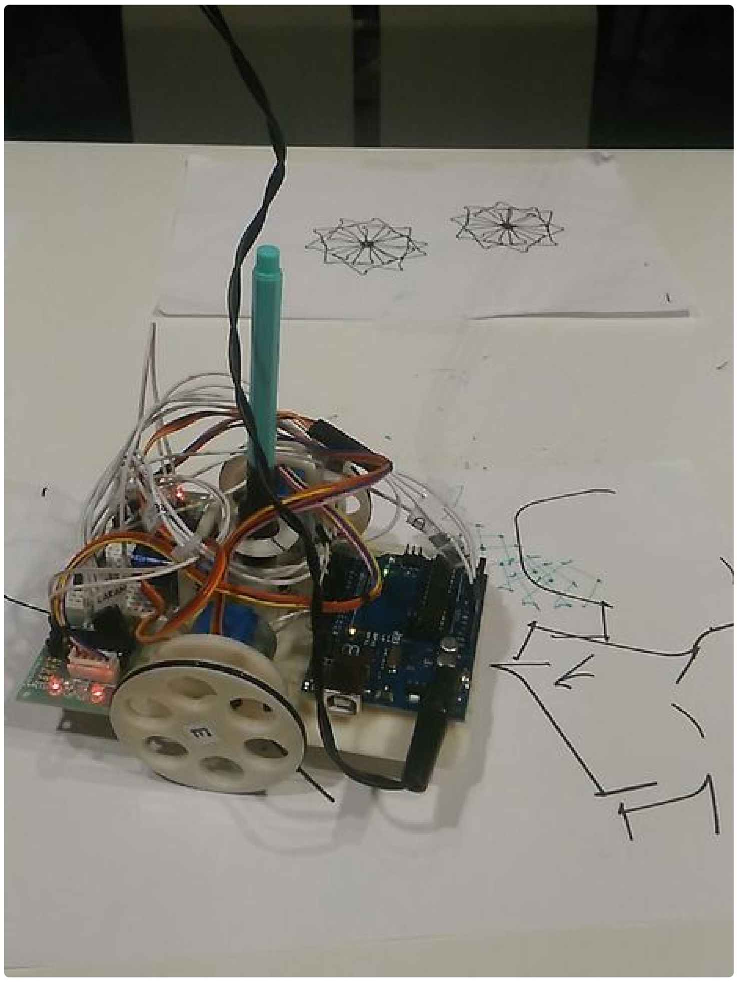
Russ Hughes does workshops and has created a custom circuit board, uses Tiny Basic, and has implemented the Hersey Vector Font.


Instructable user seancuttlefish added both sonar and line following



An in the "where there is a will, there is a way" category, Samarsis created a version with no 3D printing!

If you make a Turtle bot inspired from my examples, use the "I Made It!" button below, upload a picture, and make my day!

Just a heads up, the notes above about the resistor values are incorrect. The 10K is Brown-Black-**Orange**-Gold, and the 100 ohm is Brown-Black-**Brown**-Gold. We should have read the guidelines first to confirm. Unfortunately we soldered a couple in before we realized our mistake.

Ouch. That was a mistake. Thanks for catching and reporting it and sorry for the extra effort it caused!

I've updated the step.

Would like to purchase the kit but would also like to see a video of it working is there one?

A video is worth a million words . . .

https://youtu.be/ExhSeAjVGYc

Wow, This is the first drawing bot that I have seen that can actually write decently.

Is there a video of it working?

I like the way you divide into 3 engineering sections in order to build a robot, very well done!

How Nostalgic!

I always wanted one as a kid

Awesome very neat project :)

wow! that is so cool!