

Build A Raspberry Pi Space Invaders Controller



Tutorial by Stuart Fox

Thanks to Scott Bowman & Lee Robinson

Public Domain

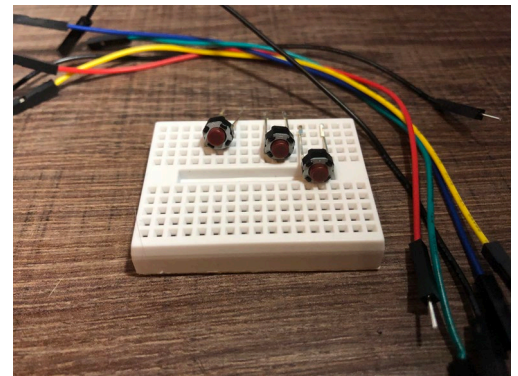
28th Jan 2017 - www.cotswoldjam.org

In this tutorial, we'll build a mini game controller, wire it to a Raspberry Pi's GPIO pins and write a program in python to recognise the button presses. Finally we'll examine and run a space invaders game, modified with the methods we've just learnt about to work with our controller.

What's in the kit?

In your bag you will find the following components:

- 6 x M-F Jumper leads (pin to socket)
- 1 x Mini Breadboard
- 3 x Momentary Push Buttons

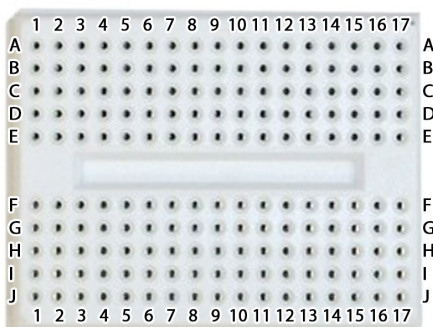
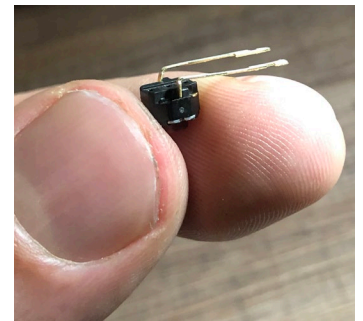
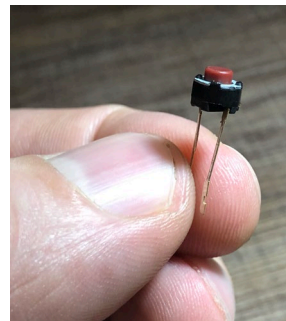


Part 1 - Let's build the controller!



Take a button by its connectors as shown and carefully bend the button so it is at a 90 degree angle to the pins.

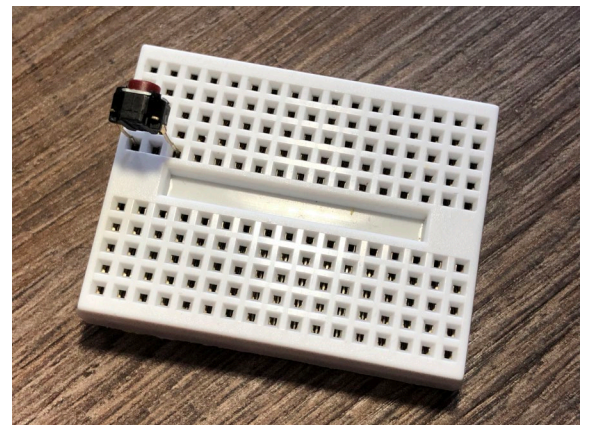
Do this slowly and gently to avoid snapping the connectors.

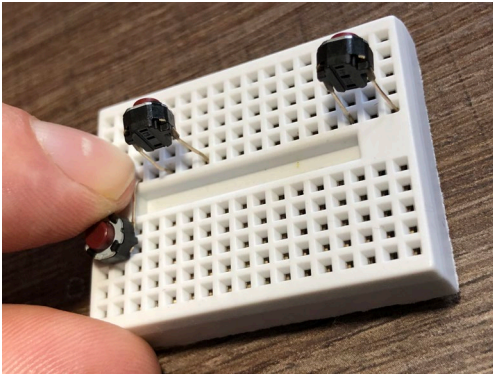


Place the breadboard in front of you as in the image to the left. Use the numbers and letters shown in the image as reference.

