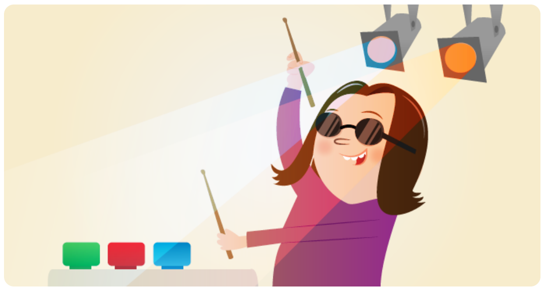
Projects

GPIO music box

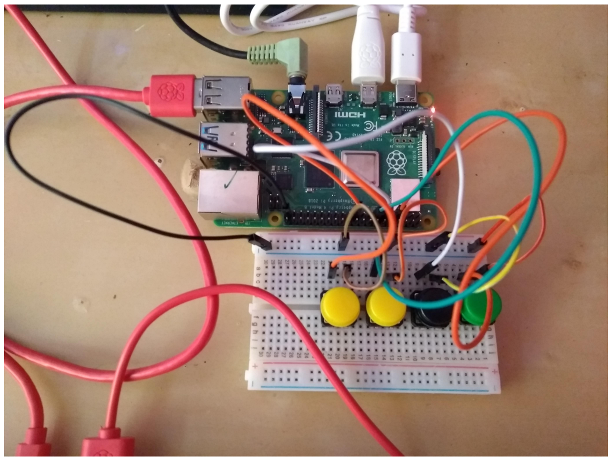
Make a device that plays music when you press its buttons



Step 1 Introduction

In this project, you will build a button-controlled “music box” that plays different sounds when different buttons are pressed. You can find a shortened version of this project on **YouTube** (https://www.youtube.com/watch?v=2izvSzQWYak&feature=youtu.be)

What you will make



What you will learn

- Play sounds in Python with `pygame`
- Use the Python `gpiozero` library to connect button presses to function calls
- Use the dictionary data structure in Python

What you will need

Hardware

- A Raspberry Pi computer
- A breadboard
- Four (4) tactile switches (to make buttons)
- Five (5) male-to-female jumper leads
- Four (4) male-to-male jumper leads
- Speakers or headphones

Software

- The latest version of the **Raspbian** (https://www.raspberrypi.org/downloads/raspbian/) operating system

Additional information for educators

If you need to print this project, please use the **printer-friendly version** (https://projects.raspberrypi.org/en/projects/gpio-music-box/print).

You can **find the solution for this project here** (http://rpf.io/p/en/gpio-music-box-get).

Step 2 Set up your project

You will need some sample sounds for this project. There are lots of sound files on Raspbian, but it can be a bit difficult to play them using Python. However, you can convert the sound files to a different file format that you can use in Python more easily.

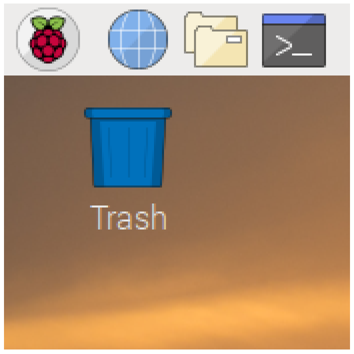
First, in your `home` directory, create a directory called `gpio-music-box`. You will use the new directory to store all your files for the project.

Copy the sample sounds

Use the same method as before to create a new directory called `samples` in your `gpio-music-box` directory.

There are lots of sample sounds stored in `/usr/share/sonic-pi/samples`. In the next step, you will copy these sounds into the `gpio-music-box/samples` directory.

