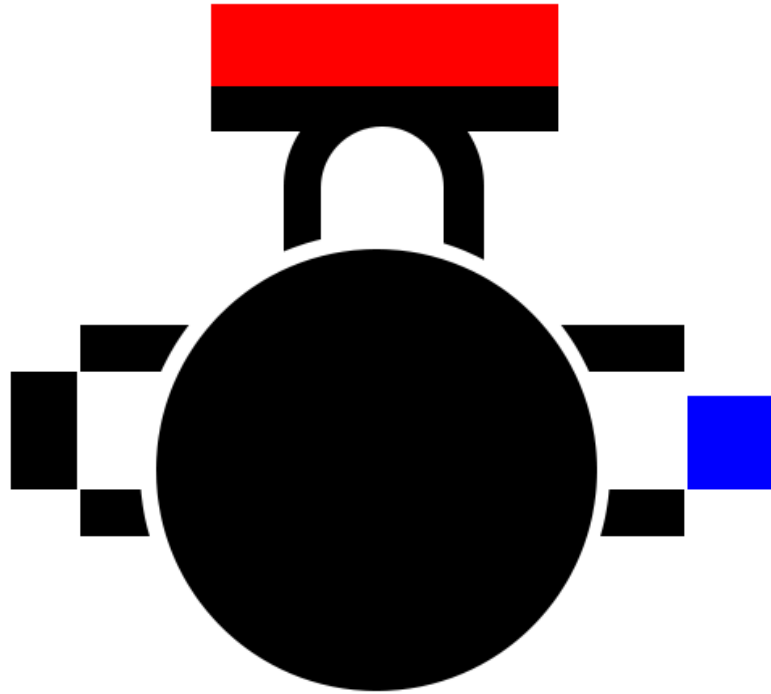


Build a Hull Pixelbot



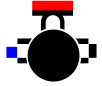
Rob Miles

Version 1.0



Table of Contents

Introduction	3
Why do I need to build a robot?	3
How much will this cost me?	3
What will I learn?	4
How does the course fit together?	4
Is it all really easy?	4
Is there any sample code?	5
What do I need to get started?	5
Tools	5
Software	6
Components	6
Robot Chassis	6
Set up HullIOS	7
What you will need	7
Install HullIOS on the Arduino Uno	7
Test the drive motors	12
What you will need	12
Connecting the motor boards	12
Connect the battery	15
What you will need	15



Introduction

Welcome to the world of robotics. If you've ever wondered what it takes to build a robot you can find out right here. In this book we are going to build a Hull Pixelbot. This is a little robot from Hull that is designed to be cheap and easy to make, infinitely extensible and great fun to play with.

All the software and the robot designs are free and open source. You can find them here:

github.com/HullPixelbot

You can find out the latest HullPixelbot news here:

www.hullpixelbot.com

You can download the sample code here:

github.com/HullPixelbot/Learning

Why do I need to build a robot?

Actually, you don't. The Hull Pixelbot is great fun to play with, but it won't help you wash the dishes or clean the car. It's only very little and its powers are comparatively weak, so it is no good for world domination either.

But the skills that you are going to get from building the robot will enable you to make some really useful devices. You'll find out how to create programs that can control physical hardware. The Hull Pixelbot is actually a bunch of active components on the end of some software. Just like every other robot that's ever been made.

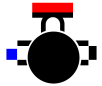
So, you can take your skills and use them anywhere you like. If you think your life (or perhaps that of your cat) would be improved by a remote controlled, networked cat-flap, then building a Hull Pixelbot will give you the skills to realise this dream.

How much will this cost me?

One of the wonderful things about the world today is just how easy and cheap it is to get powerful hardware. For less than the price of a burger you can pick up a tiny computer that can be connected to the internet and used to read sensors and control devices.

We will be building our robot at "pocket money" prices. The aim is to get you an effective robotic platform for less than the price of a video game. And we will be building the robot in stages, so you can pick up the parts as we go along. At the start of each chapter we'll have a "shopping list" of things you'll need to complete this stage of the robot development.

The motor chassis is designed to be 3D printed. If you have a friend who owns a 3D printer you can give them a chance to show off their printing skills by making you the parts that you need. Printing an entire chassis takes around 8 hours or so of printing time, and it uses around 10 metres of filament, which should cost less than five pounds. If you are having problems finding someone to print your robot, I'd strongly advise you to seek out a local "hackspace". These are clubs run with the aim of sharing hardware and expertise. If you go along with the robot designs they might even print some robots of their own.



What will I learn?

You'll learn how to create programs and run them. You'll also discover how programs can be made to talk to hardware, and the fundamentals of the electronics that makes the hardware work. Specifically, the book will cover:

- Building hardware. This is the fun bit. You'll discover how to fit components together and make them work. Hopefully you'll also gain the confidence to take your own ideas and translate them into hardware.
- Simple electronics. You'll learn the difference between analogue and digital systems and some design principles which will mean you can create devices that work and don't disappear in a puff of smoke when you turn the power on.
- Simple software development. You'll start with Python-Ish, a language you can use to give your robot simple behaviours. You'll plug your robot into your computer update the program it runs. Later we will move onto Python and add wireless network connectivity.

How does the course fit together?

The course is broken into chapters. Each chapter will explore a particular feature of the robot. At the beginning of each chapter I'll describe how to construct the hardware, then we'll take a look at the way this hardware is used from the software. Finally, we'll finish with some things you might like to try. During the text I've got some conventions



This indicates an activity you should perform in at this point in the text. You may be given precise instructions, or you may have to work something out for yourself.



This indicates something that you may want to think about later.

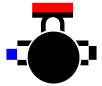


This indicates a warning to be careful about this bit.

Is it all really easy?

No. There are some bits that are a bit tricky. When you build an electronic device that is controlled by software you get a "double whammy" of problem potential. When it doesn't work, you have to figure out whether the hardware is faulty or your program has a bug in it. This is difficult even for the most experienced engineers. But during this text you'll learn techniques that will help you deal with these problems.

As you struggle to get your robot to do what you want you'll gain a new respect for the people that take technology like this and package it into your phone, TV or washing machine. But you'll also gain the skills that make it possible for you to think about making the next generation of products like these.



Is there any sample code?

Yes. Within each chapter there'll be references to sample Arduino projects that you can run. You can download the sample code from github:

<https://github.com/HullPixelbot/Learning>

What do I need to get started?

If you want to build a robot you'll need some hardware tools and some software tools. These aren't expensive, and they'll serve you in good stead during your robotics career. You'll need hardware tools to build the robot, some software tools to create the program that controls it, and some components to actually fit together. Then you'll need the components to actually build the robot from.

Tools



Figure 1 These are the tools that I've used to make all my robots.

On the left I have my indispensable wire cutters and strippers. These let me cut wire to length and remove the insulation from the metal core, so I can make connections. Next along you have some "snipe nose pliers". These are good for holding things and squeezing things together (which has the professional name of "crimping").

Next along are a pair of screwdrivers. One has a "slotted" end which is flat. I can use this for tightening screws that have a slot in the top. The next one along has a "Phillips" or "cross-head" tip, which I can use for cross head screws like the ones in the terminal blocks that we are going to be using.

The last two tools I'm rather proud of. They are "2mm hex drivers". I use these to tighten the bolts that hold the HullPixelbot together. You can use ordinary headed bolts (with a slot or a cross head), but I've found that bolts that you can tighten with these drivers are much easier to work with. There's no chance



of the tool slipping out of the head of the bolt when you tighten it. When we look at the parts that you buy, I'll tell you where to get these bolts. They aren't much more expensive, and they are a lot easier to work with.

If you look at my toolkit you'll notice that every tool is from a different supplier. I've just picked them up as I've gone along and replaced ones that I've lost or broken over the years. You don't need to buy an expensive toolkit full of shiny things you'll never use, I've found that you can get along just fine with the above.

It is really nice if you can set up a little place to work on your robot. Assembling robots on the sofa in front of the TV is possible (I've done this) but it is very frustrating when you drop a screw down the side of one of the cushions and then have to spend ages searching for it. If you can find a reasonable area of desk space that is well lit, you'll find it much easier to work there.

Software

The software tools that you want are all free, which is nice. To start with you'll not be writing programs from scratch, instead you'll be modifying ones that I've written. The tool that you will be using is the Arduino IDE. You can download this for free from:

<https://www.arduino.cc/en/Main/Software>

There are versions of this software for Windows PC, Mac and Linux computers. You can download and install the software now if you like. We'll investigate how to use it later in the text.

Components

There are two kinds of hardware components you'll need. You'll need the electronic components that power the robot and make it work and you'll also need the robot chassis, which is made up of 3D printed elements.

Robot Chassis

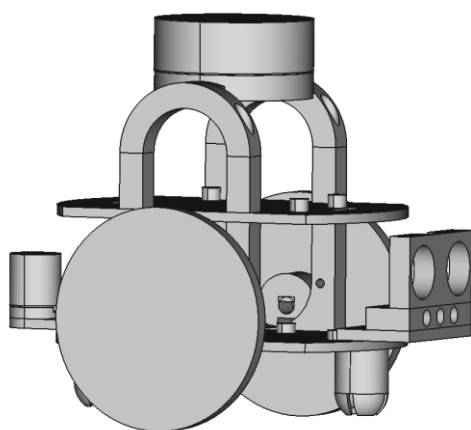
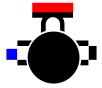


Figure 2 These are the Computer Aided Design (CAD) designs for the Hull Pixelbot

Above you can see the "skeleton" of a complete Hull Pixelbot. It comprises several elements that you bolt together. At the start of each chapter, I'll identify the elements that you need to print to complete that phase of the robot development.



Set up HullOS

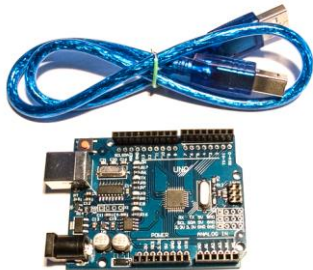
What you will do in this chapter

In this chapter you are going to get the software running that will control your robot. This software is called Hull OS and it runs inside the tiny computer which will control the robot. Once you have HullOS running you'll use it to turn the motors that will move our robot around.

What you will need

These are the things that you need for this stage.

Arduino Uno



This is the processor used to drive the motors and read the sensors of the HullPixelbot.

The blue cable is used to link the Arduino to a usb port on your computer. You'll use this to put your programs into the Arduino. Search for "**Arduino Uno**" on your favourite parts supplier site. Any device which is advertised as "Uno compatible" will work fine. Make sure you get one that comes with a usb cable.

Install HullOS on the Arduino Uno

The first thing we are going to do is install HullOS on the Arduino Uno. This provides the software environment we will use to control what the robot does. We will create Python-ish programs (in a language a bit like Python) and send them to the robot where they will be stored, ready to run each time the robot is switched on. You don't need to know anything about Python, or indeed programming to get started.

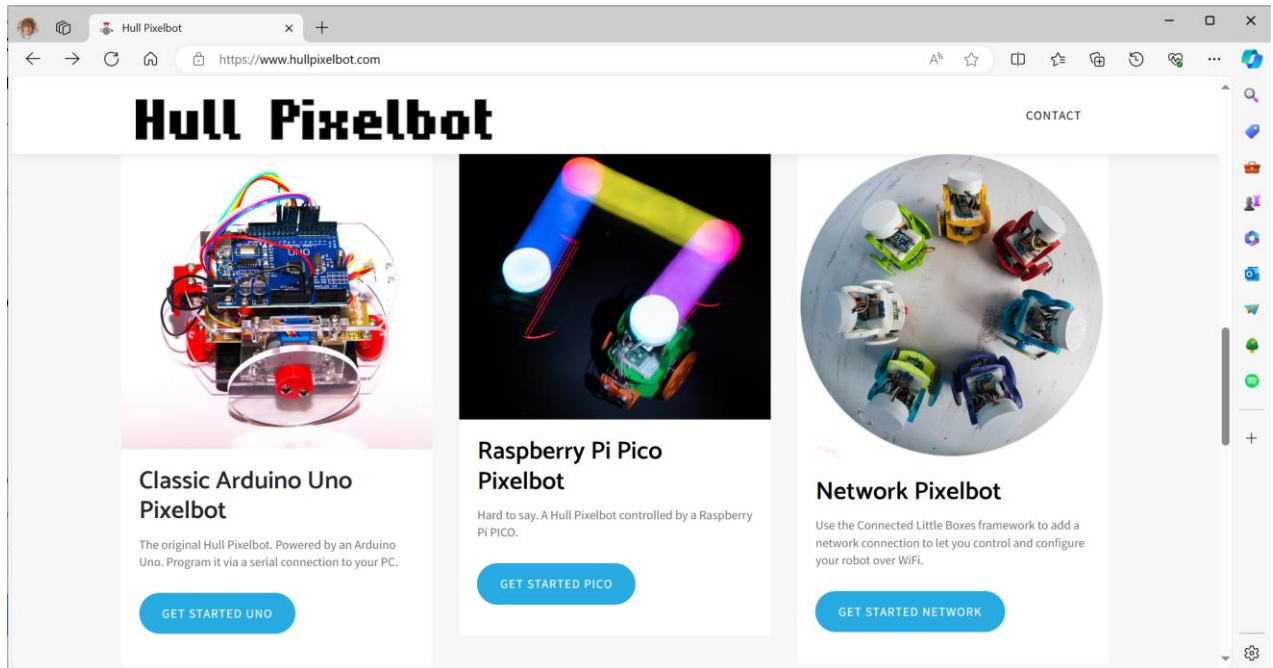
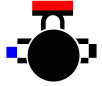


Use the cable to connect your Arduino Uno to your PC or laptop. The Arduino will be powered on but it will not do much.

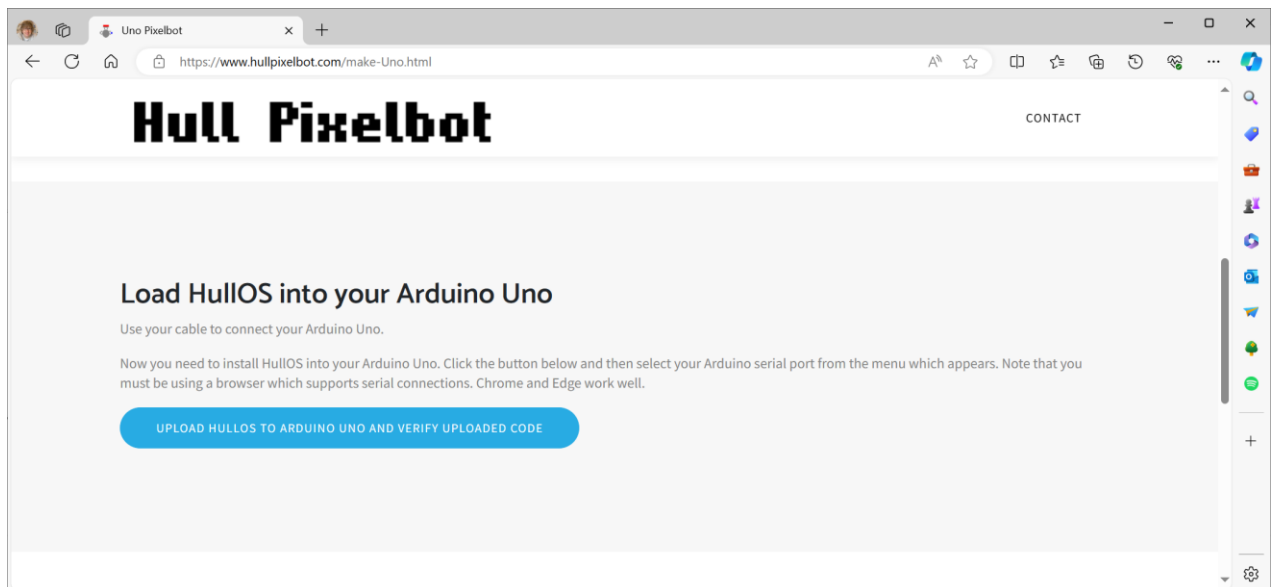


Open your web browser (Chrome or Edge – Safari and FireFox can't be used for this) and browse to the Hull Pixelbot site:

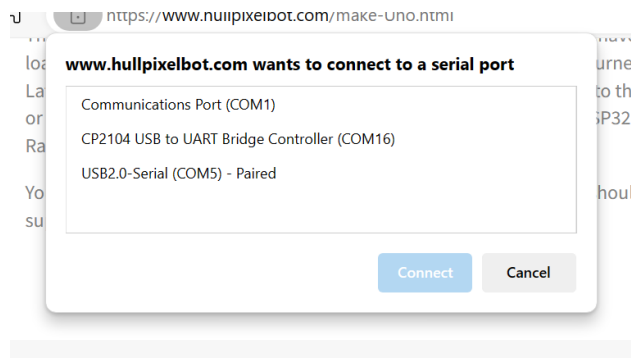
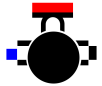
www.hullpixelbot.com



Scroll down the page until you reach the section shown above. Then click the **GET STARTED UNO** button for the Classic Arduino Pixelbot. We will be exploring other Hull Pixelbots later. This will open the page we are going to use to upload the HullOS software into our Arduino Uno.



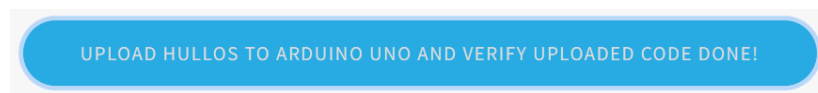
Scroll down the page until you reach the section shown above. Then click the **UPLOAD HULLOS TO ARDUINO UNO AND VERIFY UPLOADED CODE** button.



If you are using the Edge browser on Windows you will see the message above. If you are using a Mac or a different browser the message might look a bit different. The browser wants you to pick a serial port. I happen to know that the Uno is connected to COM5 on my machine. If you are not sure which port your Arduino is connected to, click Cancel and unplug the Arduino and try again. The Arduino was connected to the port that has disappeared.



Select your Arduino and click **Connect**. The content of the button will now show the progress of the upload.



If the upload completes successfully you will see the message above. Scroll down to the next part of the page.

Congratulations

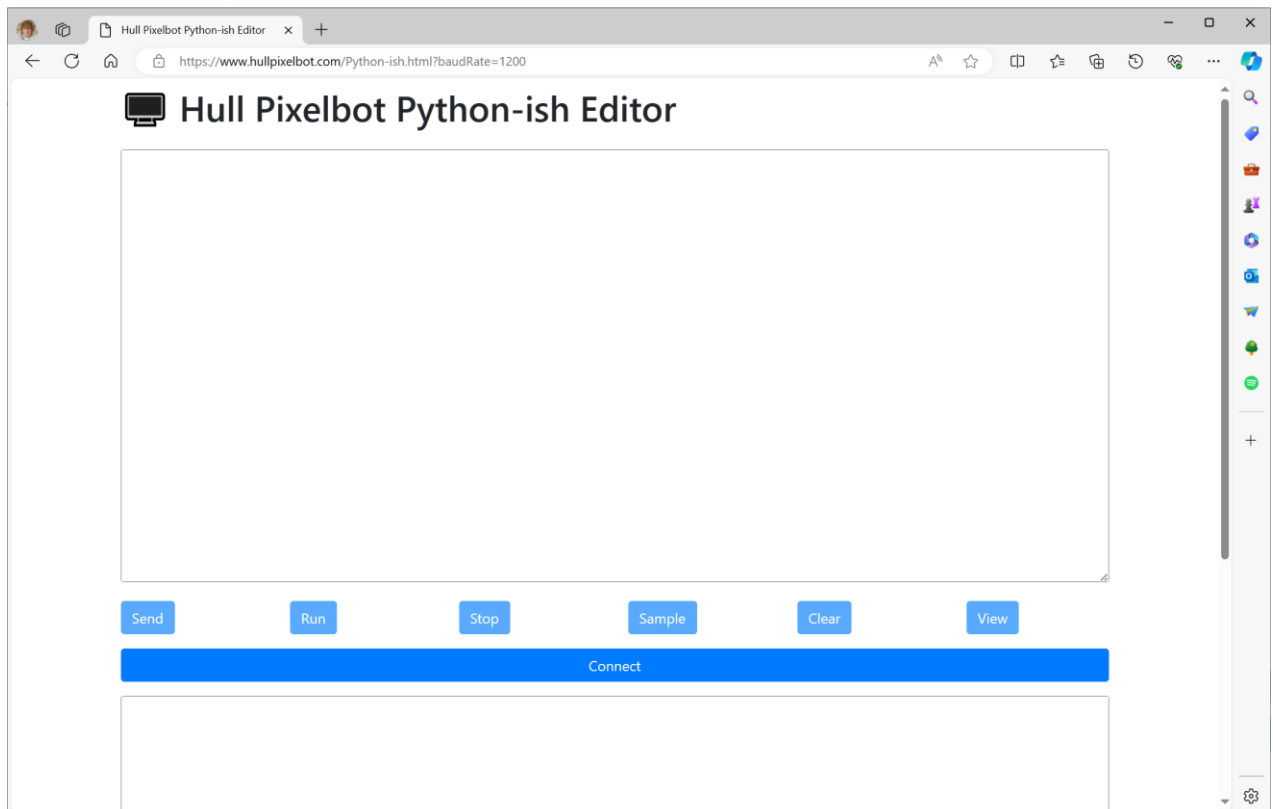
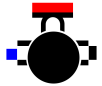
You should now have an Arduino Uno running HullOS. Visit the [Python-ish](#) page to connect to your Arduino and run your first program.

If you have any problems you should check out the [faq page](#).

If the upload fails you may have to install some extra drivers. Use the **faq page** link to the FAQ (frequently asked questions) at the bottom of the page.



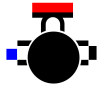
Click the **Python-ish** link to open up the Python-ish editor in your browser.



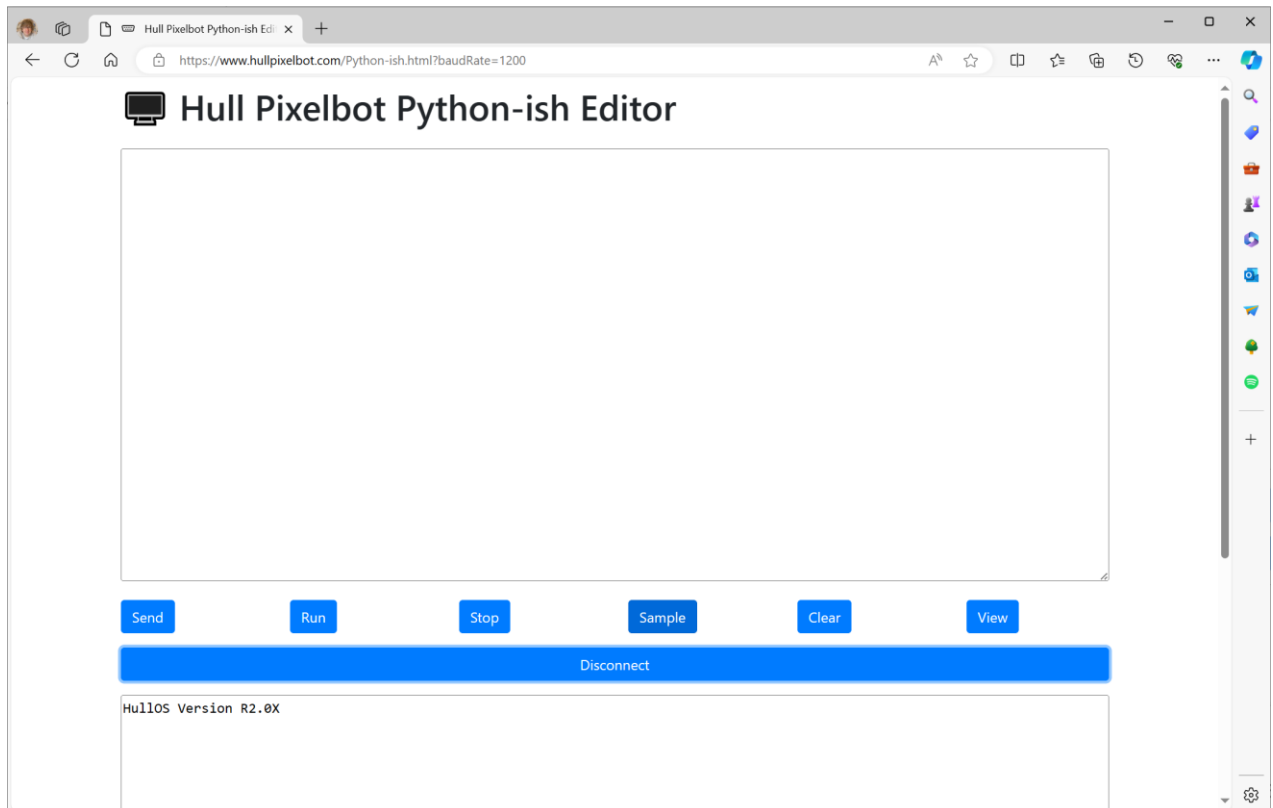
The Python-ish editor is used to create programs and load them into a pixelbot. The top half of the screen is where you would enter your programs. The Send, Run, Stop, Sample, Clear and View buttons let you work with program code and control the program running in the Pixelbot. The Connect button is used to connect the editor to the robot.



Click the **Connect** button to connect the editor to your Arduino.



You will be asked to select a serial port. Select the same one that you used to load HullOS into your robot. The button will show the status of the connection and the logging window underneath the Connect button will show replies from the robot.



Once the robot is connected the screen should appear as above. The log window shows the version of HullOS running in the robot. You can now use the buttons to interact with the robot. You could type a Python-ish program in at this point, but it is easier to press the **Sample** button and step through some examples.

```
# square dance
forever
  move 100
  turn 90
```

The sample program above makes the robot move in a square, repeatedly moving forwards and then turning 90 degrees.



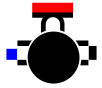
Click the **Sample** button until the square dance example is shown. Then click **Send** to send the program to the robot.

The log window should show the message "OK" to indicate that the program is now loaded into the robot.



Click the **Run** button to run the program. Nothing will happen. But the Arduino is now generating signals that would control the motors.

The next thing we need to do is connect the motors so that they will run and move the robot.





Test the drive motors

What you will do in this chapter

We now have a program running inside our Arduino which is supposed to be moving the motors. But at the moment they are not connected. So now we are going to connect the motors to the Arduino and get them turning..

What you will need

These are the things that you need for this stage.

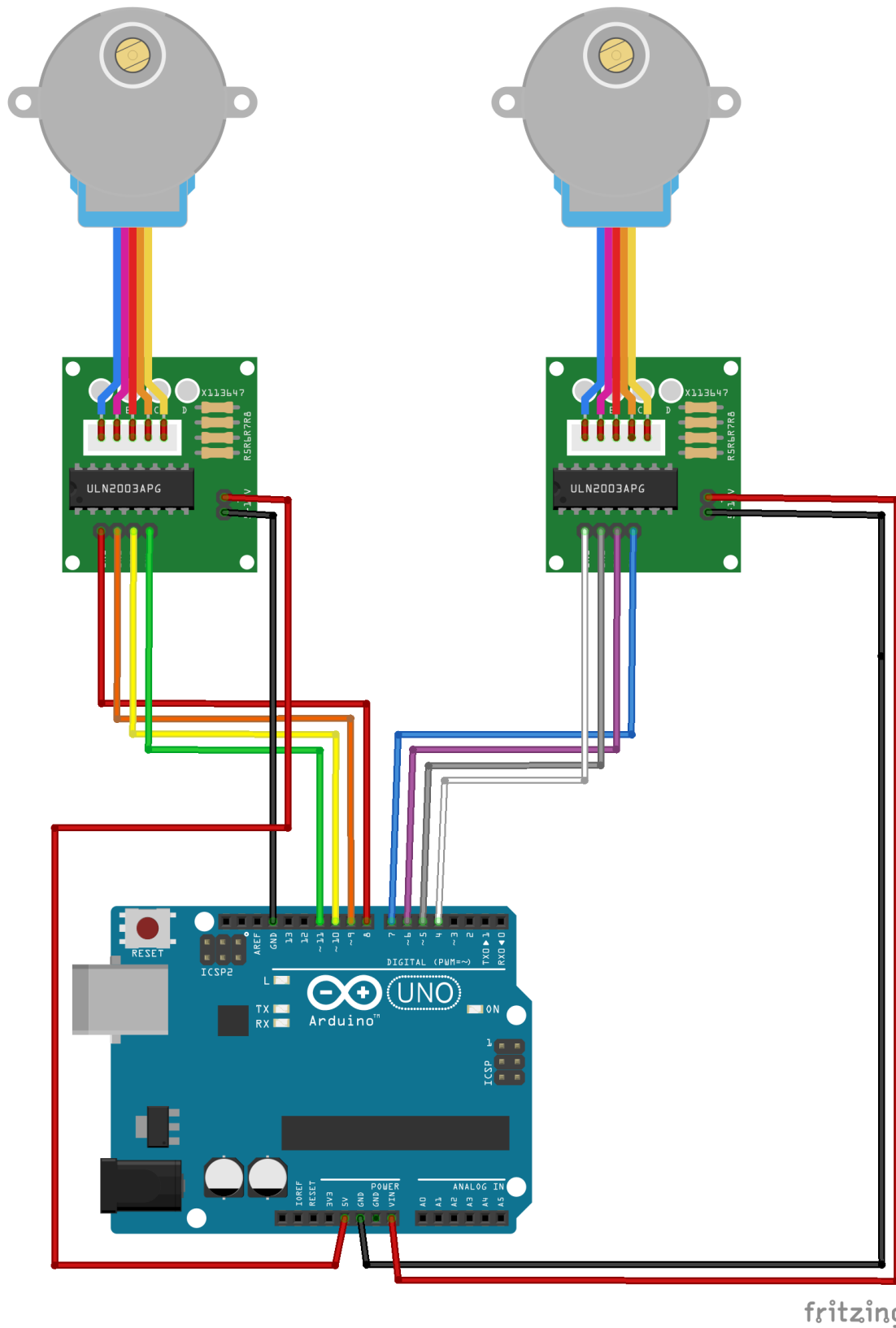
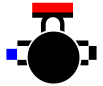
<p>Stepper Motors</p> 	<p>These little motors will provide the power to move your robot around. You'll need one for each wheel. You can use them for lots of other projects too.</p> <p>Search your favourite parts supplier for "Arduino stepper"</p>
<p>Jumper cables</p> 	<p>Use 20cm male to female jumper cables. You'll need six per motor, making twelve in all. These are usually sold in "ribbons" of 40 cables, you then just peel off the ones that you want to use.</p> <p>Search for "Arduino jumper" on your favourite parts supplier site. Make sure you buy male to female (M-F) cables. The motor controller has pins on it and the Arduino has a socket on it. You might like to buy some male to male and female to female cables as well for stock. Some devices have sockets on them that you might want to connect to your robot.</p>



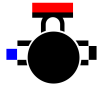
Make sure that you disconnect the Arduino from the computer before you do the next bit.

Connecting the motor boards

The stepper motors that we are using to drive the robot each contain four coils that become magnetic when you pass current through them. These coils attract a magnet on a shaft inside the motor and pull the shaft round, making it turn a tiny bit.



We are going to connect our motors and control them from HullIOS. The connections are as shown above.



Use the jumper cables to make the above connections. The colours of each wire aren't important, but they need to connect the correct terminals, otherwise the motor will not turn. Once the jumper cables have been connected, plug the motors into the driver boards.



Be particularly careful with the power connections (the red and black cables the connect to the sides of the driver boards as shown above). If these are connected the wrong way round you could damage the Arduino or the driver board.

