Anastasia build guide

Romilly Cocking

Monday 30 March - 04:02:33 PM

Introduction

Anastasia is an easy-to-build robot using off-the-shelf components.

Anastasia's motors are controlled by a micro:bit, and the driver uses a second micro:bit to tell Anastasia what to do.

The two micro:bits communicate via their built-in radio.

Building your own Anastasia

I built Anastasia from parts I had to hand. They are all readily available, apart from the Dyalog Phone Charger Battery for which there are plenty of alternatives.

Version one of Anastasia needed no soldering. If you don't want to solder your version, that will restrict your choice of motor. Some motors have leads pre-soldered, some do not.

Version 2 (the current version) has a little strip board power controller, with a USB socket, a switch, an LED, a resistor and a terminal block for output.

That's entirely optional, and I discuss alternatives in the build instructions. If you don't want to build it, and you manage to get motors with leads attached, the current version of Anastasia needs no soldering.

I'm working on adding an inexpensive LIDAR distance sensor to Anastasia. There are some technical challenges, but if I can overcome them I will add details to this repository. That will be a rather more advanced project, and it will need some soldering.

Here's the BOM (*Bill of Materials*), along with the sources I used.

When I started the project Anastasia was connected to a battery holder and powered by 4 AA rechargeable batteries.

Later I replaced the battery holder by a little strip board circuit with an on-off switch, a power LED, and a USB power socket.

I've listed these as alternatives in the BOM and in the assembly instructions, # Bill of Materials

Anastasia and Controller

Notes

1. *Generic* is my shorthand for *available from lots of places - any will do!*
2. The Caster and Brackets come with their own mounting nuts and bolts.
3. You'll need *two* micro:bits - one for Anastasia, one for the controller.
4. Alternative motors are widely available, but you'll need to check a few things.
    1. Make sure the motors are described as *TT motors*.
    2. Make sure the motors can run off 3-6 volts. Some appear to be 3v only; these might burn out.
    3. Make sure the motors will fit the Motor brackets. Some have a large raised pillar around their shaft.
    4. Make sure the size and shape of the motor shaft matches the wheels you get.
    5. If you don't want to do any soldering, make sure the motors have wires attached.

For Anastasia:

-	Component	Qty	Where I got it/them	Comment
1	Pololu board	1	Proto-PIC	
2	Adafruit caster 1.3"	1	The Pi Hut	
3	micro:bit	1	Kitronik	for Anastasia
4	TT motor	2	The Pi Hut	Slow but powerful
5	TT wheel	2	The Pi Hut	Fits the motor above
6	Kitronik Motor Controller	1	Kitronik	v 2.0
7	M3 spacers, bolts, nuts	4	Generic	
8	Adafruit motor brackets	2	The Pi Hut	

For programming the micro:bits:

Component	Qty	Where I got it/them	Comment
USB A to micro_B lead	1	Generic	Make sure it's a data transfer cable!

For the controller:

Component	Qty	Where I got it/them	Comment
AA battery	2	Generic	Rechargeable batteries are greener.
battery holder with switch	1	Kitronik	
micro:bit	1	Kitronik	for controller

To power Anastasia you'll need **one of the alternatives below**:

If you use a simple battery holder for Anastasia

Component	Qty	Where I got it/them	Comment
battery holder	1	Generic	4 cells. Make sure it has leads attached!
AA cells	4	Generic	Rechargeable batteries are greener.

If you use a Phone charger and stripboard with switch

Component	Qty	Where I got it/them	Comment
switch	1	BitsBox	
Stripboard (9x27)	1	BitsBox	If cut down, >= 6x20
Adafruit USB socket	1	Pimoroni	
Mobile charger	1	Generic	Use any charger that can provide 5V @ >=1A
Resistor 470R 1/8W 5%	1	Generic	1/4 W would also be OK
LED 3mm	1	Generic	
short USB lead	1	Generic	to connect battery + power unit

Adding headers (Optional)

I’m working on additional components that will let you do more with Anastasia, and these will require extra connections to the micro:bit.

Fortunately the Kitronik motor controller has space for a row of headers which will allow access to all the micro:bit’s features. If you want to add the extra components you will need to install the headers which need to be soldered.

You don’t have to do this when you first build Anastasia. I added my header later, and you can do the same. If you do, you will need to take Anastasia to pieces, add the headers and then assemble Anastasia again.

If you want to future-proof Anastasia by soldering on the header, you will need a 21-pin male header with 0.1" spacing. Male header strips are widely available and it’s easy to break off the length that you require.

Kitronik Controller Board

You can see the holes for the headers in this photo of the Kitronik board.

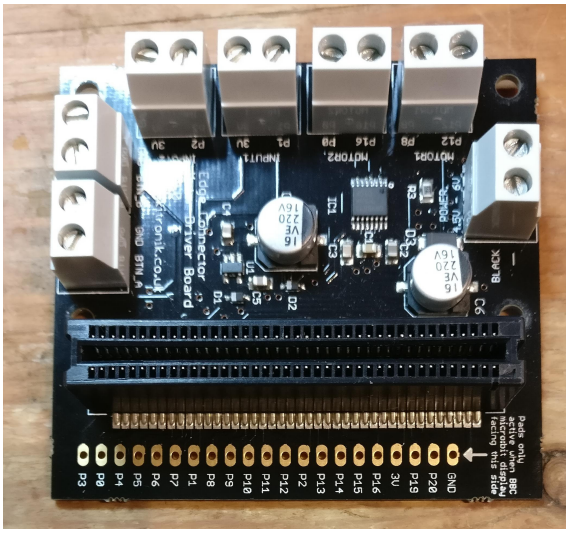

Figure 1: Kitronik Controller Board

Board and Headers

Here is the board with a header row ready to be soldered.

The headers need to be upright, and I kept them in place while soldering by using a strip of electrical tape to hold them in place.

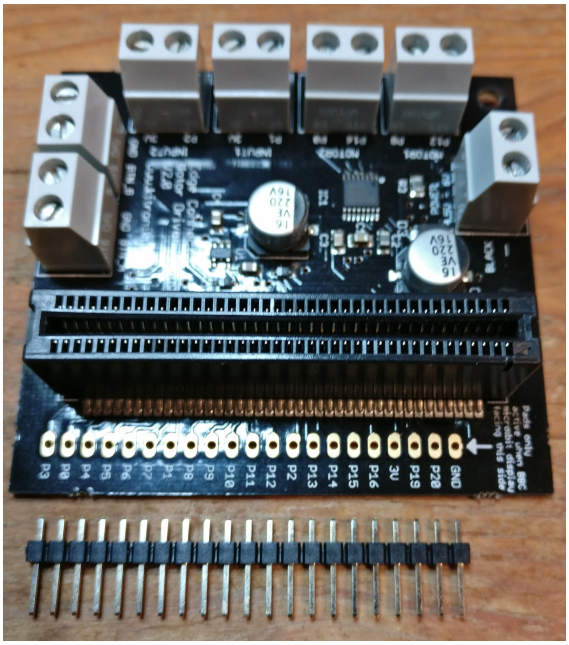

Figure 2: Board and Headers

Headers soldered in

Here are the headers after they have been soldered.

