**[Scratch controlling the GPIO on a RaspberryPi](http://cymplecy.wordpress.com/2012/08/26/scratch-controlling-the-gpio-on-a-raspberrypi/%22%20%5Co%20%22Permalink%20to%20Scratch%20controlling%20the%20GPIO%20on%20a%C2%A0RaspberryPi)**

***This article intends to make it as Simple as Pi to get up and running and make your Raspberry Pi control some lights and respond to switches and sensors.***

Whilst the Raspberry Pi is a great tool for the creation of software, using languages such as Scratch, Python, C etc., the best way to make it really come alive and to add even more enjoyment to this cheap, credit card sized computer is to start playing around with hardware hacking and physical computing. This involves using the Raspberry Pi to control things like LEDs, and respond to switches and sensors. More often than not it also includes knowledge and learning of both hardware and software in a very interesting practical environment – not just coding for the sake of coding but, for example, creating robots and programming them to do cool things!

This article is based on a post on the Cymplecy blog by Simon Walters (<http://cymplecy.wordpress.com>), a primary school teacher and general Scratch guru!

Minimum Requirements – a RaspberryPi with Raspbian installed and a working internet connection, a breadboard, some Light Emitting Diodes (LEDs), some resistors and some wire connectors. Total cost £5-£10.

How to get a Rapsberry\_Pi to control the GPIO Pins from Scratch
—————————————————————
Your RaspberryPi needs to be connected to the internet to install the software but not needed to run ScratchGPIO.

Copy the text below ( starting at sudo and ending at gpio.sh – dont click on it) and paste that into an LX Terminal window and run it to download the installer.

sudo wget https://dl.dropbox.com/s/k56uv4s6rxqt0bi/install\_scratch\_gpio\_1.4a.sh -O /boot/install\_scratch\_gpio.sh

and then type

sudo /boot/install\_scratch\_gpio.sh

This will install all the necessary extra software and some simple examples.

*(If you do not have internet on your Pi then,  put your SD card into a card reader and try using your browser to right-click and save the script direct to your SD card and then put it back into you Pi and run the second instruction)*

Connecting Components Up
———————————————————————
**EXTREME care should be taken when connecting hardware to the GPIO pins.**You can cause your Pi to die by connecting the wrong things together – only do this if your confident of your ability to follow instructions correctly 
**AT A MINIMUM**- get a breadboard and use some female-male 0.1 leads (available from RS/CPC or even your local Maplin)

Check out GPIO pin guides to make sure you know what pins are what.

Wire up Pin 1 (3.3V) to (at least) a 330ohm resistor – connect that resistor to the long lead of an LED and then connect other end of LED to Pin 6 (0V).

It should light up. If it doesn’t try reversing your LED.

Now move the lead from Pin 6 to Pin 11.

Run the special Scratch icon (Scratch GPIO) on your desktop.
*(It is actually a completely normal version of Scratch, it just runs a little Python background program as well that handles communications between Scratch and the GPIO and automatically enables Scratch’s Remote Sensor Connections(RSC))*

To test out control from Scratch, open up blink11 from /home/pi/Scratch and click on the Green Flag

Your LED should now blink on for 1 second and off for 2 seconds – see trouble shooting if this doesn’t happen.

**What more can I do with Scratch and the GPIO**

You can control seven pins as outputs (Pins 11,12,13,15,16,18,21) and treat seven as simple inputs (7,8,10,19,22,24,26)

As you can see in the blink11 script , you can simply use a broadcast message telling Pins to go on or off (Up to 3.3V and down to 0V)

The valid messages are **pin11on**, **pin12on**, **pin13on**,**pin15on**, **pin16on,** **pin18on, pin21on** along with the corresponding **pin11off** etc messages.

You can also say **allon** and **allloff**.

And you can replace the work **on** with **high** and replace **off** with **low** if you want to talk in pure logic levels.

You can combine msgs together to make a single broadcast so to turn Pin11 and Pin13 on and all others off you can say

**broadcast pin11on pin12off pin13on pin15off pin16off pin18off pin21off**

or

**broadcast alloff pin11on pin13on**

To check an input, you should go into the Sensing block and click on the word “slider” at the bottom and you’ll notice that you have pins 10,19,22,24,25,7 and pin8.  If you connect a switch to one of these pins (through a resistor don’t forget) to OV, then you can detect when the switch is open or closed.  The inputs will normally read 1 and go to 0 when they are connected (through a resistor) to ground.  Click on the checkbox next to pin7 and try it out.

**Using variables instead of broadcasts**

For advanced Scratchers, you can use variables instead (or as well as broadcast messages) .

For example:  create a global variable called pin11

To make pin11 go on  use

**set pin11 on**

To make is go off use

**set pin11 off**

On can be replaced with high or 1 and off can be replaced with low or 0 so that you can use whatever logic scheme you’d like.
To set all outputs on use

**set allpins on**

To use a “bit-pattern” to set/unset multiple outputs simultaneously use

**set pinpattern 1010001**

(this will set Pin 21 , Pin 16 and Pin 11 on and all the rest off)

*Note – currently there is an unfortunate “bug” in Scratch in that it remembers variable states and only sends changes out.  Even when you press the Green Flag, it will not send the state of all the variables out, it will only send them when a variable changes.  I recommend setting any gpio variables to a . and then to their initial state in a Green Flag start-up script.*

**When GreenFlag clicked**

**set pin11 .
set pin11 off**

**Motor Control**

To control a motor, it is usually required to be able to not just switch it on or off but to vary its speed.

To do this in Scratch you can create 2 special variables called MotorA and MotorB and then assign these values from 0 (off) to 100(fully on)

MotorA controls pin11 and MotorB controls pin12.

e.g To make MotorA go at half speed use

**set MotorA 50**

To make MotorA switch off use

**set MotorA 0**

The motor variables can also be be used to simply vary the brightness of an LED.

**Ultrasonic Sensor**Support for cheap 4pin ultrasonic sensors has been added.  To use one, connect Pin23 to the trigger or pulse input and connect the echo output **(USING A 10K RESISTOR TO AVOID BROKEN PI!)** to any of the input pins  (7,8,10,19,22,24 or 26).  This simple script

**forever
broadcast sonar7
wait 1**

will send a pulse out on pin23 once per second and setup up a new sensor input called sonar7 that will display the distance (in cm) that the sensor detects.   (NB You need to have done at least 1 broadcast before the sonar7 sensor will show up as a valid option)

**Troubleshooting**
To test if the software necessary to control the GPIO is correctly installed open a LXTerminal session and type

sudo python blink11.py

If this doesn’t give an error but doesn’t make a LED on Pin 11 blink then we have real problems Houston 

Try connecting the lead going to Pin 11 back to Pin 1 to make sure the LED lights up then just in case you have a loose connection.