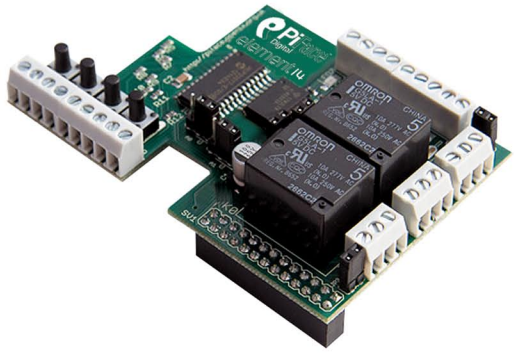
PiFace Digital

Getting started with PiFace Digital ...in less than 10 minutes!



For more applications, tutorials and software visit:
element14.com/raspberrypi

PiFace Digital is a quick and easy way to connect your Raspberry Pi to the real world. Follow this guide and your Raspberry Pi will be reacting to switches and controlling motors and lights in less than ten minutes.

Features at a glance

- Plugs directly onto Raspberry Pi GPIO socket
- 2 changeover relays
- 4 tactile switches
- 8 digital inputs
- 8 open-collector outputs
- 8 LED indicators
- Easy to program in Python, Scratch and C
- Graphical emulator and simulator

Attaching PiFace Digital to Raspberry Pi

PiFace Digital sits neatly above the Raspberry Pi and connects using the expansion connector. Take care to ensure all expansion pins are lined up with the holes on the PiFace socket. Check the alignment for left and right and front and back and never force the boards together if they don't slide smoothly.

Installing the software

The fastest way to get started is to download a prepared operating system image and copy it to an SD card. Images are available from:

http://pi.cs.man.ac.uk/download/

Alternatively, you can install the necessary libraries to your own Raspbian image with the instructions below.

Installing the software manually in Raspbian

PiFace Digital communicates with the Raspberry Pi using the SPI interface. The SPI interface driver is included in the later Raspbian distributions but is not enabled by default.

You can always enable the SPI driver, or you can load it by hand when required.
To always enable the SPI driver:

• After logging in, edit /etc/modprobe.d/raspi-blacklist.conf
By typing:

```
sudo nano etc/modprobe.d/raspi-blacklist.conf
```

• Insert a hash (#) at the start of the line blacklist spi-bcm2708
It should read:

```
#blacklist spi-bcm2708
```

Alternatively, to load the SPI driver by hand (will not be loaded on reboot):

• Type in a terminal:

```
sudo modprobe spi-bcm2708
```

Next, you we need to install the PiFace Digital libraries and change the permissions of the SPI interface. The following script automates this into one command.

To install and setup the software, ensure your Raspberry Pi can access the internet and type:

```
sudo apt-get update
wget -O - http://pi.cs.man.ac.uk/download/install.txt | bash
```

The software will complete installing in a few minutes.
Reboot your Raspberry Pi by typing:

```
sudo reboot
```

Testing

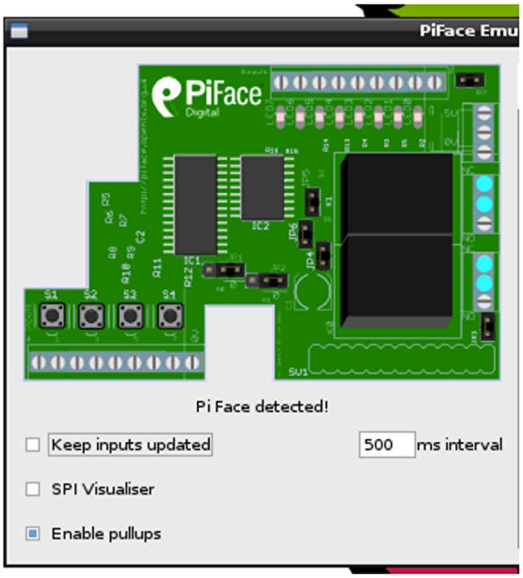
After installing the software and restarting, login and type startx to launch the desktop environment.

Start the PiFace emulator by typing in a terminal:

```
piface/scripts/piface-emulator
```

Outputs

We want to manually control the outputs, so in the PiFace Emulator window, click **Override Enable**.