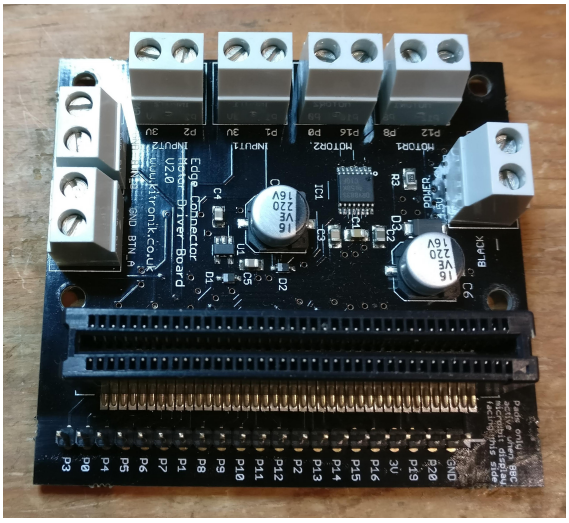

Figure 3: Headers soldered in

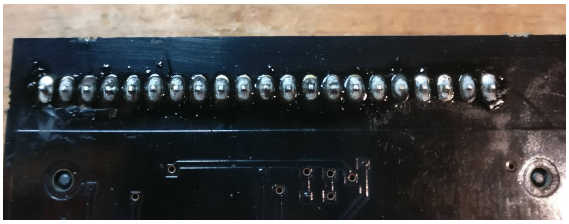

Figure 4: Soldered headers from below

And finally, here's a picture of the soldered headers with their labels on the board.

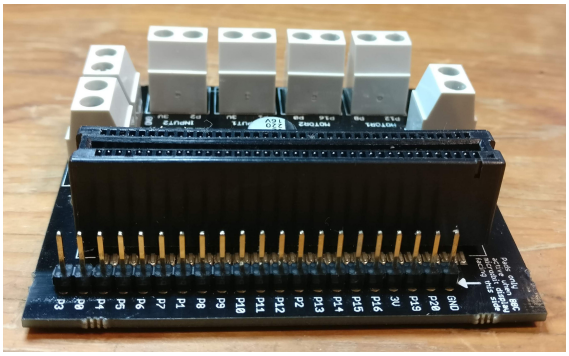

Figure 5: Headers close-up

Main build

Whether or not you decide to add headers to the board the remaining steps are the same.

Meet the components!

Here are the main components I used to build Anastasia.

The controller just needs a micro:bit and a battery holder with two batteries.

You'll also need to provide power to Anastasia. You can chose to power Anastasia with 4 AA batteries in a suitable holder or you can use a phone charger; details of each are covered below.

The components you can see here are

1. The base-plate
2. The caster with bolts and nuts
3. The micro:bit
4. Two motors
5. Two wheels
6. the Kitronik Motor Controller
7. Spacers, bolts and nuts for mounting the motor controller
8. Two motor mounting brackets with bolts and nuts

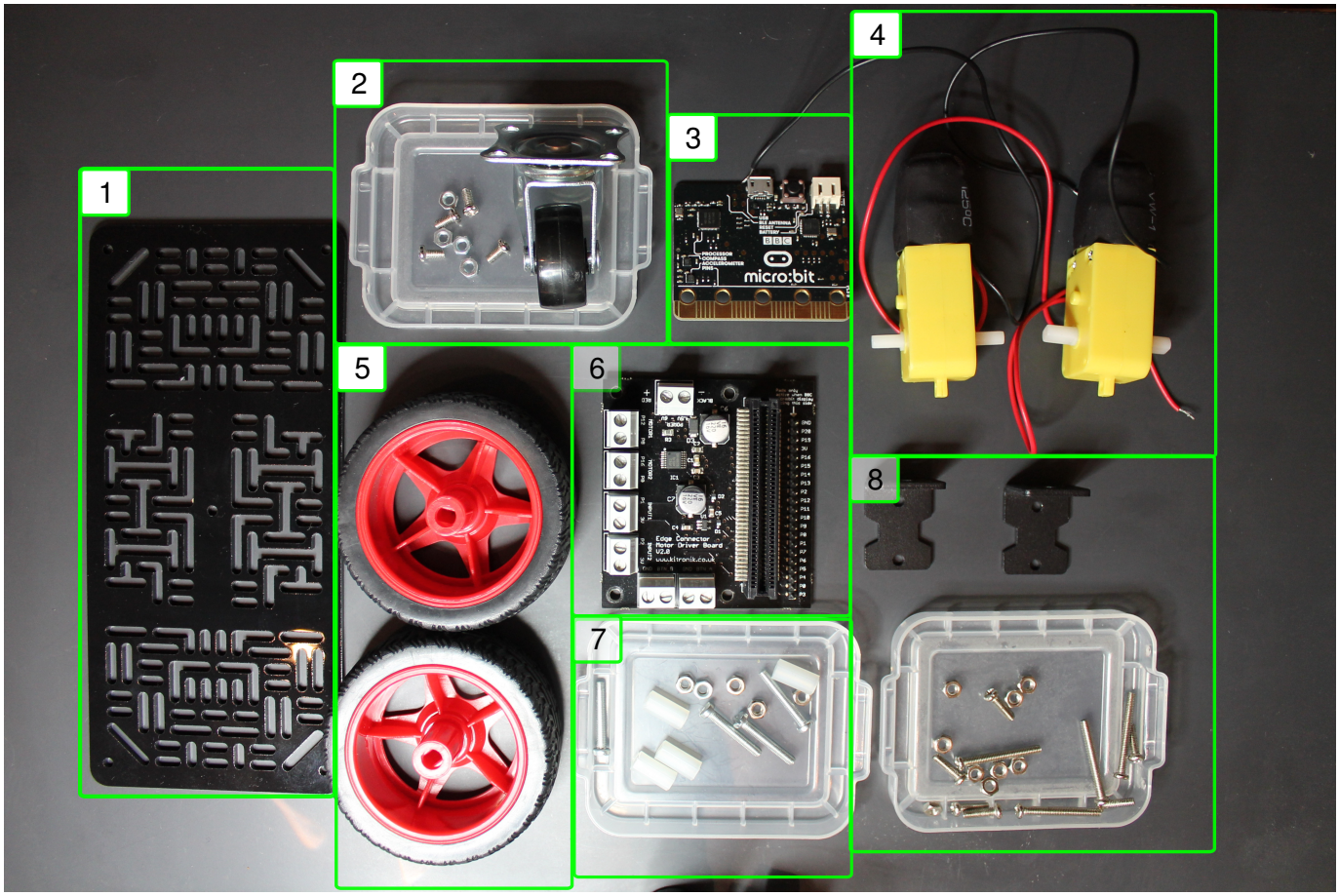

Figure 6: Meet the components!

Adding the caster

The Adafruit caster comes with its own set of cross-head bolts and nuts.