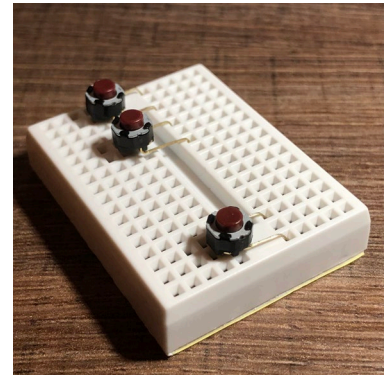
Now take your first button and push its connectors into holes **E1** and **E3** with the button facing away from you as shown on the right.

Repeat this process, placing your second button's connectors into holes **E5** and **E7** and the third button's connectors into holes **E15** and **E17**.





With all three buttons installed, gently bend the connectors towards you by applying pressure as shown on the left so that the buttons are facing upwards above rows **F** and **G** of your breadboard.

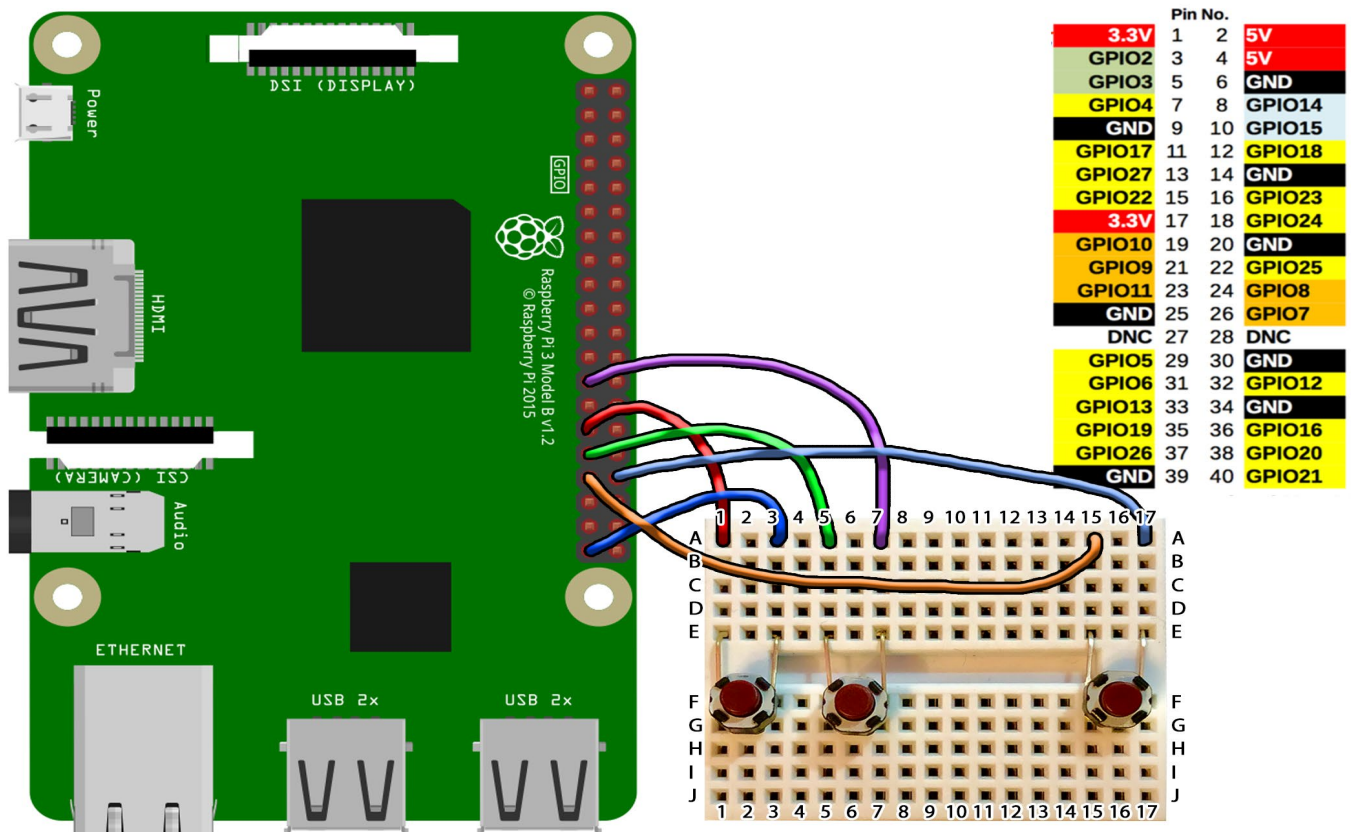


Your control panel should now look like the picture on the right.