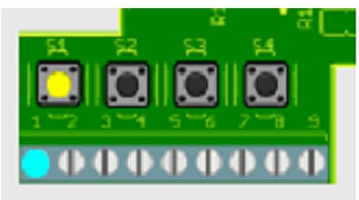
Toggle **Output Pin 1** on by clicking on it. The PiFace interface will click as the relay turns on and the corresponding LED will illuminate. Notice the graphic onscreen updates to show the LED being on, the contacts have changed over on the relay and the first output pin is on.

The LEDs are in parallel with the outputs terminal connectors and indicate when the output is enabled.

Inputs

We want to observe the inputs so click **Keep inputs updated** checkbox so the emulator reads the buttons and updates the screen. The interval sets how often the inputs are read, for most cases, it is fine to leave it on 500 ms.

Press one of the tactile buttons on the bottom left of PiFace. Notice how the on-screen representation changes to indicate the switch has been pressed.



First steps with Python

To use Piface with Python import the piface.pfio module:

```
import piface.pfio
```

Before use, the board must be initialised with a call to `init()`.

There are three main functions to control the interface:

• `digital_read(pin_number)`
returns 1 or 0 depending on the state of the input numbered pin_number

• `digital_write(pin_number, state)`
sets the output numbered pin_number to state 0 or 1. State 1 turns the LED on and enables to open collector to sink current

• `digital_write_pullup(pin_number, state)`
sets a 10k pullup on input numbered pin_number to be state 0 or 1. State 1 is pullup enabled

Simple Python examples

1. Controlling an output (turn a relay on)

To turn the relay on function as shown below. If a button is pressed the function returns a 1, otherwise it returns a 0.

Start a new python interpreter and type the following:

```
import piface.pfio as pfio
pfio.init()
pfio.digital_read(1)
```

2. Flashing a LED

```
from time import sleep
import piface.pfio as pfio
pfio.init()
while(True):
    pfio.digital_write(0,1) #turn on
    sleep(1)
    pfio.digital_write(0,0) #turn off
    sleep(1)
```

3. Reading an input

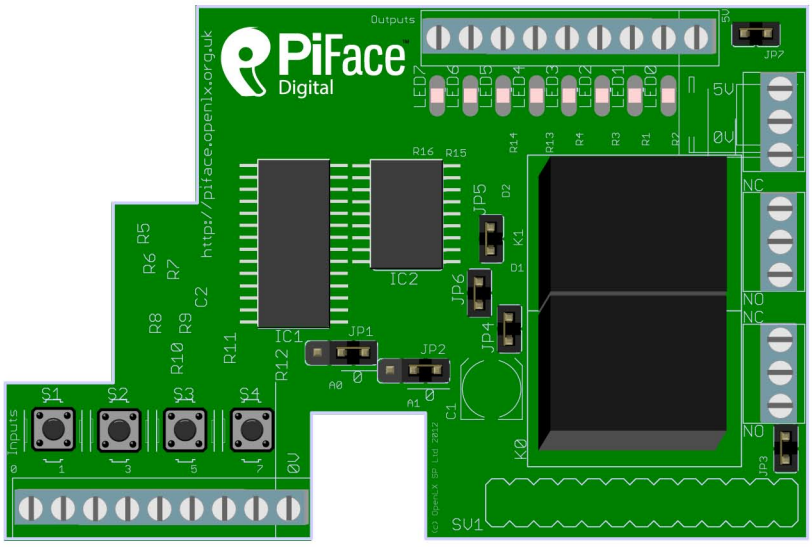
To read the state of an input use the **`pfio.digital_read(pin)`** function as shown below. If a button is pressed the function returns a 1, otherwise it returns a 0.

1. Start a new python interpreter and type the following:

```
import piface.pfio as pfio
pfio.init()
pfio.digital_read(1)
```

2. Python prints 0.
3. Hold down switch number 1 (marked S1) and type **`pfio.digital_read(1)`** again.
4. Python prints 1.

Tour of the PiFace Digital

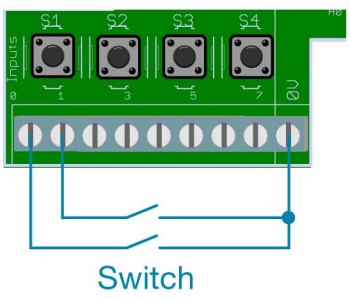


Input ports

The 8 inputs are found on the bottom left of the board and are used to detect if a switch or contact is open or closed. An input will register if the input pin is connected to 0V. The inputs are numbered 0 to 7 from left to right. The right most pin is 0V.