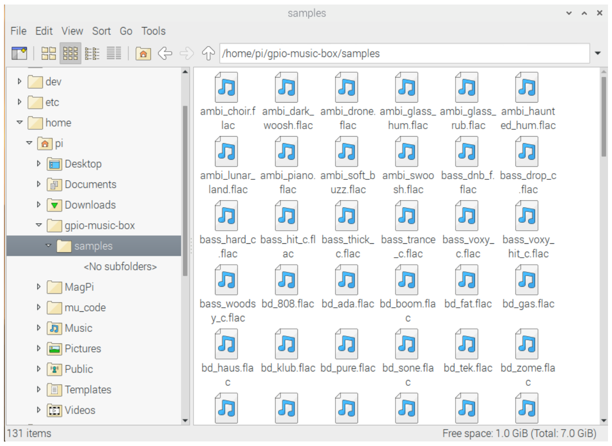
Click on the icon in the top left-hand corner of your screen to open a terminal window.



Type the following lines to copy all the files from one directory to the other:

```
cp /usr/share/sonic-pi/samples/* ~/gpio-music-box/samples/.
```

When you have done that, you should be able to see all the `.flac` sound files in the `samples` directory.



Convert the sound files

To play the sound files using Python, you need to convert the files from `.flac` files to `.wav` files.

In a terminal, change into your `samples` directory.

```
cd ~/gpio-music-box/samples
```

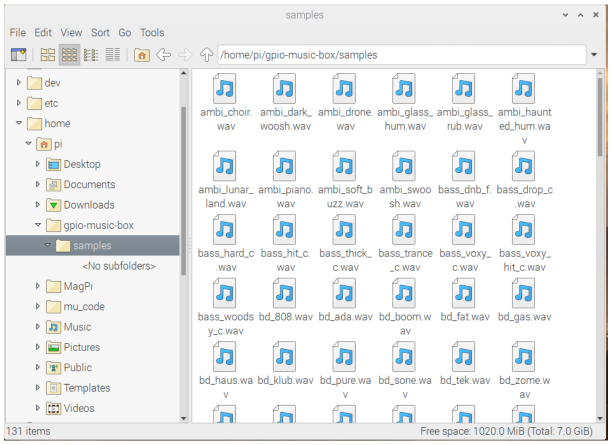
If you want to learn more about converting media files and running commands on multiple files, look at the two sections below.

In your terminal, type the following commands. This will convert all the `.flac` files to `.wav` files, then delete the old files.

```
for f in *.flac; do ffmpeg -i "$f" "${f%.flac}.wav"; done
rm *.flac
```

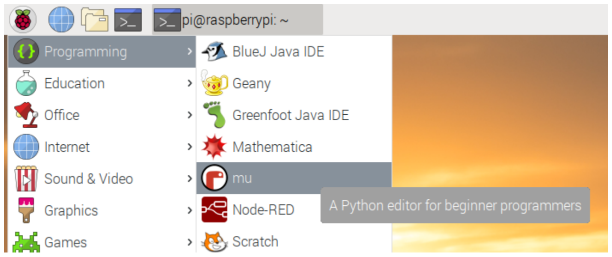
It will take a minute or two, depending on the Raspberry Pi model that you are using.

You should now be able to see all the new `.wav` files in the `samples` directory.



Step 3 Play sounds

Next, you will start to write your Python code. You can use any text editor or IDE to do this — Mu is always a good choice.



To start to create the instruments of your music box, you need to test whether Python can play some of the samples that you have copied.

First, import and initialise the `pygame` module for playing sound files.

```
import pygame

pygame.init()
```

Save this file in your `gpio-music-box` directory.

Choose four sound files that you want to use for your project, for example:

```
drum_tom_mid_hard.wav
drum_cymbal_hard.wav
drum_snare_hard.wav
drum_cowbell.wav
```

Then, create a Python object that links to one of these sound files. Give the file its own unique name. For example:

```
drum = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_tom_mid_hard.wav")
```

Create named objects for your remaining three sounds.

Here's what your code should look like:

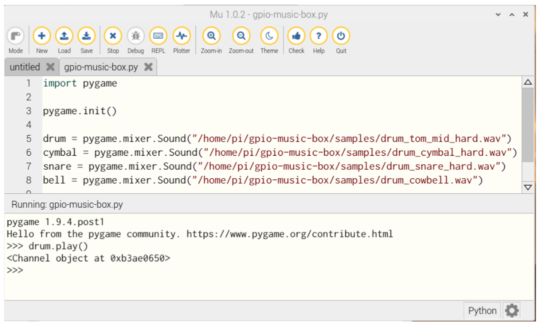
```python
import pygame

pygame.init()

drum = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_tom_mid_hard.wav")
cymbal = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cymbal_hard.wav")
snare = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_snare_hard.wav")
bell = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cowbell.wav")
```

Save and run your code. Then, in the shell at the bottom of the Mu editor, use `.play()` commands to play the sounds.

```
drum.play()
```



```python
import pygame

pygame.init()

drum = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_tom_mid_hard.wav")
cymbal = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cymbal_hard.wav")
snare = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_snare_hard.wav")
bell = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cowbell.wav")
```

```
pygame 1.9.4.post1
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> drum.play()
<Channel object at 0xb3ae0650>
>>>
```

If you don't hear any sound, check that your speakers or headphones are working and that the volume is turned up.

Step 4 Connect your buttons

You will need four buttons, each wired to separate GPIO pins on the Raspberry Pi.

- Place the four buttons into your breadboard. ✓

- Wire each button to a different numbered GPIO pin. You can choose any pins you like, but you will need to remember the numbers. ✓

Here’s a video that shows how you can wire the buttons.

Step 5 Play sounds at the press of a button

To see how a function can be called using a button press, have a look at the section below.

When the button is pressed, the program should call a function such as `drum.play()`.

However, when you use an event (such as a button press) to call a function, you don't use brackets `()`.

This is because the program must only call the function when the button is pressed, rather than straight away. So, in this case, you just use `drum.play`.

First, set up one of your buttons. Remember to use the numbers for the GPIO pins that **you** have used, rather than the numbers in the example.

```
import pygame
from gpiozero import Button

pygame.init()

drum = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_tom_mid_hard.wav")
cymbal = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cymbal_hard.wav")
snare = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_snare_hard.wav")
bell = pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cowbell.wav")

btn_drum = Button(4)
```

To play the sound when the button is pressed, just add this line of code to the bottom of your file:

```
btn_drum.when_pressed = drum.play
```

Run the program and press the button. If you don't hear the sound playing, then check the wiring of your button.

Now, add code to make the remaining three buttons play their sounds.

Here's an example of the code that you could use for a second button.

```
btn_cymbal = Button(17)

btn_cymbal.when_pressed = cymbal.play
```

Step 6 Improve your script

The code that you have written should work without any problems. However, it’s generally a good idea to make your code a bit cleaner once you have a prototype that works.

The next steps are completely optional. If you’re happy with your script, then just leave it as it is. If you want to make your script a bit cleaner, then follow the steps on this page.

You can store your button objects and sounds in a dictionary, instead of having to create eight different objects.

Have a look at the steps below to learn about creating basic dictionaries and then looping over them.

First, create a dictionary that uses the `Button`s as keys and the `Sound`s as values.

```
button_sounds = {Button(4): pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_tom_mid_hard.wav"),
                 Button(17): pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cymbal_hard.wav"),
                 Button(27): pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_snare_hard.wav"),
                 Button(10): pygame.mixer.Sound("/home/pi/gpio-music-box/samples/drum_cowbell.wav")}
```

You can now loop over the dictionary to tell the program to play the sound when the button is pressed:

```
for button, sound in button_sounds.items():
    button.when_pressed = sound.play
```

Challenge: a mobile disco

- Can you use different sounds in your program?
- Can you add more buttons for a greater variety of instruments?
- Can you add some LEDs to your project and make them blink whenever a sound is played?

Step 7 What next?

Try a few more physical computing projects:

- **Build a robot buggy** (https://projects.raspberrypi.org/en/projects/build-a-buggy)
- **See like a bat** (https://projects.raspberrypi.org/en/projects/see-like-a-bat)
- **Whoopi cushion** (https://projects.raspberrypi.org/en/projects/whoopi-cushion)