GCSE Computing

Raspberry Pi Recipes

www.ocr.org.uk/computing

Oxford Cambridge and RSA

RASPBERRY PI RECIPES

INTRODUCTION AND MINIMUM REQUIREMENTS

OVERVIEW

Minimum requirements tested with

Raspberry Pi Board	Rev 1 or Rev 2
Minimum Peripherals	SD Card, monitor, keyboard, mouse and power supply Working monitor setup
Optional Peripherals	Network connection may be required for initial setup or later uses
Operating System	Raspbian – tested with 2013-09-25-wheezy-raspbian NOOBS V1.3.2 (November 2013 load Raspbian)
Programming Language	Python 3 (version as supplied with operating system)
Additional Libraries or Modules	RPi.GPIO (V0.5a or higher) and standard python modules as supplied with operating system Any additional libraries/modules will be stated in examples
General Components (suggested for general use and to have to hand)	LED Resistors (1M,100k, 1k, 470R, 330R) Paper clips (normally fully or half straightened out) Jumper leads (female to female) Small switch
Other Components	Some will require small breadboard jumper leads (female to male) Other items detailed in each recipe

BASIC STRATEGY

These recipes have been designed to be easy to assemble and to use components that can be reused afterwards.

Unless otherwise stated the recipes use standard built-in software that comes with the Raspbian operating system.

GPIO USAGE

The Raspberry Pi has many GPIOs, some of which can have other functions (UART, I2C, SPI, PWM clock), in order to maintain consistency leave as many functions available as possible.

1. Same pin used for one input, normally switch or similar.
2. Same pin used for one output normally an LED.
3. I2C pins left available.
4. UART pins left available.
5. SPI pins left available.

Due to the many numbering systems...

- chip pin numbers
- chip GPIO register naming
- connector pin number
- general signal naming by Raspberry Pi Foundation
- different numbering schemes for device numbers and other libraries e.g.
  - o WiringPi
  - o Quick2Wire

... all recipes will use the connector pin numbering scheme in wiring up and using the RPi.GPIO library for consistency. Signals will be referred to by the names as in the table below that follows the Raspberry Pi Foundation method. This also reduces problems with changes made between Rev 1 and Rev 2 boards that had different pins used to produce the same output signals.

All GPIO signals are at 3V3 levels, whilst power is available as 3V3 and 5V relative to 0V (GND). All GPIO signals are measured relative to the 0V (GND) connection from the Raspberry Pi.

Simple recipes that have

- one output (normally a LED) will use GPIO-0 (pin 11 of P1)
- one input (normally a switch or device as a switch) to use GPIO_GCLK (pin 7 of P1).

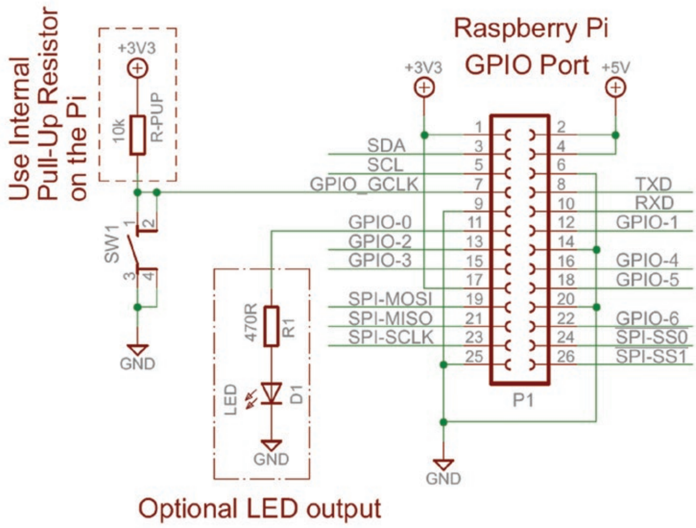
So any code written for driving an LED will be same for all recipes, and shows code reuse and hopefully later use of functions.

signal	P1 Pin		signal	Note
+3V3	1	2	+5V	
SDA (I2C)	3	4	+5V	
SCL (I2C)	5	6	GND	
GPIO_GCLK	7	8	TXD	UART From Pi
GND	9	10	RXD	UART To Pi
GPIO-0	11	12	GPIO-1	
GPIO-2	13	14	GND	
GPIO-3	15	16	GPIO-4	
+3V3	17	18	GPIO-5	
SPI-MOSI	19	20	GND	
SPI-MISO	21	22	GPIO-6	
SPI-SCLK	23	24	SPI-SS0	Low ON
GND	25	26	SPI-SS1	Low ON

CIRCUIT DIAGRAMS

Circuit diagrams will be drawn using European style of components and gates to fit in with any later work learners may do.

For example:



PROGRAMMING STRATEGY

Python version 3 will be used as the main programming language as it is structured better to match easier transfer to other languages later; also gradually more of Linux will be converted to Python 3 (a long process). Python 2 versions are classed as legacy versions even by the Python organisation. Python in general provides an easier save text file and run method which is more suitable for classrooms.

Note to access Python3 using the graphical interface requires running IDLE3 not IDLE (which is for Python 2.x), see later about Userland Access.

Where possible to use exception handling to show handling of errors and special conditions like <CTRL>-<C> to exit program.

Examples to show use of functions where possible, either as an alternative, or how to create more compact and readable code.

TRANSFERRING PYTHON PROGRAMS

Sometimes you will create programs on a PC and transfer them to a Raspberry Pi, either via network or USB pen drive. If you then try editing the program on the Pi you will find most lines ending in '^M' representing a carriage return. This is due to Microsoft Windows terminating text lines in a sequence of Carriage Return and Line Feed characters, whilst Linux use just Line Feed to terminate lines. To get around this use the command line program to convert the program. For example –

```
dos2unix filename.py
```

or

```
dos2unix *.py
```

This will change the line termination in the files to match the Linux operating system used on the Raspberry Pi.

In order to run this utility you must install it onto the Pi, this requires a network connection and executing the following commands will install the package dos2unix –

```
sudo apt-get update

sudo apt-get install dos2unix
```

USERLAND ACCESS

Userland is the loose terminology in Linux that refers to the level of permissions given to the normal user 'pi', as compared to the system administrator/operating system user of 'root'. Note by default you cannot logon to user 'root' but can temporarily gain root privileges by prefixing a command with 'sudo'.

Unfortunately to access the GPIO root privileges are needed as this is software communicating with the actual hardware of the system. So in order that any Python programs run, you must gain root privileges by doing the following –

- From command line (or terminal window)
  - `sudo python3 program.py`
  - where '`program.py`' is your Python program name.
- In graphical interface
  - On menu select "Run.."
  - Enter "`sudo IDLE3`"
  - Use Idle for Python 3 as you normally would.

COMMAND LINE EXECUTION OF SCRIPTS

Sometimes you will want to have a program run from a command line or another script, rather than using the command line

```
sudo python3 filename.py
```

(for program requiring root permissions)

```
python3 filename.py
```

(for normal programs)

First edit your python program and put the following as the **first** line in the program

```
#!/usr/bin/python3
```

When executed this line is read by the operating system which then knows to use Python 3 to interpret the program.

Then you can change the execute bit on the file permissions so it can be executed as follows –

```
chmod +x filename.py
```

Which means you can now execute the script, by entering the following in command line mode

```
./filename.py
```

The './' is required for the system to search current directory first to find your program to run, otherwise, the program will not be found.

RASPBERRY PI RECIPES – WATER DETECTOR

MAKING A FLOOD ALARM

IS IT RAIN … OR MAYBE OLD PIPES?

Still the brand new swimming pool indoors as a result of a flood isn't great. And what if you are out when it happens?

It would be nice if you had a Raspberry Pi and a couple of wires near a suspect pipe to signal when there is water flooding.

WHAT YOU WILL LEARN

During this recipe you will learn how to control the GPIO (General Purpose Input Output) pins and you will get the chance to create a working detector of water floods or rain.

You will discover water conducts electricity, making a variable resistor, resistor dividers, how parallel resistance can improve detectors and different detector shapes and sizes. Further experimentation can show different materials added to the water and its effects.

PARTS REQUIRED

Raspberry Pi Board	Rev 1 or Rev 2
Minimum Peripherals	SD Card, Monitor, Keyboard, Mouse and Power supply Working monitor setup
Optional Peripherals	Network connection may be required for initial setup or later uses
Operating System	Raspbian – tested with 2013-09-25-wheezy-raspbian NOOBS V1.3.2 (November 2013 load Raspbian)
Programming Language	Python 3 (version as supplied with operating system)
Additional Libraries or Modules	RPi.GPIO (V0.5a or higher) and standard python modules as supplied with operating system Any additional libraries/modules will be stated in examples
General Components (suggested for general use and to have to hand)	LED Resistors (1M,100k, 1k, 470R, 330R) Paper Clips (normally fully or half straightened out) Jumper leads (Female to Female) Small Switch
Other Components	Some will require small Breadboard Jumper Leads (Female to Male) These and other items detailed in each recipe

HOW THE WATER DETECTOR WORKS

A water detector uses the fact that water is a conductor.

Two conductors act as a variable resistor, open-circuit (infinite Ohms) when dry and a resistance of 20k to 40k when wet, depending on how wet and how good the detector is. You will get different resistance values when only slightly wet.

Depending on how much water we want to detect (a single drop to a flood) determines the size and shape of our detector. The best water detectors have lots of parallel wires (or tracks on a board) to cover a large area and increase chance of detecting water, and when more water is found to have an even lower resistance due to parallel resistance of multiple wires conducting at same time. Remember if trying to detect rain, this appears in various amounts from a few drops to torrential downpour it is necessary to catch the earliest possible moment of when a few drops fall.

One side of the detector has a voltage and the other is a resistor to 0V (GND) so when the detector is dry the end of the resistor to GND, that is connected to the water detector is measured as being 0V. When wet, the water detector's resistance decreases, so the voltage measured at the same point is the resistor divider of the two resistances of the water detector and resistor to GND (0V).

So when dry the Raspberry Pi reads the GPIO as 0V (equivalent to a '0') and when the water detector is wet enough the voltage rises to above 2V and is read by the Raspberry Pi as a high voltage 2V to 3V3 (equivalent to a '1').

What we do when detecting these different levels is what the software program has to do.

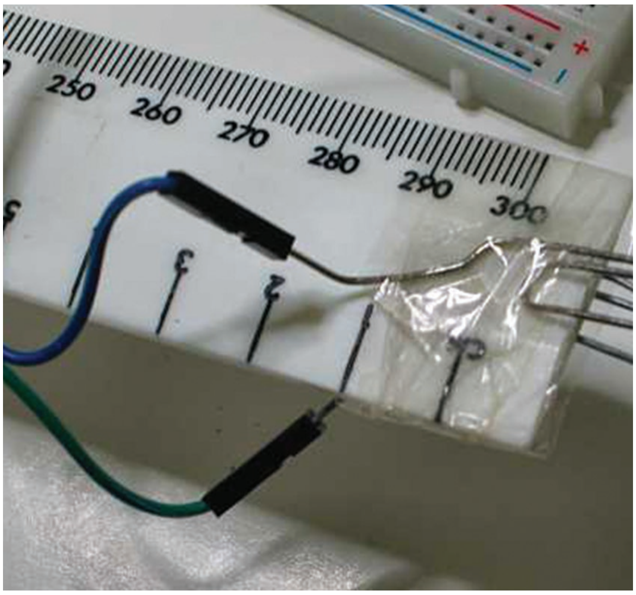
MAKING THE WATER DETECTOR

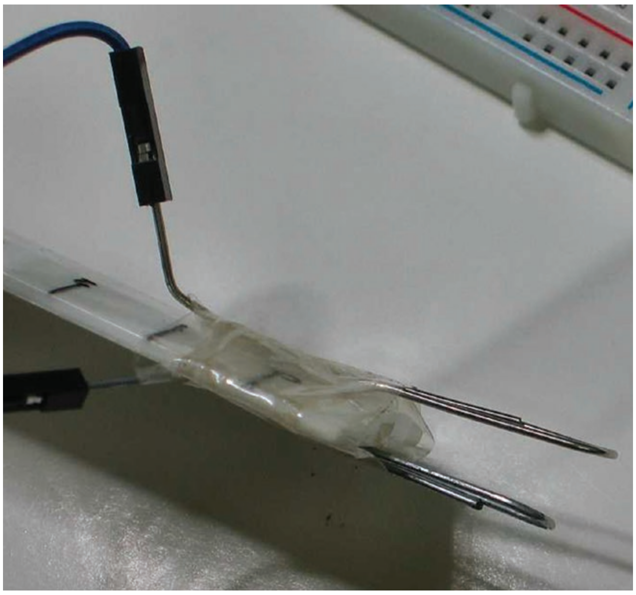
1. Take two paper clips, straighten out the outside loop of each one.

