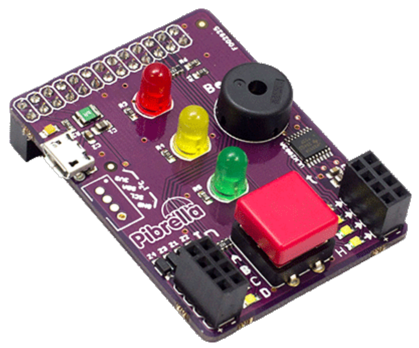
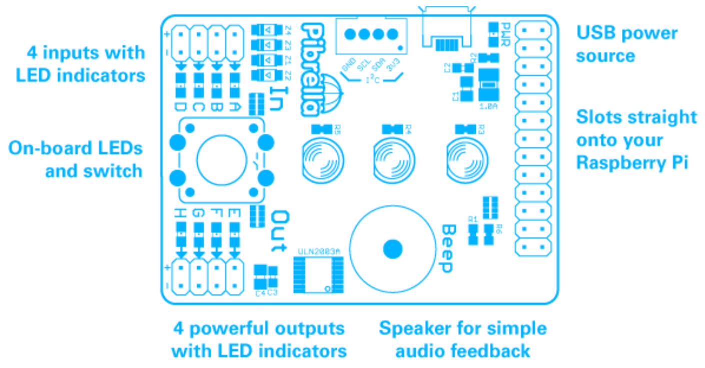
Pibrella & Python

What is it?

The Pibrella is an add-on for the Raspberry Pi that provides ready made outputs (lights and sounds) and input (a big red button!). It also has the option of extra inputs and outputs which are protected so we should blow anything up!




4 inputs with LED indicators
On-board LEDs and switch
USB power source
Slots straight onto your Raspberry Pi
4 powerful outputs with LED indicators
Speaker for simple audio feedback

Images from http://www.pibrella.com

Table of Contents

Oooh, flashy lights! Let’s get started

Step 1 - Power up and log on (probably done already)

1. If it’s not already running plug the micro-USB (phone charger) lead into the connector on the board
2. If prompted for a login use ***pi*** as the user and ***raspberry*** as the password, don’t panic; you won’t see the password as you type, just press the Enter key when you’re done
3. Type ***startx*** and press Enter
4. The “magic” will start to happen and eventually you’ll get to a desktop with a giant raspberry on it. Congratulations, Step 1 complete!

Step 2 - Starting IDLE (aka getting your Geek on)

1. Double click the ***LXTerminal*** icon - a small black console window will appear
2. Create a folder for your code - we’ll use the current time so we don’t clash with others:
    a. At the prompt type ***mkdir*** a space and the time, so ***mkdir 1030*** or ***mkdir 1451***. Check the clock in the bottom right if you need to. Don’t forget to press Enter
    b. Now type ***cd*** and the time you just entered (i.e. ***cd 1030***) and press Enter
    c. You should now be in your own folder ready to code
3. At the prompt type ***sudo idle*** and press Enter - another window will open
4. From the *File* menu select *New Window* - a blank white window will appear
5. Aaaand breathe - you’re ready to code

Task 1 - Flashy Things

Don't worry, we'll start off small. Down the left will be the code you need to enter, on the right some explanation of what you are doing. One thing to remember when coding in Python; indentation matters! So if a line of code is further to the left than the line above use the Tab key to put in the extra space. Let's get coding!

`import pibrella`
`import time`
	The Pibrella comes with it's own set of commands, this tells Python we want to use them. You'll see what `time` is for in a bit

`pibrella.light.red.on()`
`time.sleep(1)`
`pibrella.light.red.off()`
`pibrella.light.amber.on()`
`time.sleep(1)`
`pibrella.light.amber.off()`
`pibrella.light.green.on()`
`time.sleep(1)`
`pibrella.light.green.off()`
`pibrella.light.on()`
`time.sleep(1)`
`pibrella.light.off()`
	Red LED goes on, then goes off, etc. See, not that complicated

Before we get too far let's check it works. *File - Save* (or CTRL+S) and give it a filename of ***task1.py***. Now it's saved press F5; if the LEDs went on then off in sequence then we're in business. Otherwise, have a look at the code above and check for typos.

So, we can light up LEDs by colour and all at once, what next? Let's have a private disco! Modify your code to the example below

`import pibrella`
`import time`
 	
`pibrella.light.green.blink(2, 1)`
`pibrella.light.red.pulse(1, 3, 0.5, 0.5)`
`pibrella.light.amber.fade(0, 100, 10)`
`time.sleep(10)`
`pibrella.light.off()`
	I'm not going to give you all the answers :-)

Save and run to see what happens. Have a play with the numbers to see what happens, try a command without specifying the colour (i.e. *light.blink* etc). Theres a quick command reference at the back of this booklet.

Close the code window when you're done and move on to task 2.

Task 2 - Let’s make some noise!

Flashy things are all well and good, but let’s upset your neighbours and make good use of that buzzer. Open a new code window for the following…..

`import pibrella`
`import time`
 	
`pibrella.buzzer.fail()`
`time.sleep(2)`
`pibrella.buzzer.off()`
`pibrella.buzzer.success()`
`time.sleep(2)`
`pibrella.buzzer.off()`
	Things go buzz (hopefully)

Dead simple I know, but let’s see if it works, save the code (call it ***task2.py***) and run it. Assuming it all works, let’s knock up a tune….

```
import pibrella
import time

for count in range(1,3):
```

- The *for* function allows us to repeat code
- The *range* function tells the *for* function to start at one number (1) and keep adding to *count* and stop before it reaches the second number (3)

```
        pibrella.buzzer.note(7)
        time.sleep(0.5)
        pibrella.buzzer.off()
        time.sleep(0.25)
        pibrella.buzzer.note(5)
        time.sleep(0.5)
        pibrella.buzzer.off()
        time.sleep(0.25)
        pibrella.buzzer.note(3)
        time.sleep(0.5)
        pibrella.buzzer.off()
        time.sleep(0.25)
```

- Don't forget the indent, this tells Python this block of code is part of our *for* loop
- The *note* function attempts to play a musical note. Note 0 = middle A (440Hz), 1 = A#, 2= B and so on

Run it and name that tune! Hopefully it's the start of Three Blind Mice (e d c), but I'm no musician. Again, have a play, build on the tune if you like.

Task 3 - Big Red Button!

So by now you've probably been pushing the big red button. Lets face it NO ONE can resist a big red button. So crack on…..

`import pibrella`
`import time`
 	
`isRunning = False`
 	
`while True:`
	● *isRunning* is a variable to hold a true or false value - you'll see why in a b
	
	● *while* is another loop type, in this case we using it as a loop that never stops (infinite)

`    if pibrella.button.read() == 1:`	● An *if* statement; "if this then do that", the == is telling Python to compare *read()* and 1 to see if they're the same
`        pibrella.buzzer.note(0)`	● Be rude not to

`        if isRunning == True:`
`            pibrella.light.off()`
`            pibrella.output.off()`
`            isRunning = False`
	● Another *if*, then some code to run when the condition (*isRunning* == *True*) is met

```
        else:
            pibrella.light.red.pulse(0.2)
            pibrella.light.amber.pulse(0.5)
            pibrella.light.green.pulse(0.3)
            pibrella.output.e.pulse(0.2)
            pibrella.output.f.pulse(0.4)
            pibrella.output.g.pulse(0.1)
            pibrella.output.h.pulse(0.3)
            isRunning = True
```

- *else* - as the name suggests, what to do if the *if* condition is NOT met
- The *output* e - f are to the right of the button. These would normally be wired to something (more LEDs, a motor etc), but they also have a white LED to show they're on - woohoo more LEDs!

```
    time.sleep(0.2)
    if pibrella.buzzer.read == 1:
        pibrella.buzzer.off()
```

- This is to make sure the button press isn't detected several times if it's held down too long, it is indented to the same level as the *if* and *else*

You're done, save as ***task3.py*** and run it. You now have a disco at the touch of a button! To stop the code from running press CTRL+C

Achievement Unlocked!

Assuming everything up to this point has gone according to plan good job, you've learnt a bit of Python and how to program the Pibrella. If you fancy a go at a couple of projects read on. Equally, hack your own ideas.

Project - Singing Jellybaby

Ask for the Jelly Baby kit. Insert the long ends of the two wires into the jelly baby so they're close but not touching. Put the other ends into the two holes for Input A. Now try the following

```
import pibrella
import time

tune = [3, 5, 7, 8, 10, 12, 14, 15]
note = 0
step = 1

while pibrella.button.read() == 0:
    if pibrella.input.a.read() == 1:
        pibrella.buzzer.note(tune[note])
        time.sleep(0.5)
        pibrella.buzzer.off()

        note += step
        if note >= len(tune):
            note = 0

print 'Encore! Encore!'
```

The `tune` variable uses a concept we've not covered; this is called a List. All we are doing here is specifying a sequence of notes (represented by 0 to 7) that we want to play. In the example this is a scale, but you could make it anything you wanted.

The rest of the code hopefully makes sense, but just to make sure:

- While the red button has not been pressed
- Check to see if the jelly baby (Input A) has been squeezed
- Play a note from the list
- Add 1 to the note variable
- If we've got the the end of the list, start from 0 again

Extensions: Make the jelly baby go up and down the scale (without changing `tune`)
Make the jelly baby only sing one note after each squeeze (i.e. not continuous)
Teach the jelly baby a new song

You can eat the jelly baby when you're done if you wish, otherwise please dispose of it humanely ;-)

Pibrella & Python

Project - Reaction Game 1

```
import pibrella
import time
import random

startTime = 0
endTime = 0

print 'Press the button when you see the LEDs light up'

time.sleep(random.randint(1,6))
pibrella.light.on()
startTime = time.time()

while pibrella.button.read() == 0:
      time.sleep(0.01)

endTime = time.time()
print 'Your reaction speed was', endTime - startTime, 'seconds'
time.sleep(5)
```

- *random* as the name suggests is a random number generator. *randint* selects a whole number between 1 and less than 6
- *print* outputs text to the console

Extensions:

Use the buzzer
Use the LEDs as a count down
Put it in a loop and remember the fastest time

Project - Reaction Game 2

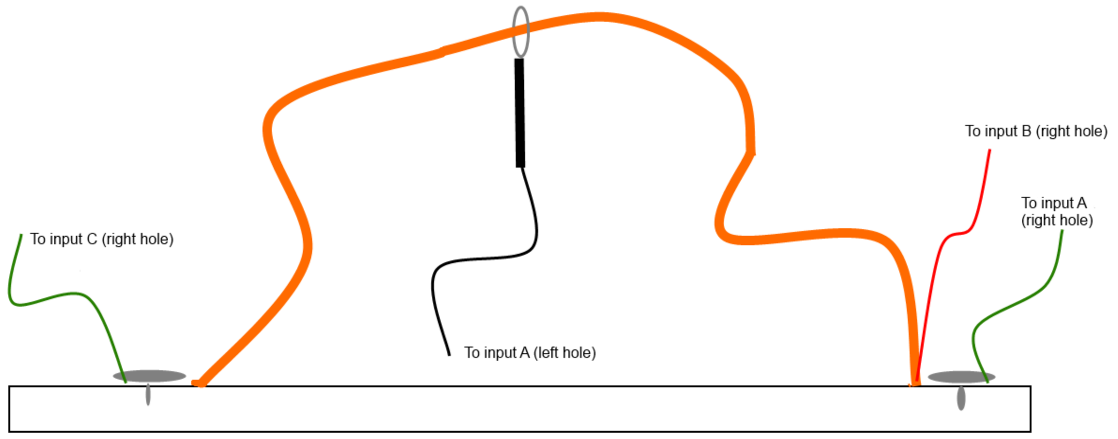
```
import pibrella
import time
import random

missed = False
hits = 0
led = 0

print 'Press the button when you see the amber (middle) LED light up'
pibrella.light.fade(100,0,5)
time.sleep(5)

while missed == False:
      led = random.randint(1,3)

      pibrella.light.off()

      if led == 1:
            pibrella.light.red.on()
      elif led == 2:
            oldHits = hits
            pibrella.light.amber.on()

      elif led == 3:
            pibrella.light.green.on()
```

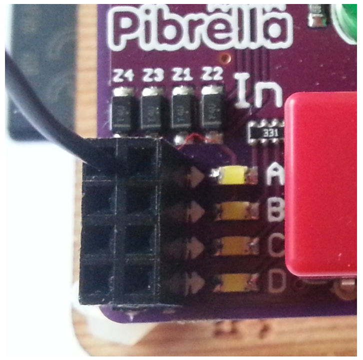
```
    for timeout in range(1, 10):
        time.sleep(0.1)
        if pibrella.button.read() == 1:
            if led == 2:
                hits += 1
                print 'hit'
            else:
                missed = True
                print 'miss'

            time.sleep(0.5)
            break

    if led == 2 and oldHits == hits:
        missed = True
        print 'too slow'

print 'You reacted correctly', hits, 'times'
time.sleep(5)
```

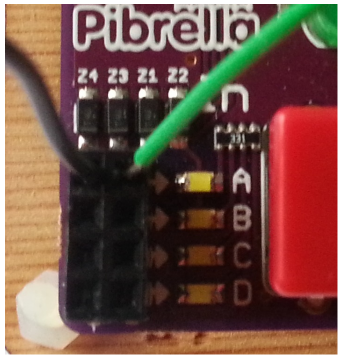
Extensions: You may notice this game has a flaw in its logic, can you find it and fix it?
Can you make the speed the lights change less predictable?

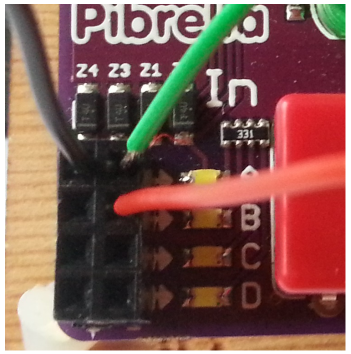
Project - Steady Hand Game

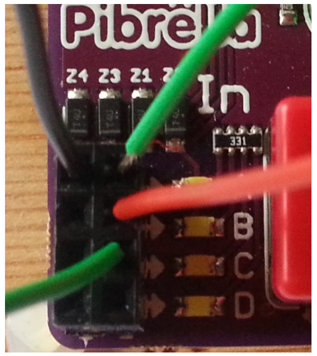
This one will require you to grab one of the Steady Hand kits and wire it up as shown below:




Black wire from handle


Green wire from right nail

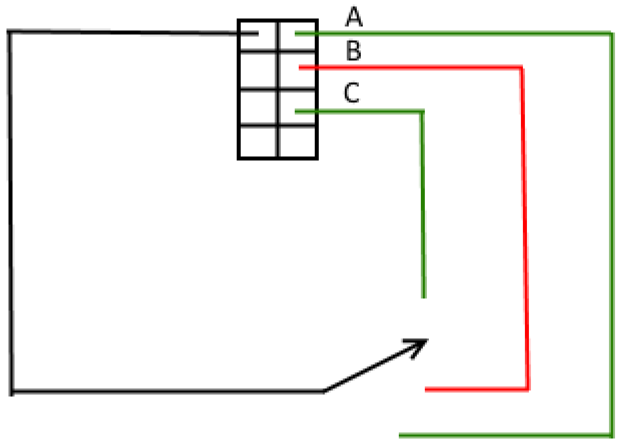

Red wire from copper wire


Green wire from left nail

NOTE: Only the black goes in the left side of the connector, the green and red go in the right side.

How it works

In principle you've just created a three way switch, with the handle (black wire) as the common "pole". When the wire loop touches the right nail, it completes the circuit for the A input, just like a switch. If the loop touches the copper wire then it completes the circuit with the B input (the red wire). Touch the end nail and close the circuit for C.



When the circuit is closed the Pibrella and the Pi detect this; just the same as when you pushed the big red button. Go and enter code on the following page and see how quickly you can get from one end of the copper wire to the other!

Pibrella & Python

```
import pibrella
import time

state = 'end'
startTime = 0
endTime = 0
touch = 0

while True:

    if pibrella.input.a.read():
        if state != 'reset':
            startTime = 0
            touch = 0
            state = 'reset'
            print 'Home and reset'

    elif pibrella.input.b.read():
        if state == 'started':
            print 'Touch!'
            touch += 1
            state = 'touch'
            pibrella.light.on()
```

- This time we're using the *input* from the Pibrella (a - c) rather than the button
- != is the short hand for 'not equal to', i.e. the opposite of ==
- *elif* this is short hand for 'else if' and is used as part of the if statement if you want to test a number of conditions
- *state* is just a string variable, but here we're using it to track what our program is doing, i.e. the state it's in. This allows us to perform certain tasks only when we need to. In programming terms this is known as a state machine

```
    elif pibrella.input.c.read():
        if state == 'started':
            endTime = time.time()
            state = 'end'
            print 'You made it in', endTime - startTime, 'seconds!'
            print 'You had', touch, 'touches of the wire'

    elif not pibrella.input.a.read():
        if state == 'reset':
            print 'Timer started - Good luck!'
            state = 'started'
            touch = 0
            startTime = time.time()
        elif state == 'touch':
            state = 'started'
            pibrella.light.off()

    time.sleep(0.1)
```

Extensions: How about changing it so you have 3 lives?
Set a time penalty for each touch?
High scores?
Buzzer?

Acknowledgements

Raspberry Pi and the Raspberry Pi logo are trademarks owned by the Raspberry Pi Foundation (http://www.raspberrypi.org)
Pibrella is a joint venture by Cyntech (http://www.cyntech.co.uk) and Pimoroni (http://www.pimoroni.com)
Pibrella Cheat Sheet by Dan Aldred http://www.tecoed.co.uk/uploads/1/4/2/4/14249012/pibrella_commands.pdf
Singing Jelly Baby inspired by http://www.raspberrypi.org/learning/screaming-jellybaby/
Steady hand game inspired by http://www.themagpi.com/issue/issue-5/article/steady-hands/