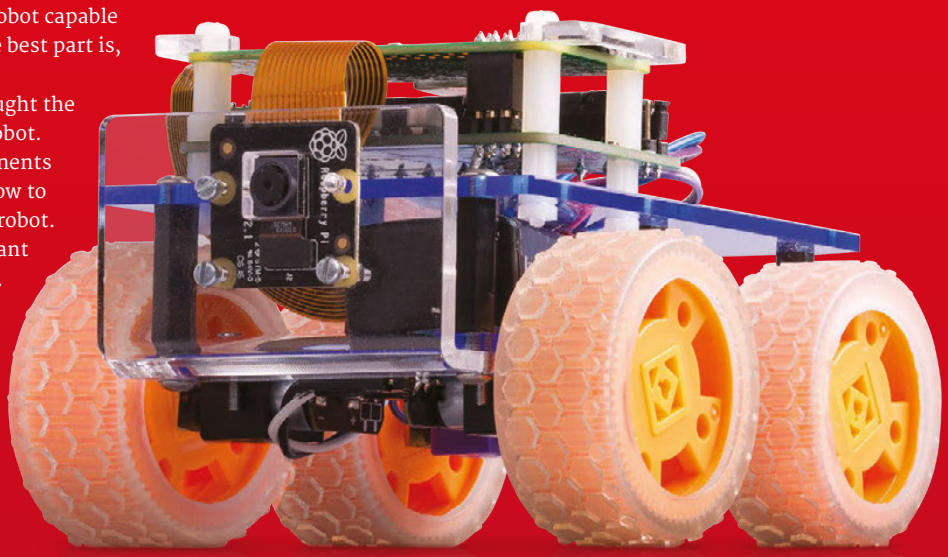
BUILD A REMOTE CONTROL ROBOT

Make a Pi Wars-winning robot with our expert **Brian Corteil**

This is no ordinary robot build. Over the next few pages, we are going to show you how to design, build, and program a robot capable of winning a robotics competition; the best part is, anyone can make one!

In this feature, we will take you thought the steps required to build this amazing robot. We'll look at different types of components you could select, how to program it, how to build it, and then how to control your robot. We'll even have some tips for if you want to enter your robot into a competition. Read on and we can get started...



SELECT YOUR STYLE

What kind of robot do you want to make?

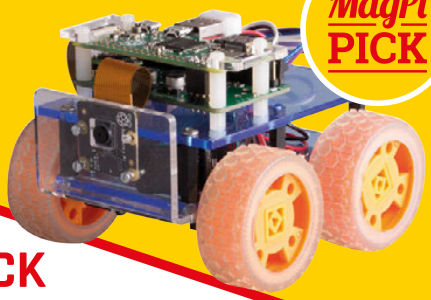
Robots come in many configurations. The type we will be looking at are commonly called rovers (ROV or remotely operated vehicle). There are many types of rovers, including the classic two-wheel tail dragger, omnidirectional, tank tracked, four-wheel, and the six-wheel Mars rover. Each type has its pros and cons.

FOUR-WHEEL DIRECT DRIVE

This is the robot we're building. The four-wheel direct drive chassis is a common design for a robot, and can be bought as a kit from many stores. Four-wheel direct drive robots have many advantages over two-wheeled robots: they have more torque, more grip, they're faster, and are able to turn on the spot. This makes the robot nimble and very suitable for challenges like the maze and the obstacle course.

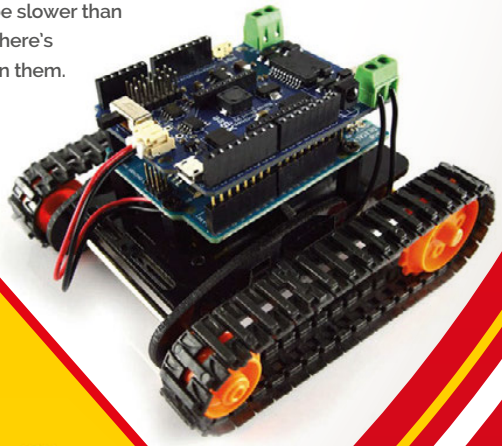
Disadvantages include a higher cost due to the greater number of motors and more expensive motor controller to use them.

The MagPi PICK



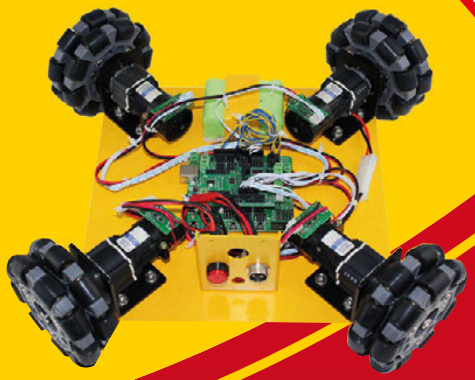
TANK TRACK

Tank tracks are cool. They have loads of grip when matched to the surface you are running on, are great over rough ground, and you can turn on the spot. However, they can difficult to build. The alignment of the tracks needs to be spot on when installing them, or you will risk the track slipping off. Tracks can also be slower than wheels because there's more friction on them.



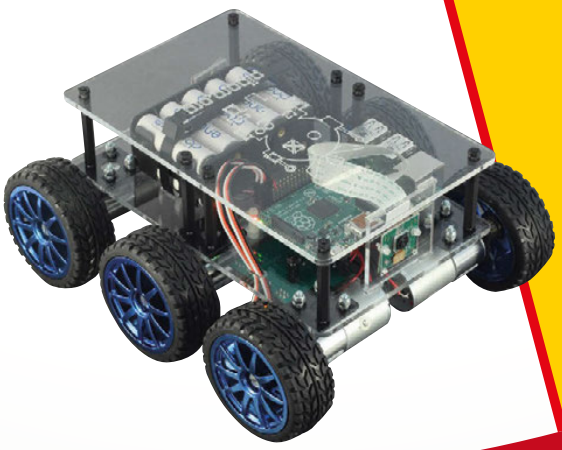
OMNI

Omnidirectional robots are great for avoiding your opponent in sumo-style contests like Pi Noon at Pi Wars, as they can move in any direction. The downside is they are not so great on rough ground due to the design of their wheels. The wheels are complex and can be quite expensive to buy, and you'll need to do a lot more coding on them. The maths behind getting it to work can be very cool, but it's also scary enough to make you want to run and hide behind the sofa.



SIX WHEELS

Six-wheel direct drive robots have most of the advantages of both tracked and wheel robots, offering good grip, high torque, and manoeuvrability. The use of multiple motors balances their output. Disadvantages for this type are the cost of six motors, wheels, and a motor controller powerful enough to supply the juice required.



PICK YOUR MOTORS

How will the robot get around everywhere?

Motors come in many sizes, types, with/without gearboxes, etc. Common motor types used in maker projects are brushless, brushed, and stepper (see more details on the right). When selecting a motor, you need to consider its voltage rating, free-run current, stall current, and type. The free-run (or operating) current is how much current the motor will draw when running; with this information, you'll be able to estimate how much power the robot needs.

This is important for a reliable design, as you'll need to match the motor and controller characteristics; if you use a motor that has a voltage rating higher than the motor controller, you won't get the full potential out of the motor. If the stall current is higher than the controller's peak output current, you'll see 'magic smoke' come from the controller, which is bad. Another characteristic of the motor you need to consider is its speed: the higher the RPM of the motor, the faster it will turn the robot's wheels/tracks. For this robot, we're using the micro metal 6V N20 motors from Pimoroni.

MOTOR TYPES

BRUSHLESS

PROS:
- ⊕ No brushes to wear out
- ⊕ More torque
- ⊕ More power
- ⊕ Better control
- ⊕ Power-to-size ratio

CONS:
- ⊖ Complex & expensive controls
- ⊖ Less rugged
- ⊖ Cost

BRUSHED

PROS:
- ⊕ High torque at lower speeds
- ⊕ Simple control circuits
- ⊕ Cost

CONS:
- ⊖ Shorter operational life
- ⊖ Power-to-size ratio

STEPPER

PROS:
- ⊕ Precise control (turns in small steps)

CONS:
- ⊖ Complex controls
- ⊖ Cost
- ⊖ Slow

Our tiny robot is using this tiny micro metal brushed motor

MOTOR MARKINGS

MOTOR MARKINGS

Some things you'll see on a motor...

DC: 12V

The rated voltage of the motor. Can also be shown as '12V DC'. DC means that the motor must be powered by a direct current source like a battery.

RPM

Revolutions per minute. Can also be shown as r/min. The lower the number, the slower the motor is, so 500 rpm is faster than 25 rpm.

GEAR BOX RATIO

Not pictured on this motor. If a motor output is described as a ratio, the lower the number, the faster the motor, so 50:1 is faster than 254:1.

SELECT A MOTOR CONTROLLER

Now we have motors, we need a way to use them

Motor controllers, like the name suggests, are used to control motors. There are many types available for the Raspberry Pi, so when selecting the motor controller, you need to match the controller to your motors. An important consideration in this selection is the support documents, libraries, and software examples. Features to look out for are inputs, outputs, servo control, and whether or not it can power the Pi from a single power source. The controller you select must be able to handle the battery voltage, the peak current of the motors when stalled, and be compatible with the Raspberry Pi.

MOTOR CONTROLLER SPECS:

Peak current: How much current the driver will tolerate before releasing magic smoke

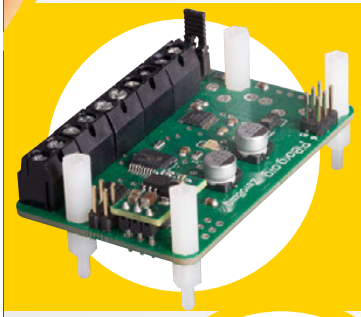
Drive voltage: Voltage used to drive the motor(s)

Drive current: Constant current the driver chip can maintain

Logic voltage: Voltage the logic must be driven at to function

Logic current: Current required to function

SOME CONTROLLERS TO CONSIDER:



ZEROBORG piborg.org/zeroborg

The MagPi PICK

The ZeroBorg is the controller we're using in this project. It has a number of features that make it stand out from some of the other boards, such as its use of two TI DVR 8833 H-bridges which allows for independent control of up to four motors. This makes it good for controlling an omnidirectional robot, or two stepper motors. You also can add/specify a DC-to-DC converter to power both the motors and the attached Raspberry Pi Zero. One final feature is the IR receiver, so you can control your robot with an IR remote control.

PICON ZERO magpi.cc/1p9wGaA

A well-thought-out Pi Zero format controller from 4tronix, it uses the same TI DRV8833 motor driver H-bridge chip as the ZeroBorg. The extra inputs and outputs are great for adding sensors, servos, and NeoPixels; there's even a dedicated socket for an HC-SR04 ultrasonic sensor! The power arrangements are very flexible, as you can drive the motors from the Pi's 5V rail or from a separate power source, from 3V to 11V. It's also worth checking out 4tronix's RoboHAT if you're using a full-sized Pi.

EXPLORER PHAT magpi.cc/1Pk5SdN

The Pimoroni Explorer pHAT was the first Pi Zero-format motor driver HAT, and is an excellent controller. It uses the TI DVR8833 H-bridge again and has four 5V-tolerant digital and analogue inputs, plus four 500mA outputs. The various inputs give options for connecting different types of sensors, and the motor driver will happily drive a pair of N20 metal gear motors. For a full-sized Pi, use the Explorer HAT Pro.

POWER YOUR ROBOT

The perfect batteries to get your robot moving

The correct battery can make a huge difference to your robot. It all comes down to four types for robots: lithium-ion, NiCad-based, lead acid, and dry cells. Battery technology has improved a great deal in recent years, thanks to the development of mobile phones, laptop computers, and tablets, with their requirement for high power and increased standby life. Whichever battery type you use, you'll need a battery holder to connect them.

NICAD / NIMH
(nominal cell voltage 1.2V)

NiCad / NiMH batteries were the number one choice before the rise of the lithium-ion batteries, due to their power-to-weight ratio and a predictable discharge voltage that changes little from 1.2V per cell until it nearly runs out. They're packaged in common battery sizes, including AAA, AA, and PP3. Chargers are also commonly available, even being sold in supermarkets. The discharge rates are not as high as a lithium battery, but they don't have a flammable metal in their construction. We're using this type for our robot.

The MagPi PICK

LITHIUM-ION INCLUDING LIPO
(nominal cell voltage 3.7V)

Lithium-ion-based batteries offer some of the highest energy density and energy release available. This means a robot powered in this fashion can use a smaller, lighter battery. Lithium batteries are more dangerous, though. There are two types of lithium batteries: type one has built-in safety circuits, to protect against under- and over-voltage and short circuit. Type two batteries have no safety circuits! If you wish to upgrade your robot to LiPo, make sure to be safe.

LEAD ACID
(nominal cell voltage 2V)

The granddaddy of all rechargeable batteries, the lead acid battery was invented in 1859 by Gaston Planté. This type of battery has a very low energy density and is made of lead. This makes it a poor choice for use in a robot, although it can supply high surge currents. While larger lead acid batteries are used in most cars, they're best ignored for this project due to their high weight and low energy density.

DRY CELL
(nominal cell voltage 1.5V)

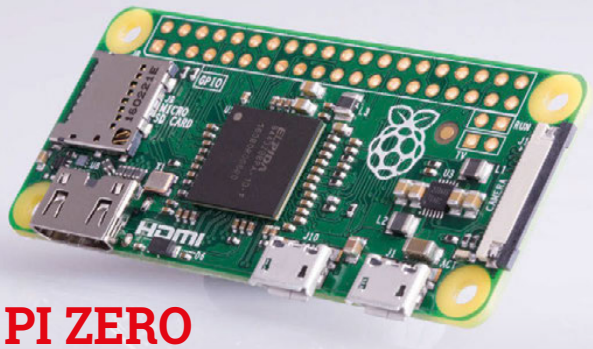
Zinc-carbon and alkaline are the more common types of dry cells, widely available in common battery sizes, including AAA, AA, and PP3. Although not rechargeable, they're useful as an emergency replacement for NiCad / NiMH; however, you need to be careful with the increased voltage. They're also expensive to continually replace.

CHOOSE A RASPBERRY PI

Finally, choose your computer

When it comes to choosing which Raspberry Pi to use in your robot, there are two Raspberry Pi models that are perfect for the role: the Pi Zero and the Raspberry Pi 3. The Model A is a close third due to its size and low power requirements, and the eventual Pi 3 Model A with on-board radio chip will make that an excellent choice in the future.

The Pi Zero is a great choice for a robot because of its even smaller size, along with its low power requirements. The Raspberry Pi 3 is the most powerful Pi to date, with 1.2GHz CPU, 1GB of memory, plus built-in wireless LAN and Bluetooth, leaving the four USB sockets free. The Raspberry Pi 3 makes sense for highly demanding applications like computer vision and multi-threading, but it will use up more power.

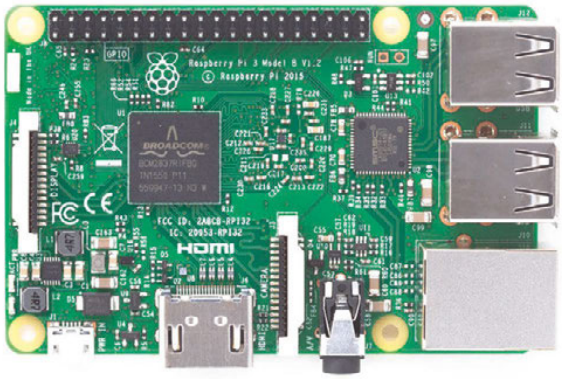


PI ZERO

PROS:
- ⊕ Very small
- ⊕ Very cheap
- ⊕ Low power consumption
- ⊕ More powerful than a Model A+

CONS:
- ⊖ Limited USB connectivity
- ⊖ Needs more soldering for GPIOs
- ⊖ No on-board wireless or Bluetooth

The MagPi PICK



RASPBERRY PI 3

PROS:
- ⊕ The most powerful Pi
- ⊕ Relatively low electricity requirement
- ⊕ 4× USB ports
- ⊕ Wireless LAN and Bluetooth

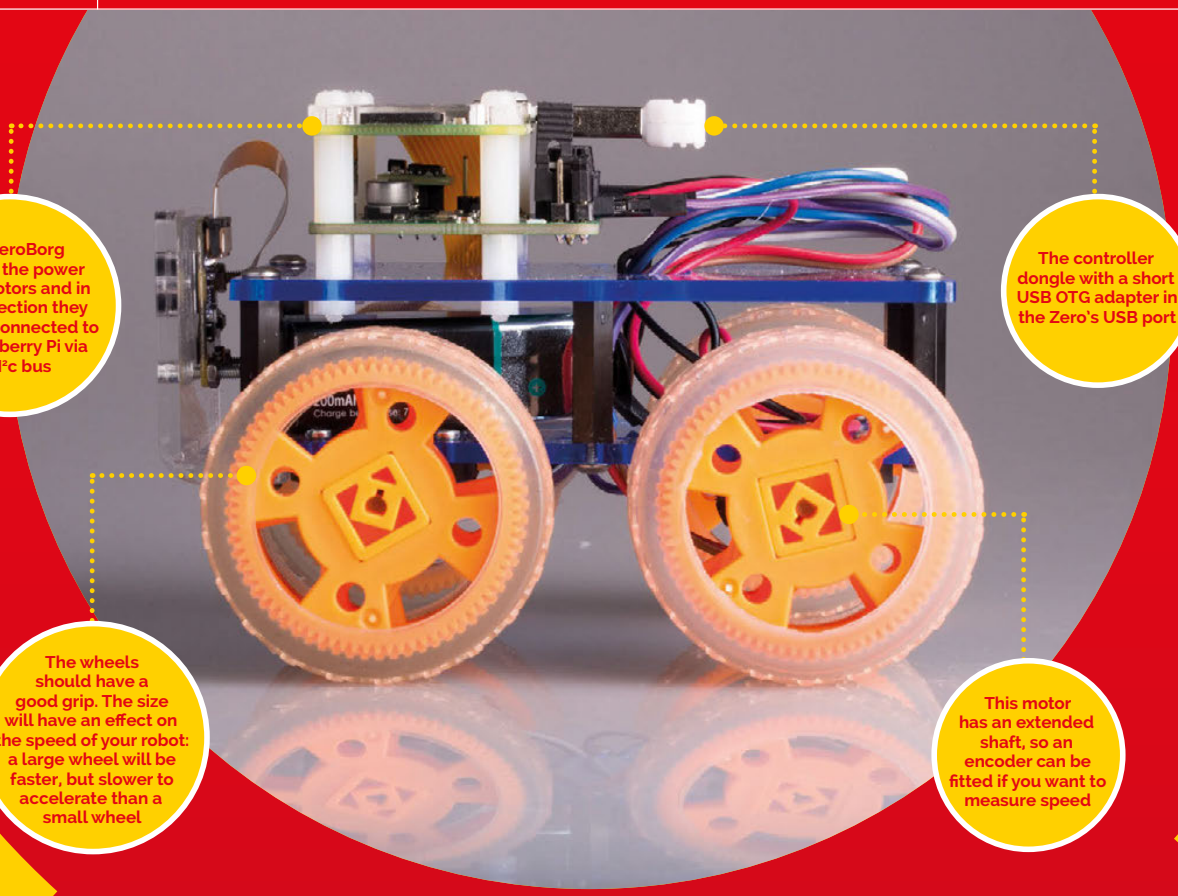
CONS:
- ⊖ Higher power consumption than other Pis
- ⊖ The biggest Pi
- ⊖ Most expensive Pi

MODEL A+

PROS:
- ⊕ Fairly small
- ⊕ Low power consumption
- ⊕ Standard-size USB port

CONS:
- ⊖ Bigger than Pi Zero
- ⊖ Only one USB port
- ⊖ No on-board wireless or Bluetooth
- ⊖ Least powerful Pi

NOW WE'RE READY TO **BUILD A ROBOT!**



DESIGNING THE CHASSIS

Now we have all the parts, they need to attach to something

The chassis for a robot needs to be strong and light, and have enough space to fit all the parts on it. For something like Pi Wars, there's an additional requirement for the robot's footprint to be less than 300 × 225mm.

Start by laying out the parts you already selected on a piece of paper the size of, or smaller than, the footprint you are building to. If you don't already have all the parts, you can model them in 3D or CAD software, or even cut them out of paper. This will give you an idea of what size your robot will need to be, and what clearance the wheels require. It also helps to think where and how additional sensors attach to your chassis.

In addition, you'll need to think about how to add and remove parts for different challenges. The distance between your robot's wheels will affect how your robot will turn/handle: if the wheel base is longer than its width, the robot will turn more slowly. This could be an advantage in a speed run challenge, as it would make it harder for the robot to turn and hopefully help keep it on track.

You also need to think about the clearance height from the surface your robot is standing on to the bottom of your motors/chassis. The higher it is, the better your robot will be at clearing obstacles. It's also a good idea to keep most of the weight low down in your design; this will stabilise your robot and make it less likely to topple over.

One of the most important things to think about is how easy will it be to change or charge your battery.

PRE-MADE ROBOT CHASSIS

Download our files so you can make the rover from our cover!

If you don't fancy designing a chassis just yet, we've got some files you can use. They're for the robot we've built; it's made up of two Perspex plates, with 3mm PCB spacers joining the two together. The motors and battery are fitted on the bottom plate, with the motor controller and Raspberry Pi mounted on the top plate. This creates a box, making a light and strong chassis with plenty of space for all the components, along with any sensors to be added in the future. The plates are laser-cut for this project, but they could also be 3D-printed or even cut and drilled by hand.

HOW TO MAKE THE CHASSIS

For the chassis plates, you'll have to get them laser-cut – or cut them out yourself – from 3mm plywood, MDF, or Perspex (acrylic). It's also possible to convert files for 3D printing.

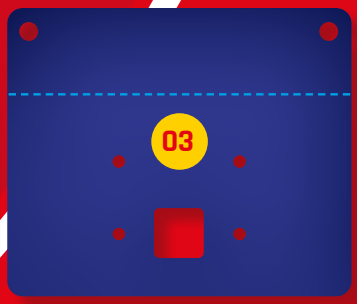
You can download a PDF for the plates from GitHub (**magpi.cc/2dx82h0**). You'll also find a DXF file, plus the original Inkscape SVG files, so you can modify the design if required. Your local makerspace, hackspace, or fab lab may be able to help cut the plates. In the UK, there's also Eagle Labs. There are other online laser cutting services – try searching for 'laser cutting services' in Google and look for local ones. A top tip to remember: the plates' edges should be cut last on the laser cutter.

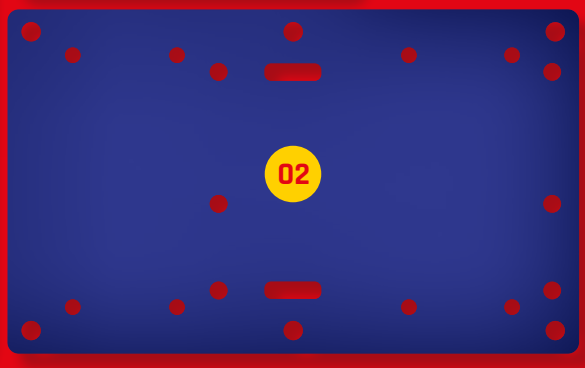
RESOURCES:

CHASSIS FILES
MagPi robot chassis files: **magpi.cc/2dx82h0**

PLACES TO BUILD THE CHASSIS
Hackspaces and makerspaces: **magpi.cc/2dxbnxr**
Eagle Labs: **labs.uk.barclays**
Fab Labs UK: **fablabsuk.co.uk**

ONLINE SERVICES
RazorLAB: **razorlab.co.uk**
Laser Make: **lasermake.co.uk**
Perspex (acrylic) supplier: **kitronik.co.uk**





01. BOTTOM PLATE
The bottom plate is the biggest part of the chassis – technically you don't need the top plate for this robot, but it's a lot more sturdier with it on

02. TOP PLATE
The Pi Zero and ZeroBorg are mounted on this. It also creates a little space between the chassis plates to store the battery

03. CAMERA MOUNT
The camera mount needs to be bent along the line going through it, so it can be mounted underneath the bottom plate while still having the camera face forward

BUILDING THE ROBOT

Get your screwdriver handy, as it's time to make your robot

PARTS LIST:

Major components:

Raspberry Pi Zero
(v1.3 if adding a camera)

PiBorg ZeroBorg complete
piborg.org/zeroborg

Chassis plates

4× 50:1 micro metal gear motors
magpi.cc/2eynuNk

4× Pimoroni motor brackets
magpi.cc/2dW6NYR

4× wheels
magpi.cc/2eq0Npp

USB to micro-USB OTG Converter Shim
magpi.cc/1JT9aZc

Wireless controller – we used the PDP Rock Candy
magpi.cc/2dvjKJS

Connectors and fasteners:

PP3 battery clip
magpi.cc/2ebjlgV

PP3 rechargeable battery

Female-to-male jumper wires
magpi.cc/2dvjMSb

6× 3mm hole, 20mm length PCB spacer posts
magpi.cc/2dvjKJR

12× 3mm, 8mm length hex pan head machine screws
magpi.cc/2ebkaWQ

8× straight header pins for motors
magpi.cc/2eeUbv9

Optional camera:

Camera holder

Raspberry Pi Camera Module

>STEP-01

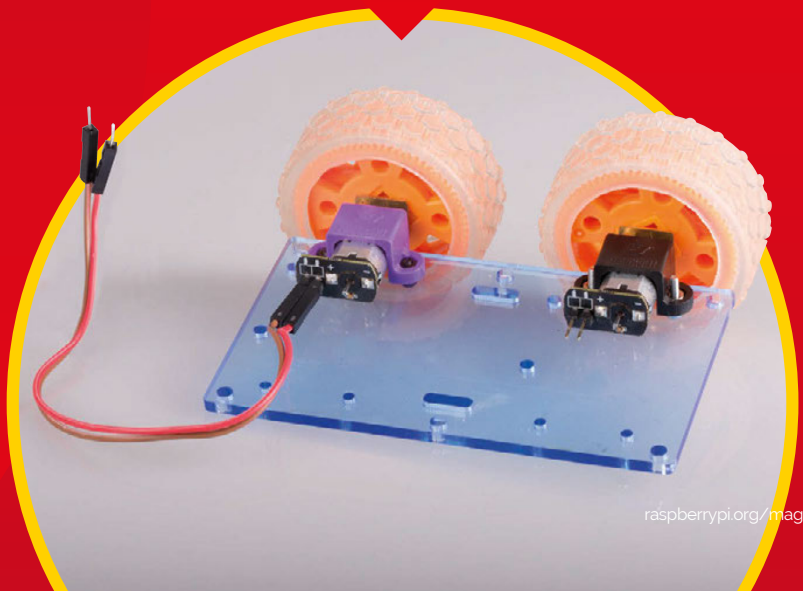
Prepare the motors

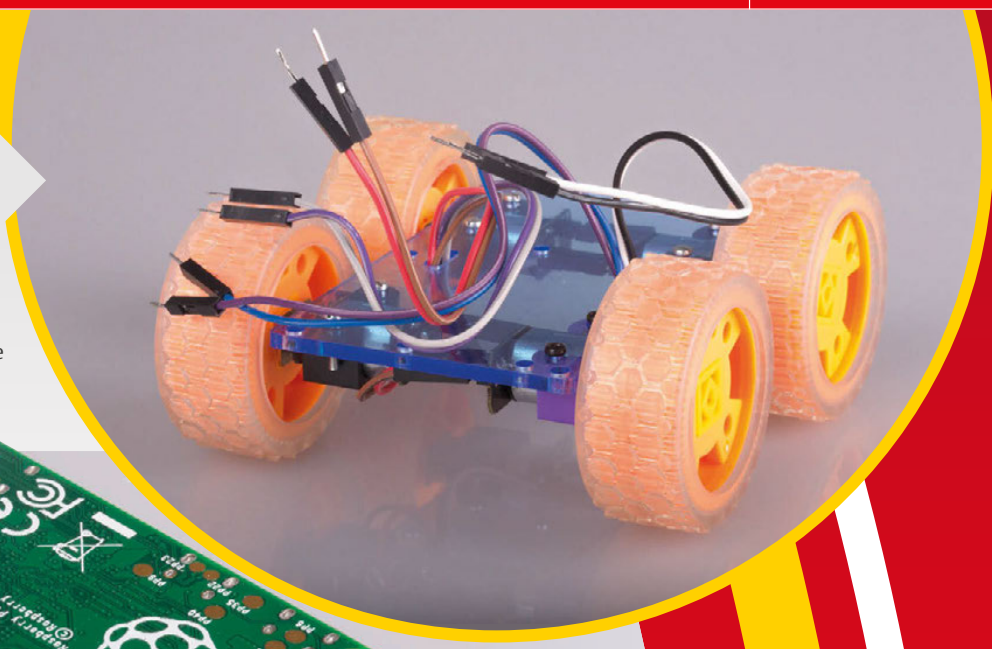
The motors must be modified slightly to make sure they fit under the chassis. Using a soldering iron and a pair of pliers, you should remove the bent pins from each motor shim, suck out the solder, and solder some straight pins back in. Once that's done, connect the wheels to the motors.

>STEP-02

Attach the motors

The motors need to be attached to the bottom plate using the brackets, as shown. You can either have the wires connected now while it's easier to do, or you can add them afterwards so they're out of the way as you connect the rest of the motors.

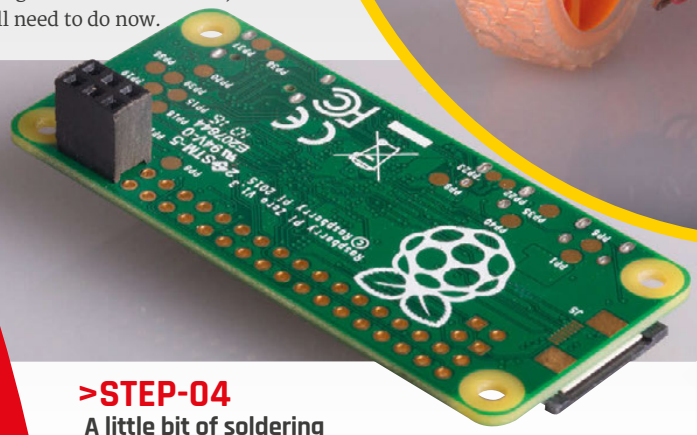




>STEP-03

Tidy the wires

Once all the motors are connected, flip the bottom plate the right way up and make sure all the wires are connected to the motors. You can push them two at a time through the rectangular holes on the side, which you'll need to do now.



>STEP-04

A little bit of soldering

Solder the header for the ZeroBorg onto the Pi Zero. It should be positioned on pins one through six, as shown above. Make sure it's soldered to the underside as in the picture as well!

>STEP-05

Make the brains

Attach the spacers to the ZeroBorg and then follow that up by placing the Pi Zero on top, making sure the header goes over the relevant pins. The USB ports and HDMI port are on the same side as the ZeroBorg connectors – refer to the step 06 image, just in case.

>STEP-06

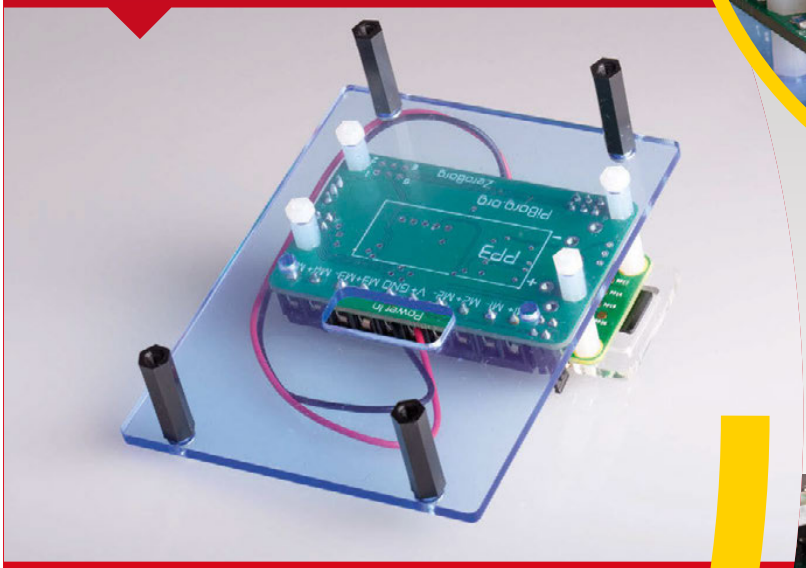
Construct the top part

You can attach the power wires now if you wish – refer to step 09 for the orientation – but either way, you need to then mount the ZeroBorg and Pi Zero combo to the top plate. It attaches to the shorter side; you'll know as it's the only spot you can slot in the screws.

>STEP-07

Complete the top

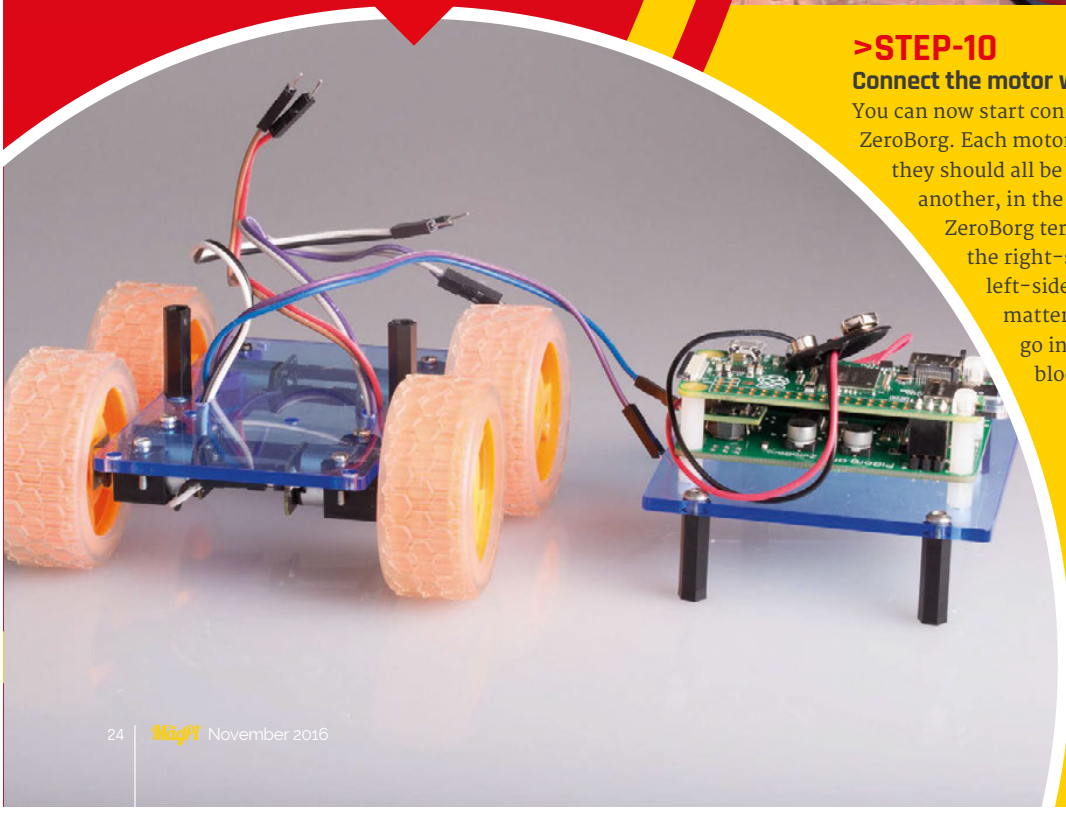
Once the Pi Zero combo is attached, attach four of the spacers for connecting the two plates onto the bottom of the top plate, as shown below.



>STEP-08

Build up the robot

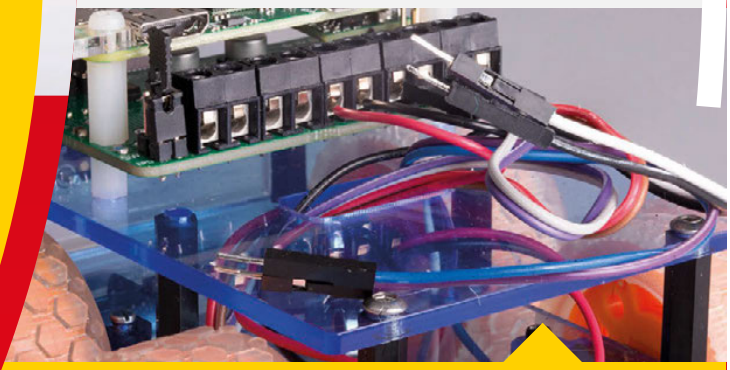
Attach the remaining two spacers onto the bottom plate; these can be secured to the top plate by removing the Pi Zero combo, but it's not necessary. Place the top plate onto the chassis and secure the top plate's spacers to the bottom plate.



>STEP-09

Check the battery wires

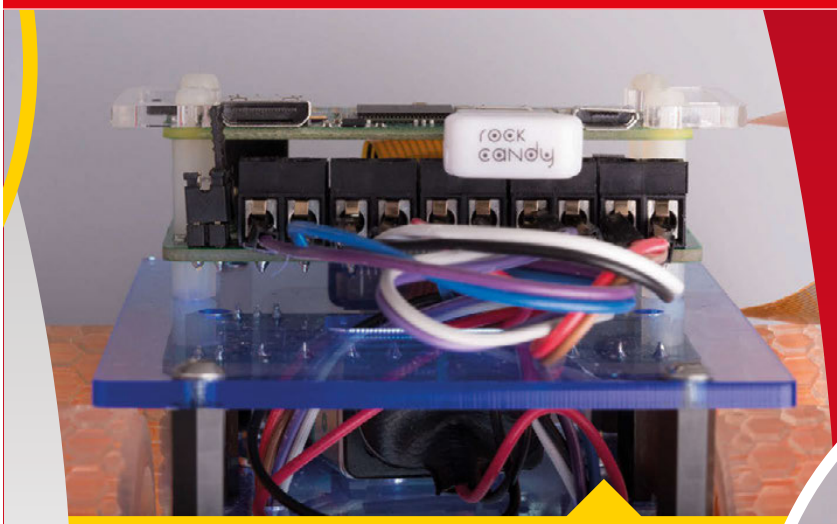
If you haven't already, connect the power wires to the central terminals, as shown. It's important to do so in this order (positive on left, negative on right), otherwise you won't be able to properly power the ZeroBorg, which in turn powers everything else.



>STEP-10

Connect the motor wires

You can now start connecting the motors to the ZeroBorg. Each motor has a pair of cables, and they should all be connected in pairs, one after another, in the remaining four pairs of ZeroBorg terminals. It's a good idea to keep the right-side motors on the right and left-side on the left as well. It doesn't matter which way around the pairs go in their individual terminal blocks, though.



>STEP-11

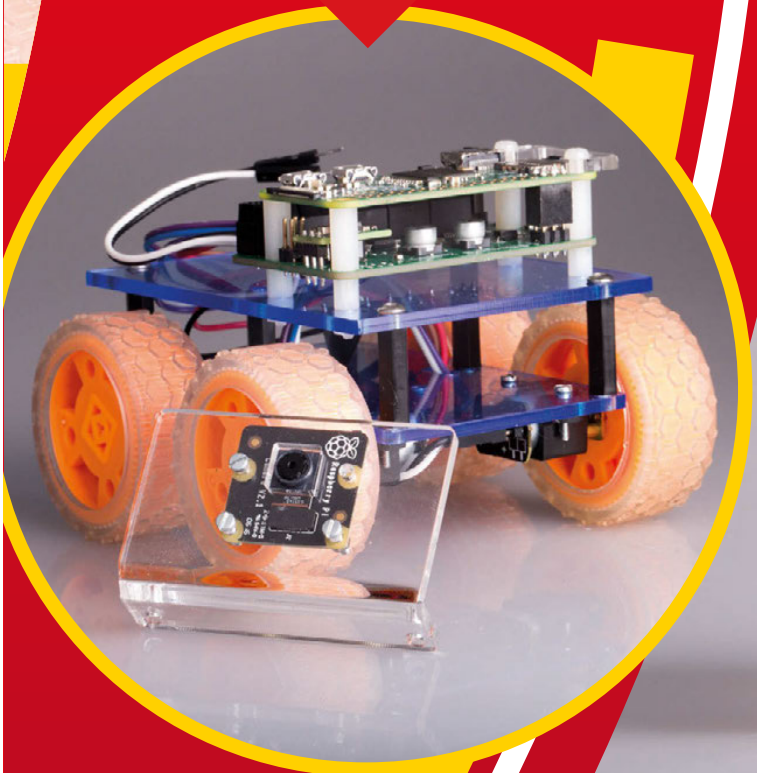
Tidy the robot up

Tidy up the wires and connect the battery; it won't turn the robot on until you move the jumper to the on position. Connect the wireless controller's dongle and you're ready to get programming, if all you want is a remote-controlled robot.

>STEP-12

Mount the camera

To add the camera, screw the Camera Module to the extra camera plate. This can then be mounted to the front of the bottom plate, removing the screws that secure the spacers and using them to add the camera.



>STEP-13

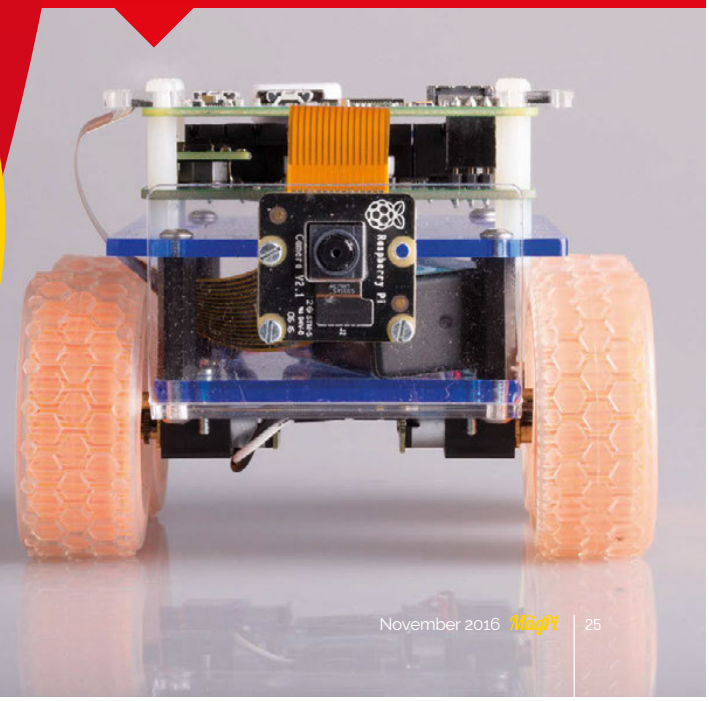
Connect the camera cable

The last thing to do is add the camera connector; it needs to go in both the Pi Zero (v1.3) and the Camera Module. Make sure the white side of the cable is facing up when you connect it to the Pi Zero, while the silver side should be facing the Pi Camera when it's inserted in that end.

>STEP-14

Robot complete!

You're finished! Now, with a bit of coding, you can get your robot working. We'll get started over the page...



PROGRAM YOUR ROBOT

It has a body. Now it's time give it a brain!

As we're using the ZeroBorg, we will use the library that comes with it to program the robot. We'll need to modify the code slightly to get it working for our needs, though. It's best to start with a fresh install of Raspbian as well – you can do the setup on another Raspberry Pi if you wish.

Start by downloading the latest edition of Raspbian or NOOBS and copying it to a microSD card. There's a quick-start guide to doing so on the Raspberry Pi website (**magpi.cc/2eopaEf**), so you can either follow the instructions given there or read our guide from issue 50 (**magpi.cc/Issue-50**).

>STEP-01

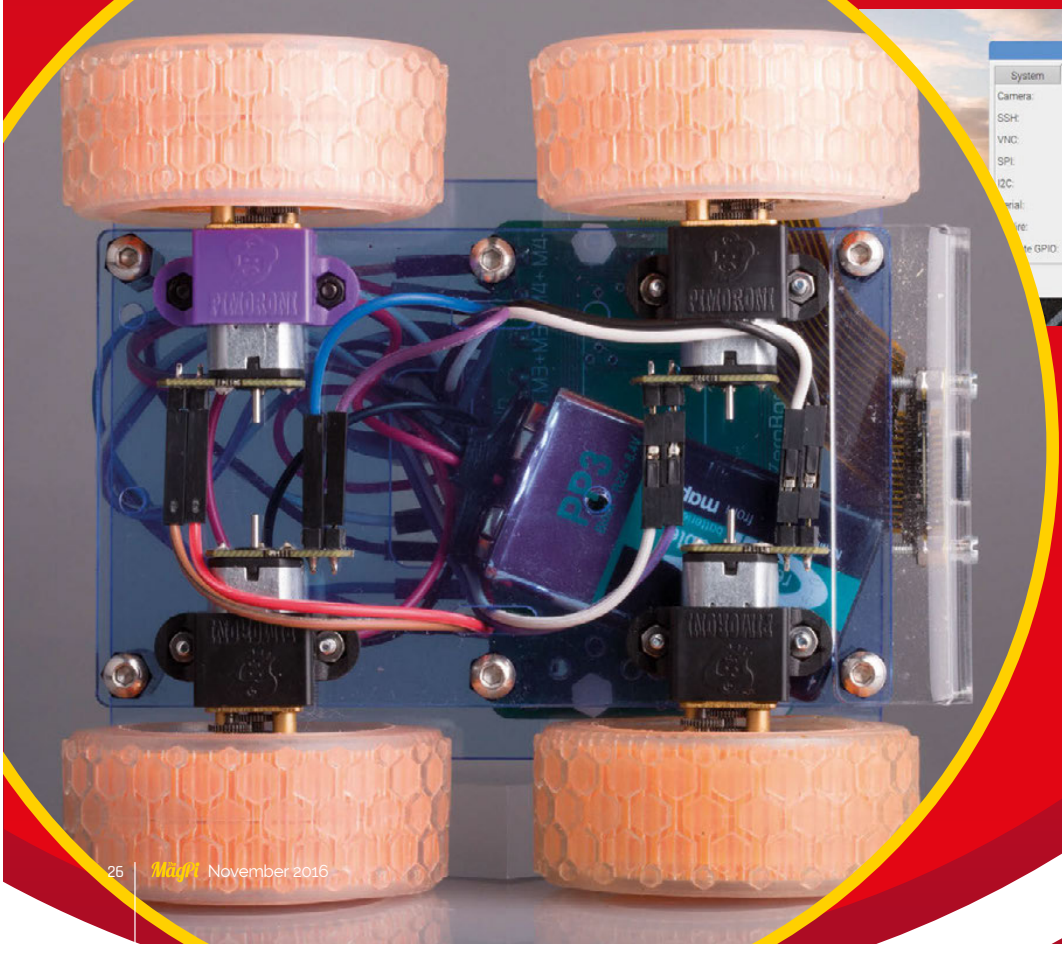
Update Raspbian

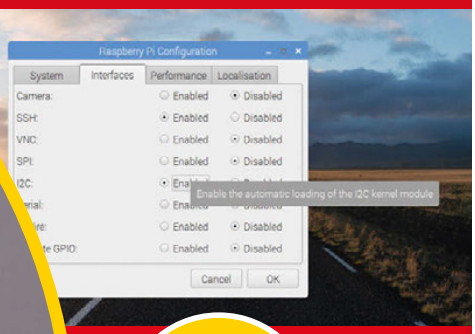
Once installed, connect your Raspberry Pi to the internet and update it in a terminal window or the command line with:

```
sudo apt-get update
sudo apt-get upgrade
```

Next, enable the I²C bus. You will need to do this via the Raspberry Pi Configuration menu. Click the Menu button at the top left of the Raspbian desktop, select Preferences, and finally, click Raspberry Pi Configuration. Find the Interfaces tab and then click the Enable button for I2C. Click the OK button to save and exit, then restart your Raspberry Pi.

It's best to test the motors with the robot upside down or in the air so it doesn't go running off





Enabling I²C here is extremely important for making sure the robot works

>STEP-02
Install ZeroBorg software

Now you need to download the ZeroBorg software. Open a terminal window and enter the following:

```
bash <(curl https://www.piborg.org/
install-zeroborg.txt)
```

You should also install the joystick app for testing your joypad and mapping the buttons. See the 'Using a gamepad' section (bottom right) for more details. The button mapping for the game controller is in **zbJoystick.py**. Use nano to open the file, and find the line `# Settings for the joystick`. For the Rock Candy controllers used in our version, it should look something like this:

```
axisUpDown = 1
axisUpDownInverted = False
axisLeftRight = 2
axisLeftRightInverted = False
buttonResetEpo = 9
buttonSlow = 6
slowFactor = 0.5
buttonFastTurn = 7
interval = 0.00
```

>STEP-03
Auto-start on boot

The final step is for the software to auto-start when you power on your robot. On the command-line or in a terminal window, use:

```
sudo nano /etc/rc.local
```

This will open the nano text editor; after the line starting with `fi`, enter the following:

```
./home/pi/zeroborg/runJoystick.sh &
```

Save and exit. You can now test your robot! If you have not done so already, put the microSD card into the Raspberry Pi Zero, then power up your robot by moving the jumper on the ZeroBorg to cover both the power pins. Test the motors by pushing forward on the wireless controller's left stick; this should result in all four wheels moving in the same direction. If some wheels are going in a different direction, swap around their cables in the ZeroBorg for that specific motor. Next, push the stick to the left and right, making sure that the correct wheels run when selected – if they do not, swap the incorrect pairs of jumpers.