The four switches, numbered S1 to S4 are connected in parallel to the first four (0-3) inputs)

example connection:

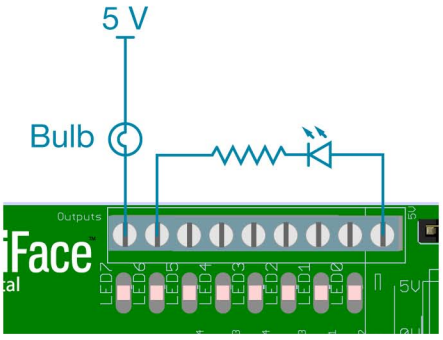


Output ports

The 8 open-collector outputs found at the top of board can be used to control devices, such as lights motors or relays. Because the outputs are 'open-collector', they do not output any voltage, instead they enable or disable current to flow to ground. This gives greater flexibility as different outputs can control devices that operate at different voltages (since the PiFace doesn't supply the voltage). An open-collector can be thought of as a switch that connects the output pin to ground, and so the circuit must be constructed taking this into account.

N.B. If the outputs are to be used for devices that operate at greater than 5V then jumpers must be set appropriately to avoid damage.

example connection:

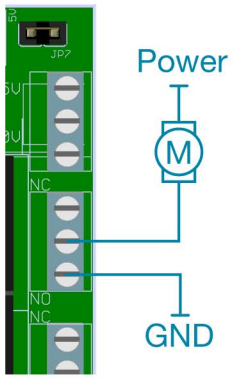


Relay outputs

There is a changeover relay in parallel with the first two outputs. The bottom set of three contacts correspond to the relay connected to output 0 and the top set to output 1. For each set, the centre terminal is common, the top being normally closed and the bottom being normally open. i.e. when the relay is off the centre terminal is connected to the top terminal, and when the output is enabled, the relay changes over so that the centre terminal is connected to the bottom terminal.

N.B. The relays may be deactivated by removing the appropriate jumper as detailed in Jumper settings.

example connection:



PiFace Digital

Power Connections

PiFace has flexible power configurations. PiFace requires 3.3 V to supply the integrated circuit (which requires negligible current), which it always takes from the Raspberry Pi. The relays require 5V to operate. The output connector block contains a terminal, which is used to supply power, which by default supplies 5 V.

N.B. As discussed in the Output ports section PiFace does not supply power on its open-collector output pins or relay contacts.

Piface can be configured to operate on an independent 5V power supply, or be connected to the Raspberry Pi 5 V pin. In this case it can take power from the Raspberry Pi, or supply the Raspberry Pi with 5 V.

The top set of terminals on the right of the board is used to supply or take power, with the top pin being 5 V and bottom two ground. In many applications these do not need to be connected to anything, as the PiFace will be powered by the Raspberry Pi. These terminals may be unsoldered and replaced with a barrel jack by the user.

Using Multiple PiFaces

More inputs and outputs can be provided by stacking multiple PiFace interfaces using an appropriate connector (e.g. PiFace PiRack). To distinguish between interfaces a different address must be set (see section Jumpers).

Jumpers

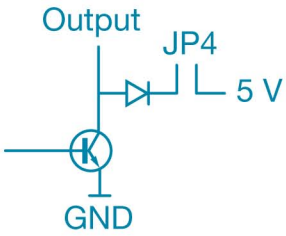
In most cases all the jumpers can be left in place.
• JP1 and JP2 set the address of the board to enable multiple boards to be used together. The default is board address 0.

Board Number	JP1	JP2
0	0	0
1	1	0
2	0	1
3	1	1

• JP3 connects to 5 V rail on Raspberry Pi. With it, Raspberry Pi can be powered from the 5V connection on the PiFace, or the PiFace can be powered from the Raspberry Pi.

• JP4 connects the snubber diodes from the ULN2803A to 5 V (snubber diodes protect the driving transistors from the high voltages that occur when a coil e.g. a relay turns off). However, if the open-collectors are connected to > 5 V, these must be disconnected (else the diodes will conduct between the outputs and 5 V)

Snubber diode and jumper (JP4) circuit:



• JP5, JP6 are used to disconnect the relays (remove the jumpers to disable). This is useful if you just want to use the open collectors outputs.
• JP7 can be used to disconnect the power to all onboard outputs (i.e. disable the relays and leds).

Equivalent Circuits

The equivalent circuits for inputs and outputs on the PiFace can be expressed as shown: