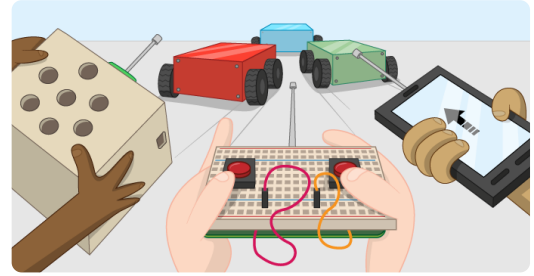




Remotely control your buggy

Create a remote control for your Raspberry Pi buggy



Step 1 Introduction

This project assumes that you've already built a motorised buggy controlled by a Raspberry Pi, such as the one in the **Build a buggy resource** (<https://projects.raspberrypi.org/en/projects/build-a-buggy>).



With your buggy built, it is now time to add in mechanisms to allow you to remotely control the buggy using one of the following devices:

1. An Android phone (or tablet)
2. A Google AIY Projects Voice Kit
3. A home-brew remote control

What you will learn

- How to receive input from an Android device on your Raspberry Pi
- How to use GPIO pins remotely to control a robot buggy
- How to process data from an AIY Projects Voice Kit to control a robot buggy

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (<https://curriculum.raspberrypi.org/>):

- **Combine programming constructs to solve a problem** (<https://curriculum.raspberrypi.org/programming/builder/>)
- **Combine inputs and/or outputs to create projects or solve a problem** (<https://curriculum.raspberrypi.org/physical-computing/builder/>)
- **Use basic materials and tools to create project prototypes/** (<https://curriculum.raspberrypi.org/manufacture/creator/>)

Step 2 What you will need

The hardware and software used in this resource are specific to the type of remote control you want to make. Have a look at the individual steps to see the requirements.

Hardware

Regardless of which remote you wish to make, you will need to have created a Pi-controlled robot buggy using the **Build a buggy resource** (<https://projects.raspberrypi.org/en/projects/build-a-buggy>).

Step 3 An Android remote control

Requirements

For this stage, you will need an Android phone or tablet as well as your buggy.

Instructions

You can use the Android app **Blue Dot** as a remote control for your robot buggy.

You'll need to do a little setting up first:

Download the **Blue Dot Android app** from here (https://play.google.com/store/apps/details?id=com.stuffaboutcode.bluedot&hl=en_GB) and install it.



On your Raspberry Pi, open a terminal window and install the **dbus** and **bluedot** Python modules.



```
sudo pip3 install bluedot
```

Have a look at the section below to learn the basics of using the Blue Dot app with your Raspberry Pi.

To remotely control your buggy, here's what you will need to do:

Open up **mu** from the **Programming** menu.



Import the **bluedot** and **gpiozero** modules, and create **Robot** and **BlueDot** objects.



```
from bluedot import BlueDot
from gpiozero import Robot

bd = BlueDot()
robot = Robot(left=(7, 8), right=(9, 10)) ##this may be different depending on your wiring
```

Create a function called **move** that has **pos** as a parameter.



```
def move(pos):
```

The function should check if **pos.top** is **True**. If it is, then **robot.forward()** can be used to move the robot forward.



The function should also check `pos.bottom`, `pos.right`, and `pos.left`, and move the `robot` accordingly.



Create a function called `stop` that stops the robot.



- When the blue dot is pressed or the finger moves, the `move` function should be called.
- When the blue dot is released, the `stop` function should be called.



Here's a video taking you through the code writing process:

Step 4 AIY Projects Voice Kit remote

Requirements

For this part of the project, you will need a Google AIY Projects Voice Kit and a second Raspberry Pi as well as your buggy.

Instructions

You can use a second Raspberry Pi running the Google AIY Projects Voice Kit software to remotely control your Raspberry Pi buggy. This section assumes you have both a **built buggy** (<https://projects.raspberrypi.org/en/projects/build-a-buggy>), and that you have an assembled **AIY Projects Voice Kit** (<https://projects.raspberrypi.org/en/projects/rpi-aiy-voice-assemble>).

To control the buggy, you can use a feature in the `gpiozero` module called **remote pins**. Have a look at the section below to learn more about how to remotely use GPIO pins.

Connect up your buggy and then open a terminal window on its Raspberry Pi. Once there, you can type the following so that the **pigpio daemon** will run on boot:



```
sudo systemctl enable pigpiod
```

Next, type in `hostname -I` to show the Raspberry Pi's IP address. Make a note of it, as you will need it in a bit.



You can now leave the buggy, as all the other code you will be writing will be for the AIY kit.