If you loaded the "Square Dance" program into your Arduino Uno at the end of the last exercise this will run when the robot is switched on. If not, connect the Arduino Uno to your computer and download it.



Plug the USB cable from your computer into the Arduino. You should see the motors moving and leds flashing on the driver boards. If the motors don't turn properly you may need to check your connections.

Congratulations. You've proved that the Arduino is running and that you can control motors with it.



You can change the move commands in the Python-ish program to make the motors do different things. How about modifying the square dance to create a program that you could use to drive two windscreen wipers on a car?



Connect the battery

What you will do in this chapter

We now have a working computer and some working motors. But they must be plugged into a computer to work. This is no good if we want our robots to be able to roam wild and free. So, we need to add a battery pack. This will power the Arduino and the motors.

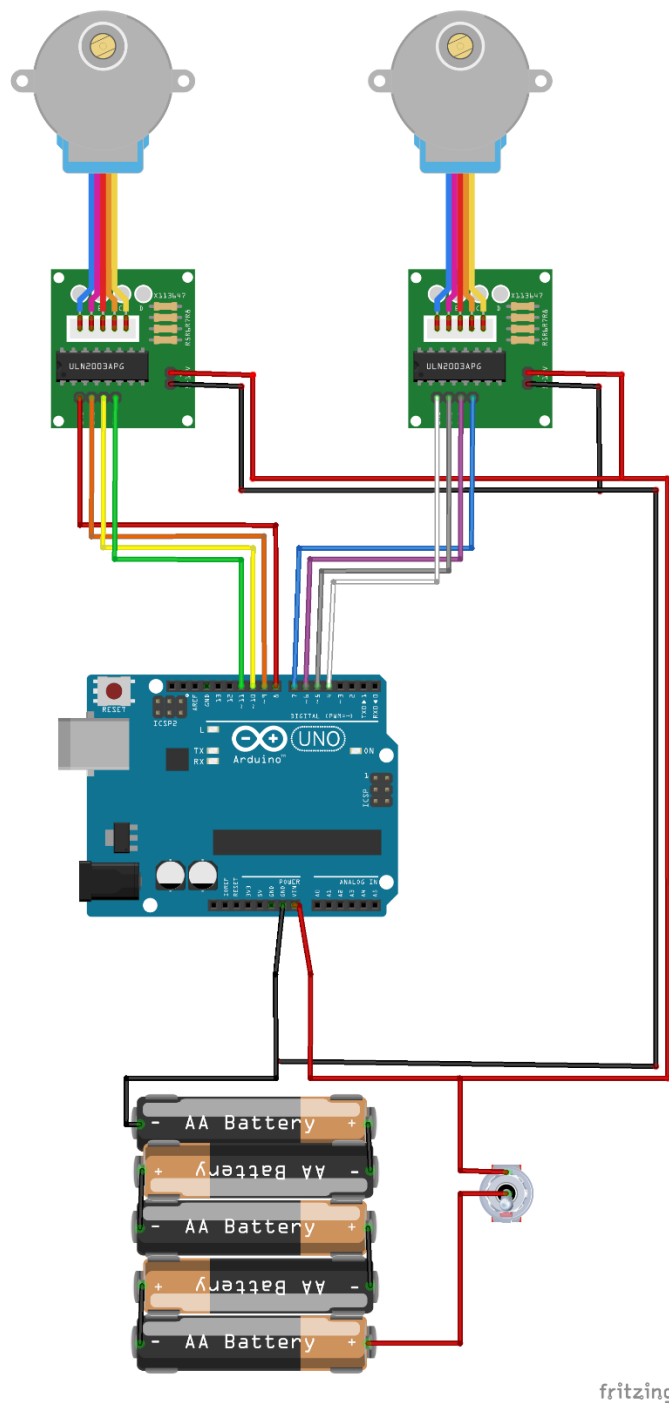
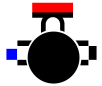
What you will need

These are the things that you need for this stage.

Battery box and wiring



This battery box has a built-in switch. The output is connected to a terminal block which splits the connections to provide two for the motor driver boards and one which goes into the Arduino Uno. The box is secured to the bottom of the robot using double sided tape.



The battery box can hold four AA batteries. The diagram above shows how this works. Note that the output from the batteries is connected to the ViN connection on the Arduino Uno.



Connect the battery box as shown above. When you turn the box on you should see the motors turn.

Congratulations. You've now got a pair of portable computer controlled motors.



The Python-lsh language contains a delay statement which can make a program pause. You could use this to make a device which suddenly wakes up and turns the motors after an interval. It's a bit late for Halloween, but this would make a great scary thing.