Part 2 – Wiring your controller

The colour of the jumper wires in your kit is not important – where you put them is important! Make sure you connect to the correct holes and pins.

First, make sure your Pi is shut down and the power lead disconnected.

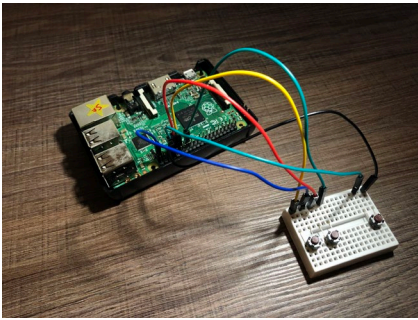


Using the diagram above for reference, follow these steps to wire your controller:

Breadboard	Pi GPIO
A1 Wire to	Pin 29 (GPIO 5)
A3 Wire to	Pin 39 (GND)
A5 Wire to	Pin 31 (GPIO 6)

A7 Wire to	Pin 25 (GND)
A15 Wire to	Pin 33 (GPIO 13)
A17 Wire to	Pin 34 (GND)

Part 3 – The Code



Now you have a controller built and wired to your Raspberry Pi's GPIO Pins, it's time to see if it works. **Make sure your wires have been checked by a tutor** before you power up your pi.

Power up your Raspberry Pi. From the desktop menu, select Programming – Thonny Python IDE.

Now use File, Save As to save your program as the name of your choice (don't forget to put `.py` on the end) **in the**

`~/python/buttoninvaders` folder.

This program shows us how to use the GPIOZero library to trigger functions in python and will also serve to test your newly built controller.

Type in the following code:


```
from gpiozero import Button
from signal import pause

button_1=Button(5)
button_2=Button(6)
button_3=Button(13)

def buttonone ():
    print("Button 1")
def buttontwo ():
    print("Button 2")
def buttonthree ():
    print("Button 3")

while (True):

    button_1.when_pressed = buttonone
    button_2.when_pressed = buttontwo
    button_3.when_pressed = buttonthree
    pause ()
```

Let's run the program. Use the run button  on the top of the window.

Now try pushing each of your buttons. If everything is working the program will show the text corresponding to each button when it's pressed.

NB: If you really don't want to type the program in you can use File, Open to open the already prepared `buttontest.py` program in the `~/python/buttoninvaders` folder.

Let's look at how this program works...

```
from gpiozero import Button
from signal import pause
```

The "from" lines tell the computer to learn about new things. Computers can learn from programs that other people have written; we call these other programs "libraries". Our

program needs the function called `button` from the `gpiozero` library which it will use to detect button presses on your controller. The function `pause` from the `signal` library so that we can insert pauses.

```
button_1=Button(5)
button_2=Button(6)
button_3=Button(13)
```

These lines set the names of our buttons and tell the program which GPIO pin they are connected to. So now the program will see our three buttons connected to GPIO pins 5, 6 and 7.

```
def buttonone ():
    print("Button 1")
def buttontwo ():
    print("Button 2")
def buttonthree ():
    print("Button 3")
```

Here we are setting up functions called `buttonone`, `buttontwo` and `buttonthree`. Functions contain multiple instructions, ready to be triggered by an event or input. In this case, each function only performs one task, and that is to print some text on the screen. When we look at the space invader game code, we will see functions like this performing multiple tasks at once when triggered.

```
while True:
```

This “`while True:`” tells the program to run forever in a loop.

```
button_1.when_pressed = buttonone
button_2.when_pressed = buttontwo
button_3.when_pressed = buttonthree
pause ()
```

Now we are telling the program to trigger a function when a button is pressed.

Part 4 – Let’s Play Space Invaders


Now our control panel is built and tested, it’s time to defend earth against an alien attack!

In Thonny Python IDE, Use File, Open to open `buttoninvaders.py` program in the `~/python/buttoninvaders` folder.

Now enable line numbers: from the Tools menu select options, choose the editor tab, then tick the box by ‘Show Line Numbers’.

Look at the following lines of code and see if you recognise the way this game has been modified to work with our controller.

Lines **12 to 20**, lines **55 to 63**, and lines **451 to 484**. These lines are commented to explain the modifications.

Now, click the run button  and our modified space invaders game will launch.

Button one is left, button two is right and button three is fire. You can press fire or the space bar to start the game.