Import the modules and set up remotely controlled pins

Open the `action.py` file and, near the top, add code to import the necessary modules to remotely control the robot.



```
from gpiozero import Robot
from gpiozero.pins.pigpio import PiGPIOFactory
```

- Now you can set up the remote pins using the IP address of your buggy.

```
factory = PiGPIOFactory(host="192.168.1.79")
```

- Then you can set up your robot with these pins.

```
robot = Robot(left=(7, 8), right=(9, 10), pin_factory=factory)
```

Creating a voice command

Scroll to near the bottom of the `action.py` file and find the section with the following comments:



```
# =====
# Makers! Add your own voice commands here.
# =====
```

Here is where you can write your custom command that will activate your robot. This might require some experimenting. Using the command 'robot' may not be a good idea, as the Google API might struggle to recognise the word. Play around with different command words to see which one works best.

```
actor.add_keyword("red leader", ControlRobot(say))
```

Create your custom action

- Scroll back up in the file until you find the section with the following comments:



```
# =====
# Makers! Implement your own actions here.
# =====
```

- Start by creating the class and initialising it with the `say` function.

```
class ControlRobot():
    def __init__(self, say):
        self.say = say
```

- Now you need a `run` method. This is where the real work is done.

```
class ControlRobot():
    def __init__(self, say):
        self.say = say

    def run(self, voice_command):
```

- If you followed our **AIY Projects Voice Kit resource** (<https://projects.raspberrypi.org/en/projects/google-voice-aiy>), then this should be fairly familiar to you. Within the `run` method you need to do the following:
 - Convert the `voice_command` to lower case
 - If the word `forward` is in the voice command, then send the robot forward
 - If the word `backward` is in the voice command, then send the robot backwards

You can continue this logic to finish the class. Have a look at the hints below if you need a little bit of help.

- Here's a video detailing how to code the Voice Kit, along with explanatory commentary:


Step 5 A home-brew game controller

Requirements

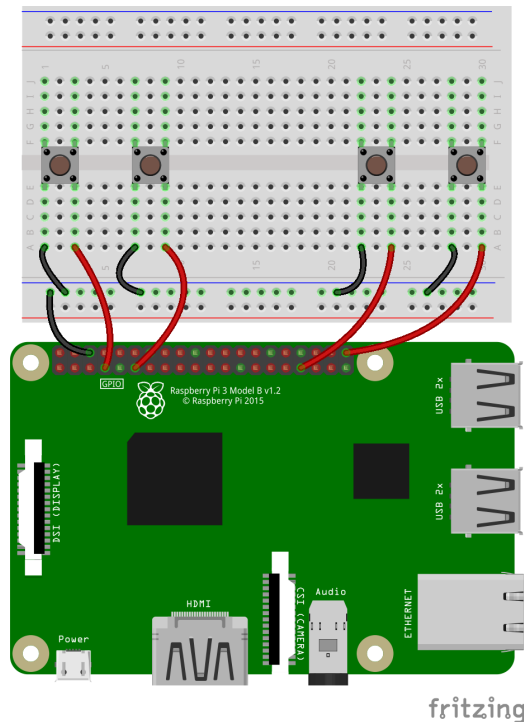
For this stage, you will need a second Raspberry Pi and the following hardware as well as your buggy.


- 1 × solderless breadboard
- 4 × tactile buttons
- 5 × male-to-female jumper leads
- 4 × male-to-male jumper leads

Instructions

The first stage is to wire up four buttons to your spare Raspberry Pi. Have a look at the section below if you haven't used buttons with a Raspberry Pi before. 

Here's one way that you could wire up your buttons, including a common ground rail to avoid using too many wires:



On the Raspberry Pi that is attached to the buggy, you will need to start the **pigpio daemon** if you haven't already done so. To do this, open up a terminal window and type the following: 

```
sudo pigpiod
```

If you want the **pigpio daemon** to start automatically at boot also run the following:

```
sudo systemctl enable pigpiod
```

You can then find the buggy's IP address using this line of code:

```
hostname -I
```

```
pi@robby:~$ hostname -I
192.168.1.113 fd00::1:5e0b:2045:b6a:cf87 Fdf4:9ff3:e432:9500:ee06:cf2:af97:c3b
pi@robby:~$
```

Which you will need later to connect to your buggy.

If you haven't used remotely controlled GPIO pins, or programmed using buttons before, you might like to have a look at the sections below.

Now you can begin to program the Raspberry Pi that is connected to the breadboard controller.

Open up **mu** from the **Programming** menu.

Begin by importing the modules you need, and setting up the remote pins for the robot:

```
from gpiozero import Robot, Button
from gpiozero.pins.pigpio import PiGPIOFactory

#                               use your Buggy's IP here
robot_pins = PiGPIOFactory(host="192.168.1.79")
robot = Robot(left=(7,8), right=(9,10), pin_factory=robot_pins)
forward = Button(4)
backward = Button(17)
right = Button(13)
left = Button(21)
```

For the rest of your project you need to add code to your script to drive the robot in the right direction when a specific button is pushed, and stop the robot when the button is released. Have a look at the hints below if you need a little help.

Here's a video to help you understand how to program the controller.

Published by **Raspberry Pi Foundation** (<https://www.raspberrypi.org>) under a **Creative Commons** license (<https://creativecommons.org/licenses/by-sa/4.0/>).

View project & license on GitHub (<https://github.com/RaspberryPiLearning/remote-control-buggy>).