Teacher's guide to the Raspberry Pi GPIO worksheet

The Raspberry Pi GPIO worksheet is an introduction to using the GPIO ports for input and output. This is suitable for running as a classroom session or for a Raspberry Pi / STEM club.

Resources required

Students can work in groups or individual. Small groups of two to three students work well. Each group need the following:

- Raspberry Pi (model A or B) complete with accessories eg. power supply, keyboard and mouse.
- SD card with Raspbian installed (eg. using NOOBs)
- Breadboard *
- Male to female jumper leads
- Male to male jumper leads (or solid core wire)
- 3 x LEDs (1 x Red, 1 x Orange and 1 x Green)
- 4 x 220Ω resistors
- 1 x push-to-make switch (breadboard mounted or with leads attached) – optional
- Internet access on the Raspberry Pi is and advantage, but not required

* A breadboard is recommended, but this can be made using jumpers or crocodile clips.

Pre-requisites

- A familiarity with the Raspbian desktop including launching a terminal
- Some experience with Python is recommended
- Basic understanding of electronics recommended – switch, LEDs and resistors

Time required

The time required will depend upon the ability of the students, and how much experience they have with the Raspberry Pi. I have tried this with an after-school Raspberry Pi club and whilst they had completed the circuit and entered the Python code within 2 one hour sessions they did not get time to test it properly. I would therefore recommend at least 3 one hour sessions.

Task 1 – Identify the GPIO ports (pg 1)

To ensure that the students work through the exercises then sheet 1 should be provided first. The students should be able to research the GPIO port allocations on the internet. If they are working directly on a Raspberry Pi that doesn't have internet access then they could be provided with the GPIO pin layout, or they can look at page 2 which shows how the ports are wired up.

There is a good diagram showing the GPIO ports at: http://elinux.org/File:GPIOs.png and a diagram showing how the physical numbering works at: http://elinux.org/File:RPi_P1_header.png

Both of these are taken from page: http://elinux.org/RPi_Low-level_peripherals

Task 2 – Wire up the circuit (pg 2 to 4)

Details of how to wire and test the circuit are provided on pages 2 to 4. This includes a wiring diagram. The colours of the wires shown are not important, they were mainly used to make the diagram clearer.

Details are provided on which way to insert the LEDs, there is no risk of damage if they are inserted the wrong way around, but they will not light unless wired correctly. LEDs should not be connected without some kind of resistor (larger can be used, but will result in less bright LEDs). Without a resistor then there is risk of permanent damage to the Raspberry Pi.

Different switches may have a different configuration. If in doubt test the pins with a multi-meter.

The instructions for testing the output using Python is provided. This must be run with root permissions to be able to update the GPIO ports. It therefore needs to be run with sudo. The LED example uses GPIO 22, but this should be repeated by the student for 23 and 24.

A common warning message when running the code is:
“This channel is already in use”.
This can normally be ignored as it indicates that the cleanup function hasn't been run. In the next stage these messages are disabled.

Task 3 – Create a program (pg 5 to 6)

Commands entered in the shell will be lost. Adding them into a File will allow them to save the programs for future use, and makes it much easier to change the program.

If the students are already experienced in writing Python programs then they should be able to write their own simple program based on the commands used in the previous stage. If not then pages 5 and 6 provide a full working program.

It is important for the correct spacing to be used within the while loops, which is a feature of Python that may trip up new users. The IDLE editor should help to some extent with this.

NO Switch?

If you don't have a switch available then two wires touched together could be used as a switch. Alternatively change the code as follows:

Replace:
```
	# loops until GPIO becomes "False" ie switch pressed
	while GPIO.input(GPIO_SWITCH):
		# sleep used to reduce processor usage
		#sleep for 1/4 sec between checking switch
		time.sleep (0.25)
```

with:
```
	# automatically change to red (in place of a switch)
	time.sleep (TIME_GREEN)
```

The worksheet does not explain how the code works. It is based on simple blocks of code and has comments explaining what is happening.

Task 4 – Improve the program (pg 6)

If time allows there are suggestions on page 6 on ways in which the program can be improved. There are not solutions for these, some of which can be worked through using the existing code and others may need some internet research.

There is a warning about turning on all 6 LEDs if added. This is a recommendation to keep the current within the capability of the Raspberry Pi, but there is only a small risk of damage.

Task 5 – Future ideas (pg 7)

The final page makes suggestions on how this can relate to other activities. There are some suggestions for things to research. Some resources are listed below:

Further resources

The circuit diagrams are created in Fritzing. The Fritzing files can be downloaded from:

http://www.penguintutor.com/programming/raspi-gpio

The robot on the final page is described at:

http://www.penguintutor.com/electronics/rubyrobot

This includes a guide to driving motors using the GPIO.