Figure 7: Adding the caster

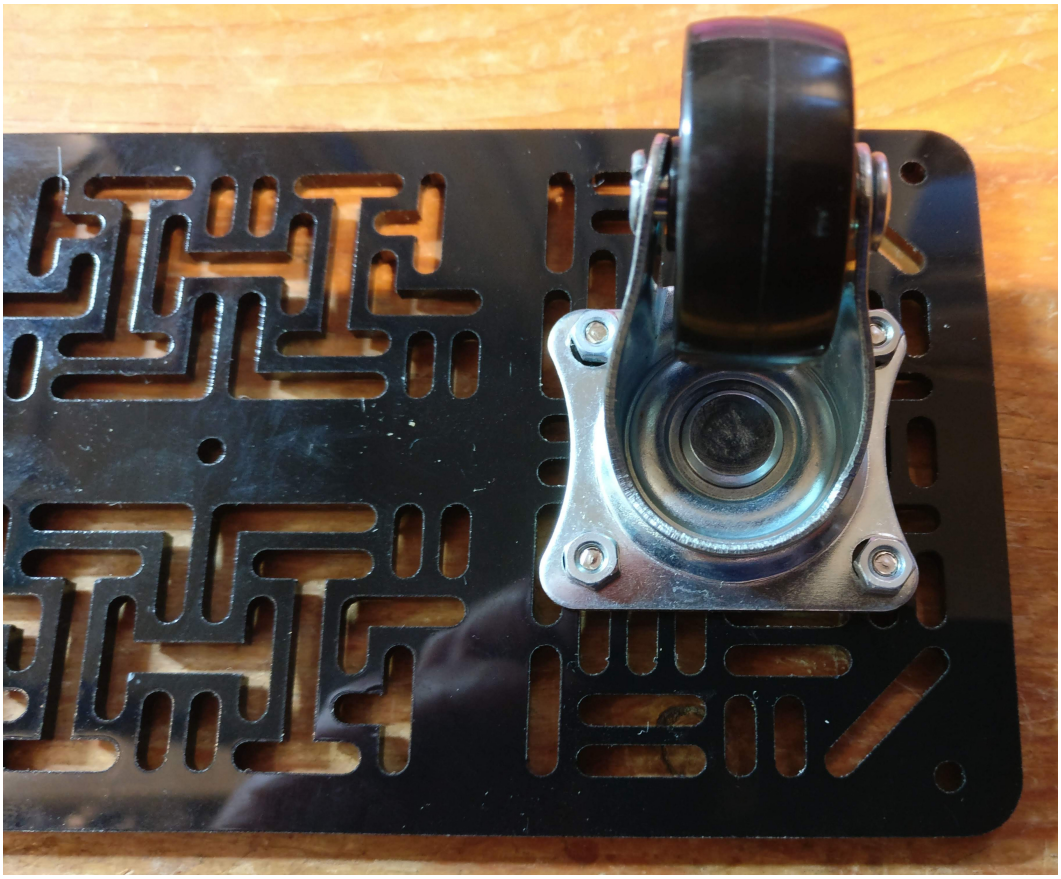

Figure 8: The caster attached

Attaching the Motor Controller to the base

The controller is attached to the base by four bolts with separators (sometimes called stand-offs).

Push the bolts down through the controller's mounting holes.

Put the separators on to the bolts below the controller. Some separators are threaded, in which case you'll need to rotate them until they are snug below the board.

Now push the free ends of the bolts thought holes in the base. You may need to experiment a bit to find holes that work well.

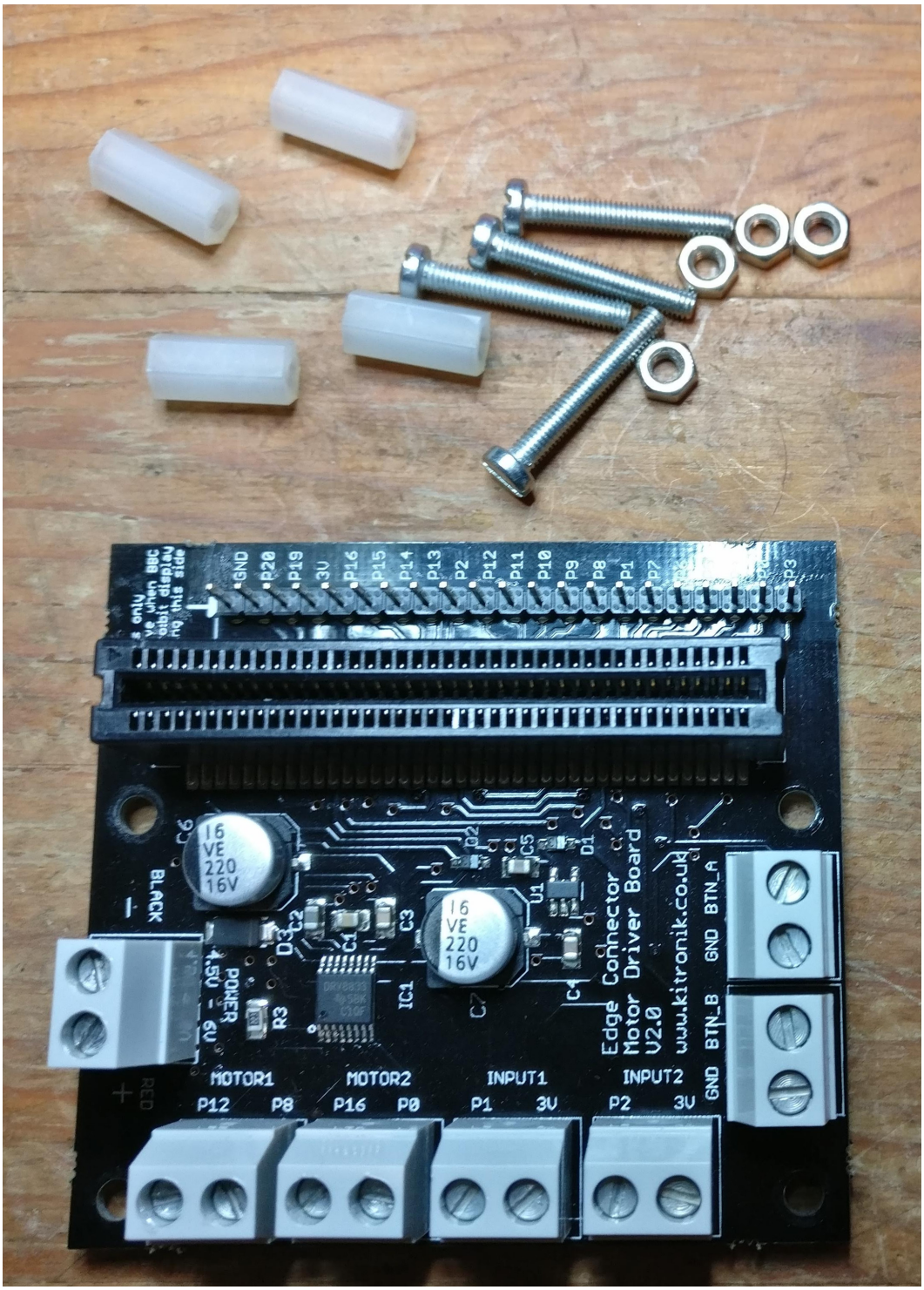

Figure 9: Attaching the Motor Controller to the base

Seen from below, the controller board should look like this.

Figure 10: Motor controller with spacers

Attaching the controller to the base

Insert the ends of the bolts into slots on the case.

You may need to wiggle it around to find convenient mounting holes.

I couldn't get the controller completely square to the board, but this didn't seem to matter.