CONTROL FROM A WEB BROWSER

Control your robot from your computer or smartphone and use the on board camera

For this way of controlling the robot, you'll have to switch out any game controller receiver for a WiFi adapter. Plug the Pi Zero into a computer and get the WiFi connected before continuing. Once that's done, you need to find the IP address of the robot by opening a terminal window and using:

```
ifconfig
```

Make a note of it – this is how we'll connect to the robot from elsewhere. Then install OpenCV using:

```
sudo app-get install python-opencv
```

Once that's done, you'll need to download the PiBorg web interface. Back in the terminal window, use:

```
git clone https://github.com/piborg/diddyborg-web.git
```

This will create a folder called **diddyborg-web**. Enter the following commands to start the web server:

```
cd diddyborg-web
sudo python diddyborg-webyetiWeb.py
```

It will confirm when the web server has started. Type the IP address you found earlier into a browser on your PC or smartphone (or even another Raspberry Pi!) and you'll be able to connect to the robot and see through the camera. Disconnect the monitor and any peripherals, then go for a ride!

USING A GAMEPAD

To get the gamepad or joystick button numbers, install the joystick software with:

```
sudo apt-get install joystick
```

Plug your gamepad in and then run the following to find out how it's listed:

```
ls /dev/input/js*
```

It will probably come up as something like /dev/input/js0. You can then test it using:

```
jstest /dev/input/js0
```

Move each stick and press each button in turn. You will see the values change for the axis or button pressed in real-time. Make some notes so you remember which button is which!

NEXT ISSUE: THE CHALLENGES

Come back next issue to learn how to add and use sensors to your robot to make it a true automaton

Pi Wars is next April and has a load of different challenges; it's a bit too late to enter them, but that doesn't stop you from learning how to get your robot competition-ready by using some excellent robo-sensors. Come back next month for our guide on how to add and use amazing sensors to conquer these challenges.

STRAIGHT-LINE SPEED TEST

The Straight-Line Speed Test is autonomous, with only a start and stop button allowed for the starting and stopping of your robot's run. The course is a straight run, just over 7.3m long, with walls 67mm high placed 522mm apart – very precisely. There are penalties if your robot touches a wall, plus points for each clean and completed run. Three runs must be attempted. Here are some methods you could use for this challenge...

Computer vision: Find the walls and guide your robot down the centre. You could also use an IR light beacon placed at the end of the course and aim to keep the beacon in the centre of the image, while steering the robot down the centre of the course.

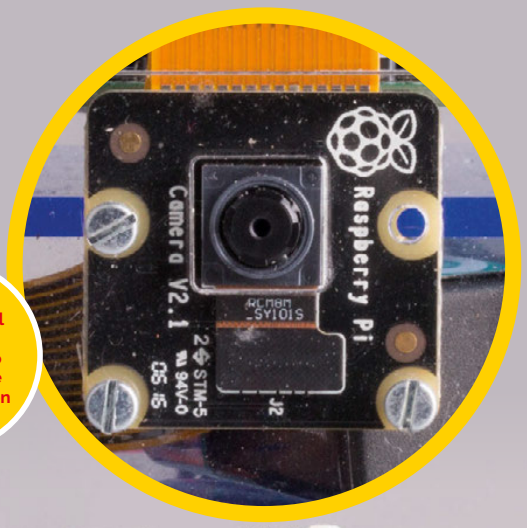
Dead reckoning using wheel/motor encoders: Count the number of turns each wheel makes in a set time. The shorter the time, the more accurate it should be.

Distance sensors: Measuring the distance between the walls, steer the robot down the centre of the course.

IMU: Set a bearing and follow it – distance sensors may be useful as a backup check.

This ultrasonic distance sensor can be used to check how far away the wall is

MINIMAL MAZE



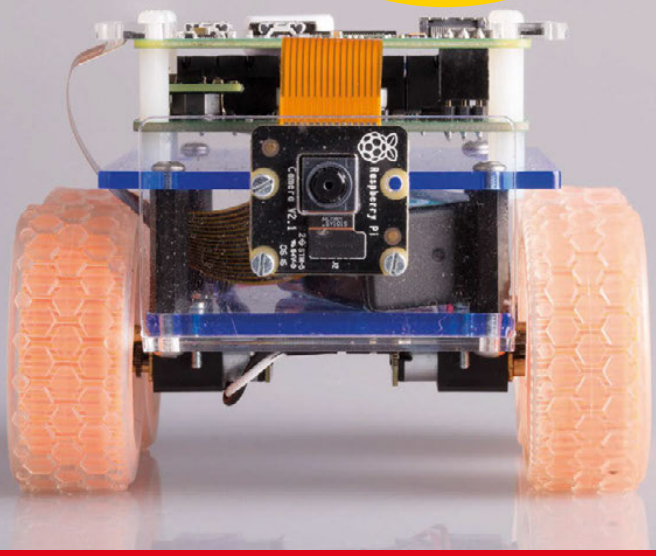
Check the wall colours with the camera to determine the robot's direction

The Minimal Maze is new to the Pi Wars contest, and it is an autonomous event. It's basically two right turns followed by two left turns in a course with 65mm-high walls. Additional points will be given for clean runs where the robot does not touch the walls, along with points for a completed run.

Penalties are given for touching walls, rescue attempts, and non-completed runs. As the measurements for the maze are not given, it will not be possible to use dead reckoning to navigate the maze, so the challenge requires the use of sensors. Not counting access to a lidar (a laser radar!), we have a number of sensors at our disposal.

You could use the Pi Camera Module and OpenCV to find the positions of the walls or determine their colour. The walls are colour-coded to their orientation, so you will be able to tell which way your robot is facing.

It may be a good idea to use distance sensors to avoid hitting the walls. Another method would to drive forward until you are a set distance away from the wall, then turn 90 degrees, either clockwise or anticlockwise depending on whether you need to turn left or right, repeat until last turn, then drive forward to the exit. The use of wheel/motor encoders and an IMU may help. If you use the data from your first run, it may be possible to improve your next run's time.



Line followers come in many varieties: choose the best one for the size and shape of your robot

LINE FOLLOWING

The classic line-following challenge returns to Pi Wars. This time, organisers Michael and Tim have promised a return to the black line on a white background for the course. You have a number of options, including building your own line-following sensor, or buying one of the many available sensor arrays. A popular choice is Ryanteck's three-way line follower (**magpi.cc/2eoEXTl**), as used by the winning robot for the line-following challenge in 2015. Other sensors are available from Pololu and Sparkfun. The line-following sensor used by Revenge, the second-place robot in the last contest, was a Pololu QTR interfaced with an Arduino.