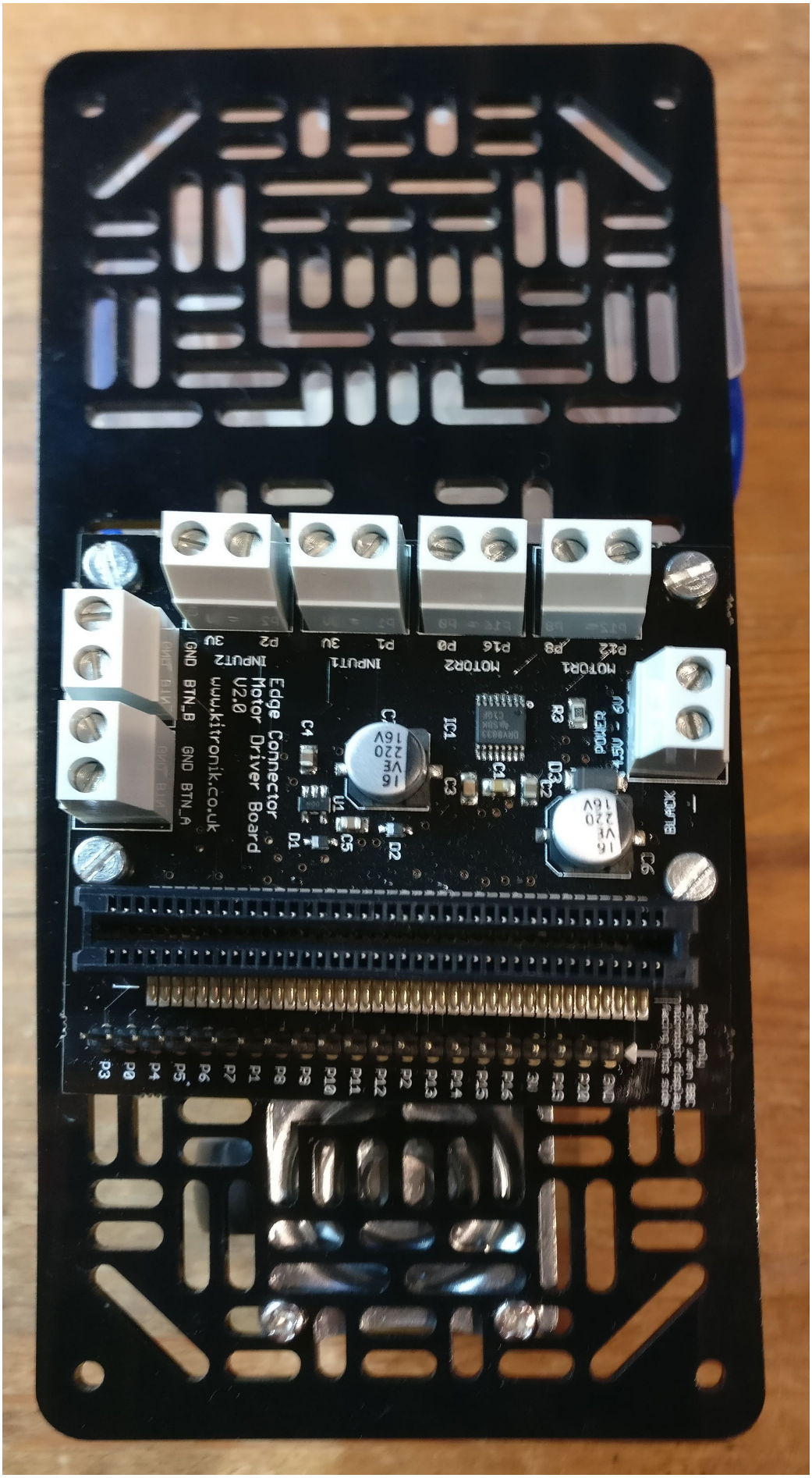

Figure 11: Attaching the controller to the base

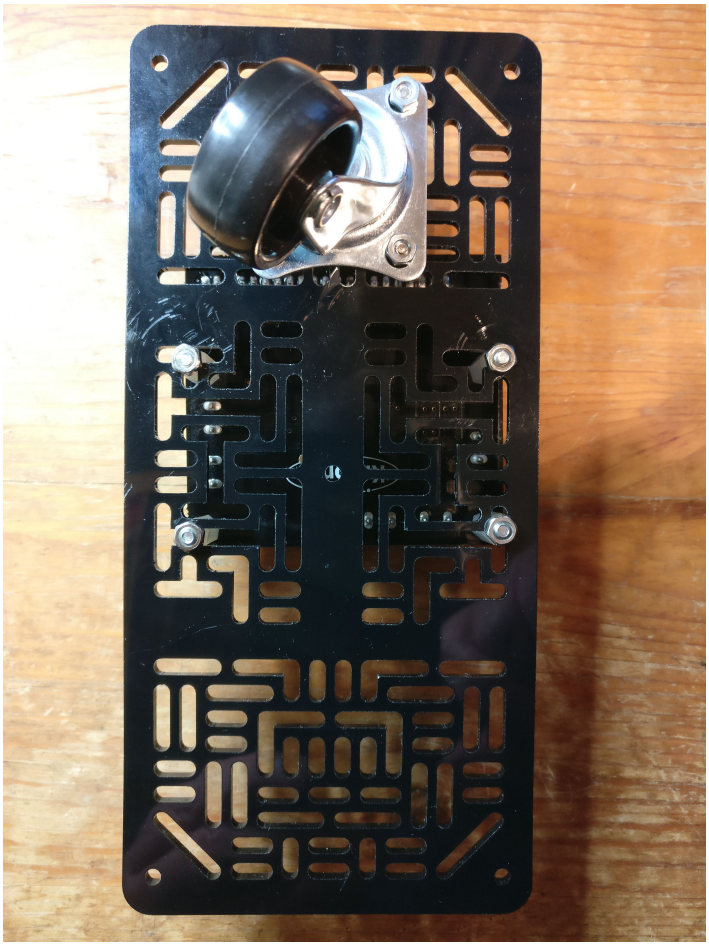

Figure 12: The base from below

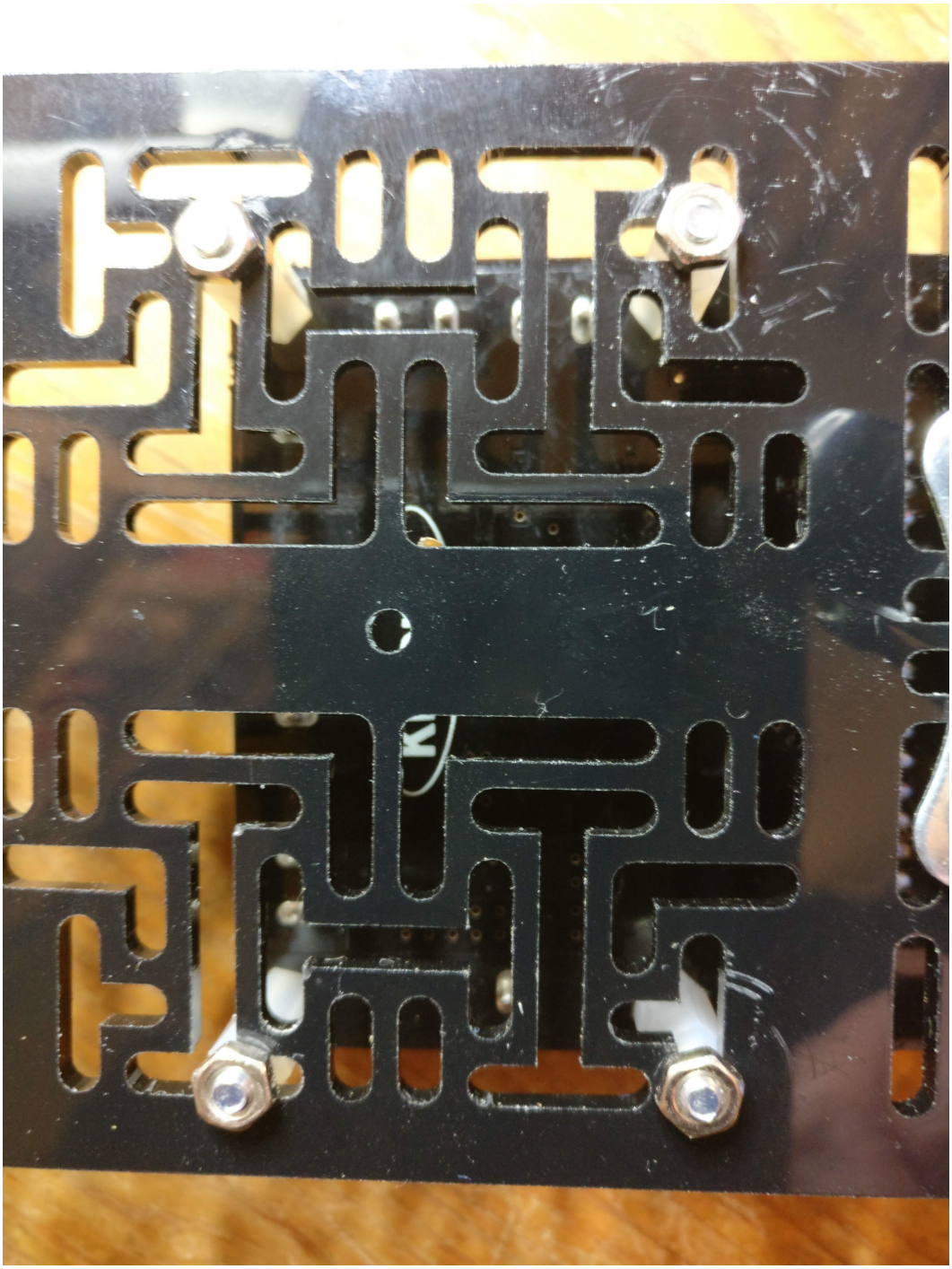

Figure 13: Close-up

Attaching the motors

Here's one of the two motors, together with its mounting bracket, nuts and bolts.

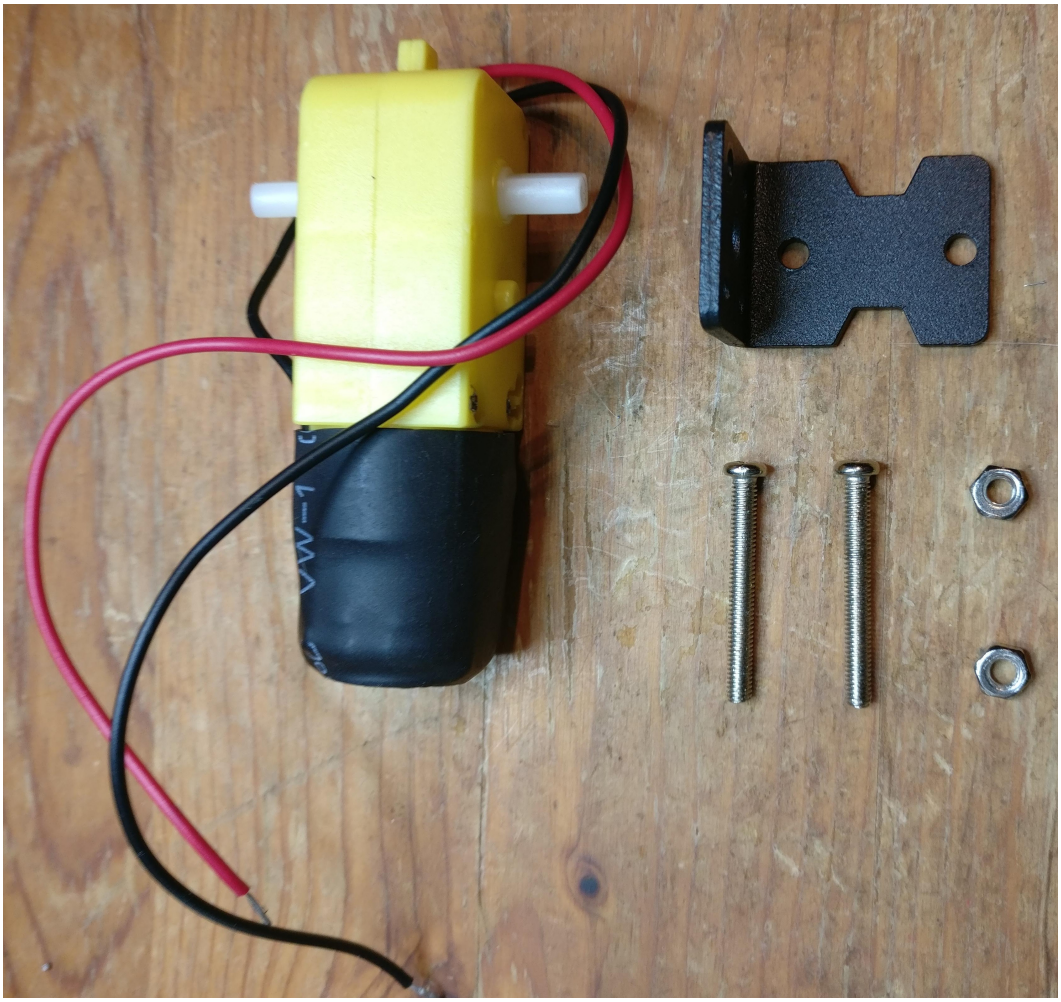

Figure 14: Attaching the motors

Fits, use the long bolts to attach bracket to the side of the motor.

Make sure the correct bracket is on the correct side of the motor!

Then repeat for the second motor.

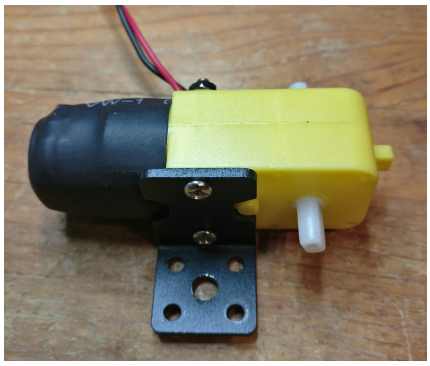

Figure 15: Bracket and motor

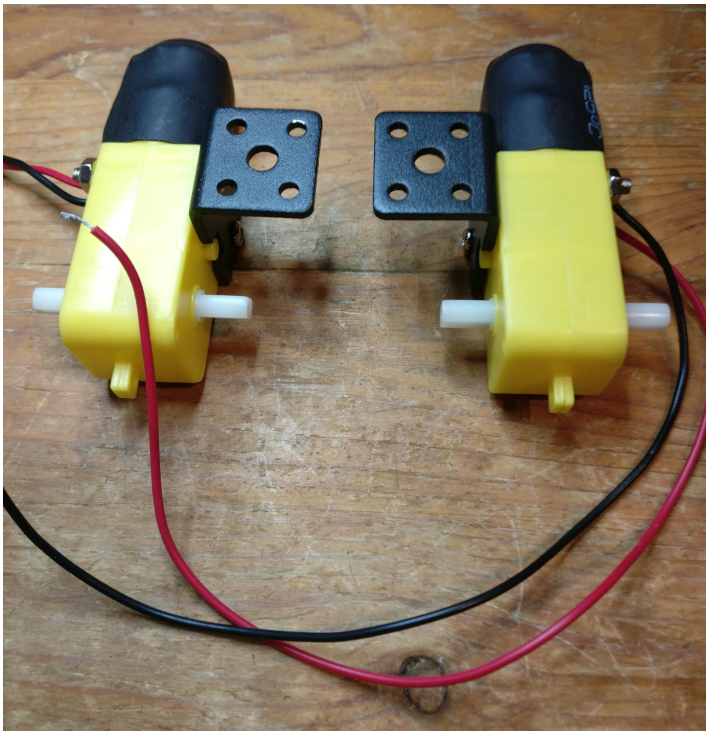

Figure 16: The two motors with their brackets

Attaching the wheels to the motors

Push each wheel onto the spindle of its motor.

You'll need to line the hole of the wheel up with the spindle, and you will need to push quite hard.

Make sure each wheel is on the correct side!

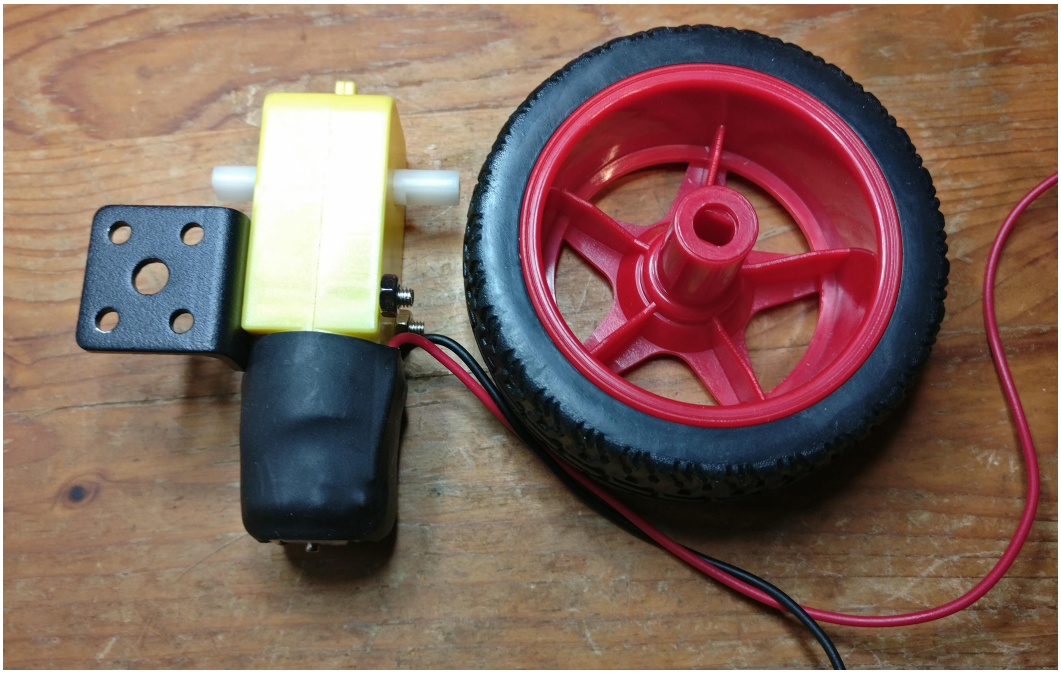

Figure 17: Attaching the wheels to the motors

Attaching the first motor

Using the four short bolts, attach the first motor to the base.

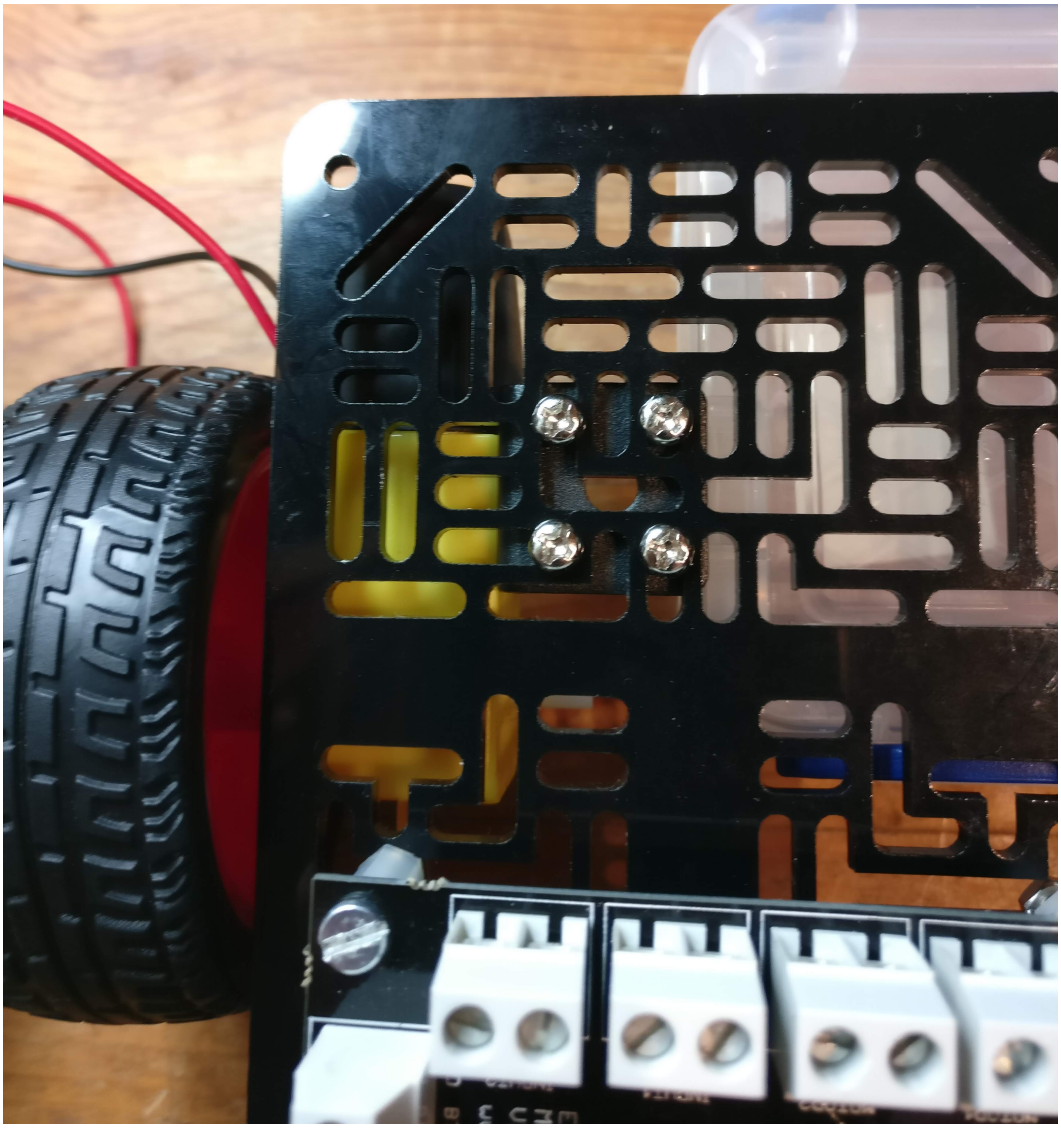

Figure 18: Attaching the first motor

Attaching the second motor

There isn't enough space to place the motors symmetrically.
Here's how I mounted the second motor.

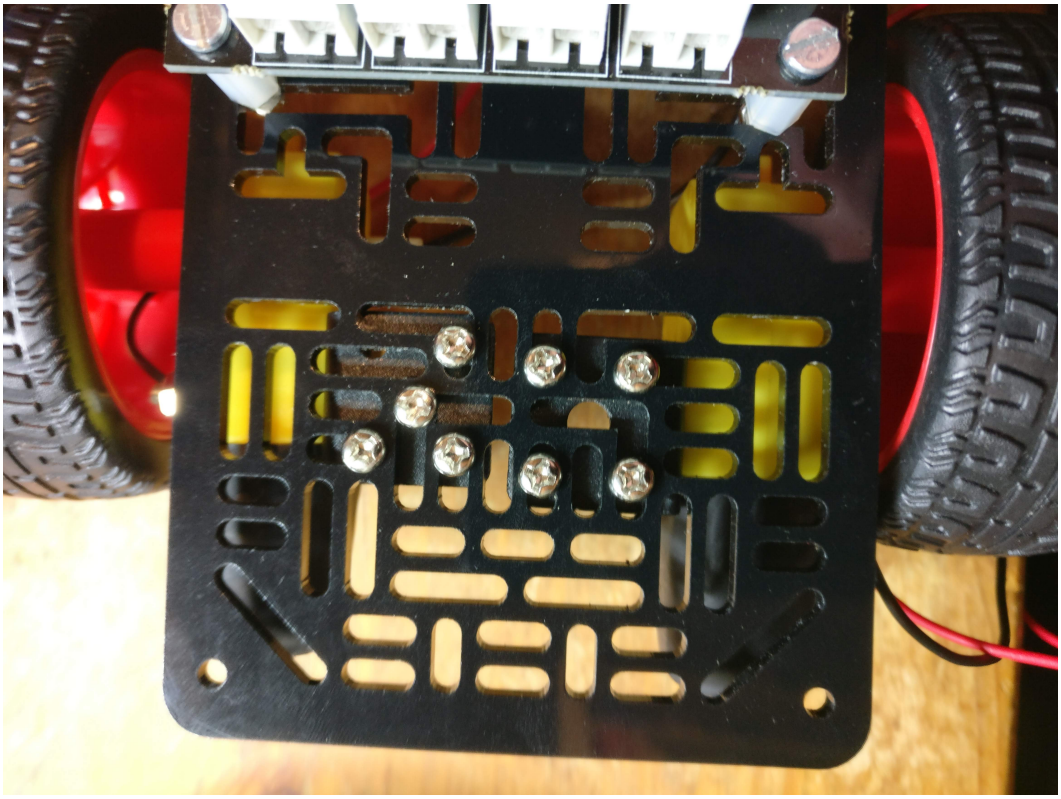

Figure 19: Attaching the second motor

Motors, battery and controllers.

Here's a close-up oif the controller, the mounted motors and the phone charger battery I used.

That version uses a little stripboard circuit to connect the charger to the controller's power.

I'll add details of the stripboard circuit later.

If you decide to use AA batteries you may need to experiment to find out where to place it on the base, and you may need to make your own mounting holes in the battery holder.

Connect the wires from the battery holder to the power inputs on the motor controller.

Be sure to insert the red wire in the plus socket and the black wire in the minus socket.

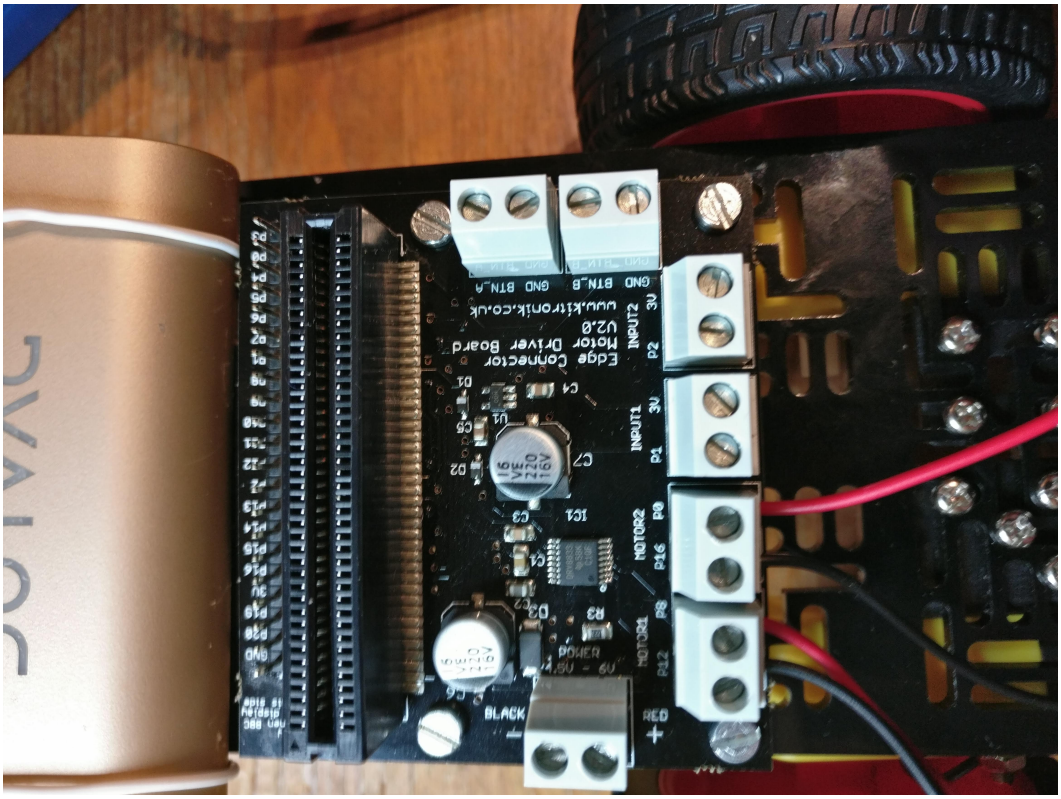

Figure 20: Motors, battery and controllers.

Controlling Anatasia

Controller Code

The code for Anastasia and her controller is simple.

You control Anastasia via a separate hand-held micro:bit, which communicates using the micro:bit's built-in radio.

You tilt the controller and Anastasia responds by advancing, retreating or spinning to the left or right. Anastasia stops when you hold the controller level.

The controller code starts with a bit of set-up

```
from microbit import *
import radio

radio.on()
# The critical value for tilt detection.
# A lower value makes the controller more sensitive.
RANGE = 250
```

Next, the *say* function checks to see if there is a command to send and sends it if necessary.

```
def say(command):
    """
    Send a command over the the radio

    :param command: The outgoing command, or None if a command was not required.
    :return: True if there was a command to send
    """
    if command is not None:
        radio.send(command)
        return True
    return False
```

The micro:bit has a built-in accelerometer which can be used to detect if the micro:bit is tilted or level.

react_to_tilt checks a value to see how much the micro:bit is tilted.

If the size of the tilt is less than the RANGE value the tilt is ignored. If the tilt is big enough, the appropriate command is returned.

```
def react_to_tilt(tilt, low, high):
    """
    Determine the command to send based on how the controller is tilted.

    :param tilt: the tilt angle of the micro:bit in the x or y direction
    :param low: the message to send if the tilt is below the critical value
    :param high: the message to send if the tilt is abovee the critical value
    :return: a message, or None if the micro:bit is held level
    """
    if tilt > RANGE:
        return high
    if tilt < - RANGE:
        return low
    return None
```

The check_tilt function measures the tilts in the x and y directions and sends the appropriate command over the radio.

```
def check_tilt():
    """
    Send the appropriate command based on how the micro:bit is tilted.
    """
    if say(react_to_tilt(accelerometer.get_x(), 'right', 'left')):
```

```
        return
    if say(react_to_tilt(accelerometer.get_y(), 'forward', 'backward')):
        return
    # micro:bit is more or less level, so stop the robot.
    say('stop')
```

The final while loop runs forever. It looks for commands to send ten times a second.

```
while True:
    """
    Loop forever looking for commands to send ten times a second.
    """
    sleep(100)
    check_tilt()
```

Anastasia's code

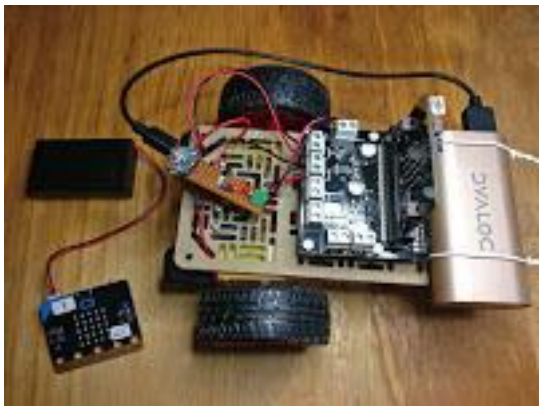

Figure 21: Anastasia and controller

As you saw earlier, the controller can send one of five commands to Anastasia via the micro:bit radio:

1. **stop** - applies a brake to the motors.
2. **left** - spins the robot anti-clockwise.
3. **right** - spins the robot clockwise.
4. **forward** - (you guessed it!) drives the robot forwards.
5. **backward** also does what you'd expect - it drives Anastasia backwards.

Anastasia's code looks for incoming radio messages. It translates those messages into methods on a Driver class. These methods then send control signals to the Kitronik motor controller.

```
from microbit import *
import radio


class Driver:
    """
    Read incoming radio messages and send commands to the motor controller.
    """
    def __init__(self, left_pins=(pin16, pin0), right_pins=(pin8, pin12)):
        """
        Configure the pins and map commands to bound methods.

        :param left_pins:
        :param right_pins:

        See https://blog.rareschool.com/2020/03/driving-anastasias-kitronik-motor.html for more details.

        See https://www.kitronik.co.uk/pdf/5620-motor-driver-board.pdf for Kitronik's data sheet.
        """
        self.pins = left_pins+right_pins
        radio.on()
        self.commands = {
            'stop' :    self.stop,
            'forward':  self.go_forward,
            'backward': self.go_backward,
            'left':     self.spin_left,
            'right':    self.spin_right,
        }
```

The Driver's `__init__` method allows you to pass in the pins you use to control the motor, but has default values recommended by Kitronik in their documentation for their Motor Controller.

The left and right motor pins are concatenated and stored in the `pins` instance variable which is used later.

The `commands` variable contains a dictionary which maps text commands to methods on the driver class.

Let's look at the definitions of the methods.

```
    def stop(self):
        """
        Set both motors to brake.
        """
        self.set_control_pins(1, 1, 1, 1)

    def go_forward(self):
        """
        Set both motors to run forward.
        """
        self.set_control_pins(0, 1, 0, 1)

    def go_backward(self):
        """
        Set both motors to run backwards.
        """
        self.set_control_pins(1, 0, 1, 0)

    def spin_left(self):
        """
        Set left motor to run backwards, right motor to run forwards.
        """
        self.set_control_pins(1, 0, 0, 1)

    def spin_right(self):
        """
               Set left motor to run forwards, right motor to run backwards.
               """
        self.set_control_pins(0, 1, 1, 0)
```

Each method uses the set_control_pins __ method to apply high (1) or low (0) voltages to the pins that are connected to Anastasia's two motors.

```
    def set_control_pins(self, *pin_values):
        """
        Set the control pin states for the motor controller.
        :param pin_values: four values for the four control pins
        """
        for pin, pin_value in  zip(self.pins, pin_values):
            pin.write_digital(pin_value)
```

Handling commands

When Anastasia tries to read a command over the radio it will either get the next command as a text string, or `None` if no command has been received.

The `obey` method does nothing if its argument is `None`. If a message *has* been received `obey` looks it up in the `commands` dictionary. If it's an unknown command `obey` ignores it.

If it's a known command `obey` translates the command to the corresponding method. Then the relevant method is executed.

```
    def obey(self, message):
        """
        Map a valid message to a method and run it.

        :param message: the message to obey, or None if no message was received.
        """
        if message is None or message not in self.commands:
            return
        # execute the bound method if a valid message was received.
        self.commands[message]()

    def run(self):
        """
        Loop forever looking for messages to obey.
        """
        while True:
            self.obey(radio.receive())
            sleep(50) # 50 ms
```

Finally, the script creates a Driver instance and tells it to start running.

```
Driver().run()
```

Installing the software

Use the editor of your choice to install *controller.py* on the controller micro:bit and motoring.py on Anastasia's micro:bit.

I used mu.

Remember that as soon as both are powered Anastasia will start to move as directed by the tilt of the controller.

I made sure that Anastasia was powered off when I installed the software on the controller, and that the controller was level when I turned Anastasia on.

Pro tip: *Don't* run Anastasia (or any other mobile robot) on a tabletop unless you've taken special precautions. There's nothing worse that seeing your creation tumble off the edge of the table and come to pieces as it falls on the floor.

Have fun with Anastasia. and let me know how you get on. I'm @rareblog on Twitter.

Appendix

About the Author

Figure 22: Romilly Cocking

Romilly Cocking has been programming computers since 1958!

He has been programming in Python for almost 20 years.

Now in his 70s, he still loves learning and sharing what he's learned with others. He's written several books about single board computers and physical computing

He's a contributor to several Open Source projects, and in 2012 he founded Quick2Wire, a start-up that made add-on boards for the Raspberry Pi.

He's a regular presenter at the London-based Raspberry Pint meetups hosted at Microsoft Reactor London.

You can read his blog about Raspberry Pi, AI, Robotics and Electronics.

Other titles by Romilly

Free guides *immediate download - no email required*

Share these free guides with your friends!

Build Anastasia the micro:bit robot

Power Strip Build Guide

I2C and SPI Guide

Low cost books on Leanpub

These books are copyright but inexpensive or free.

Published

- Making the Shrimp Build your own Arduino clone!
- micro:bit MicroPython in 60 Minutes *Free*
- A Simulation of Cerebellar Cortex *Free* -My MSc project report on Neural Networks from 1974.

In progress

- Explorer HAT tricks fun things to try with your Pimoroni Explorer HAT.
- Learn APL on the $5 Raspberry Pi

You can buy books on leanpub *risk-free*.

- You can get sample contents free, so you can find out if you want to buy it before spending any money.
- You have 45 days to request a no-quibble refund, so you've lost nothing if it doesn't live up to expectations.
- You get free updates for the books you've purchased when these become available.

- You get instant access to DRM-free content in *pdf*, *mobi* and *epub* formats.

Updates, new offers, discounts and a chance to become a rewarded reviewer

Join my low-volume email list.

- I won't share your information, I won't spam you, and you can leave the list at any time.
- You'll be the first to know about new guides, books, courses and updates.
- You'll be offered exclusive discounts on books and other products.
- You'll get an opportunity to become a reviewer for new books and courses. Reviewers get *a reward* once they have completed the review process.

Sign up now!