2. Slide female to male jumper lead onto each straightened out section of each paper clip.

3. Tape each paper clip to each side of a ruler sticking off the end with a bit of the loop overlapping the ruler.

4. Run tape around ruler to secure paper clips.





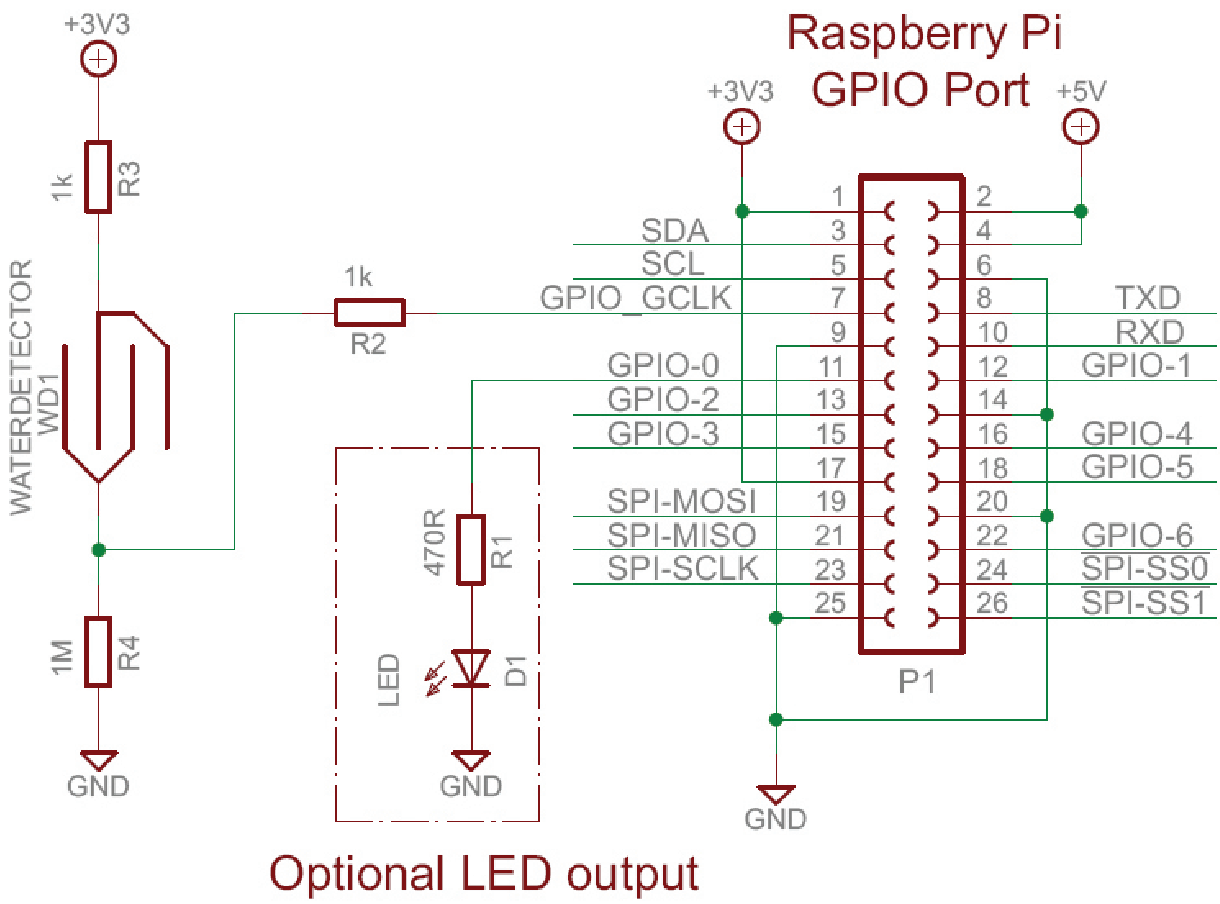
THE CIRCUIT AND CONNECTING THIS TO A PI

The circuit requires several resistors and is best assembled onto a breadboard with jumper wires to the Raspberry Pi and the water detector.

The optional LED uses a resistor to limit the current (and brightness) of the LED, this is important as directly connecting the LED will blow the LED and/or the Raspberry Pi.

See photo of breadboard circuit that shows a wired up, circuit.

When running the software, if the LED does not light, check your wiring is connected to correct pins and firmly, then also try swapping the LED round. See the section on Testing the Water Detector once you have loaded the software.



The 1k Resistors (R2 and R3) protect the Raspberry Pi, R2 stops stray voltages from the water detector entering the Raspberry Pi, whilst R3 stops accidental shorts of the water detector to other voltages to earthed metal (pipes etc). These resistors provide a bit of protection, but will not protect against all accidents.

DO NOT CONNECT OR TOUCH THE WATER DETECTOR TO HIGH VOLTAGES OR EARTHED METALWORK.

The 1M Resistor (R4) is a pull down resistor so when the water detector is open circuit, the Raspberry Pi GPIO input is pulled down to 0V (GND).

USING SOFTWARE TO READ THE WATER DETECTOR

The following is a Python3 program to continuously monitor the GPIO input the Water Detector is connected to and display a message when the Water Detector is activated (i.e. when the detector is wet enough). Save this program as *waterdetect.py* . The comments in the program explain the stages of the program, please read through it thoroughly.

The lines starting with # are comments and describe what is happening, all your programs should have comments to help others understand your program, and even you when you open the file in two weeks to see what is in the program. This way you can find which bits are useful or as an example for your next program.

```
#!/usr/bin/python3
# Water Detection
# Example to loop ‘forever’ (Use CTRL-C to exit) doing these steps
#  1 read an input for water present via the GPIO
#  2 If water detected ON (reads as 1)
#            display time stamped message on screen
#            IF connected turn a LED ON otherwise turn a LED OFF
# When
#  LED connected to GPIO bit 0 (pin 11 of P1 connector)
#   in this case a ‘1’ output turns LED ON
#  Water detection circuit connected to GCLK pin (pin 7 of P1 connector)
#
# PC Services Water Detector Example
# November 2013

# import time library for delays and time stamps
import time

# Import the necessary library to access the GPIO library
import RPi.GPIO as gpio

# First configure the port number method to match GPIO connector pins
gpio.setmode( gpio.BOARD )

# Set input pin GPIO_CLK on pin 7 is connected to Water Detector circuit
gpio.setup( 7, gpio.IN )

# Set output pin for LED off on pin 11, and initial value is OFF
gpio.setup( 11, gpio.OUT, 0 )

# Now loop forever (until someone types CTRL-C)
while( 1 ):
       # Read the input pin
       water = gpio.input( 7 )

       # Is it ON (ON = 1)
       if water == 1 :
              # Display message as Water present
              print(  “{} - Water Detected”.format( time.asctime() ) )
              # set the output to ‘1’ to turn LED on
              gpio.output( 11, 1 )
       else :
              # Otherwise set the output to ‘0’ to turn LED off
              gpio.output( 11, 0 )

       # now wait a short while, to make sure we can see a real event
       time.sleep( 0.2 )
```

RUNNING THE PROGRAM

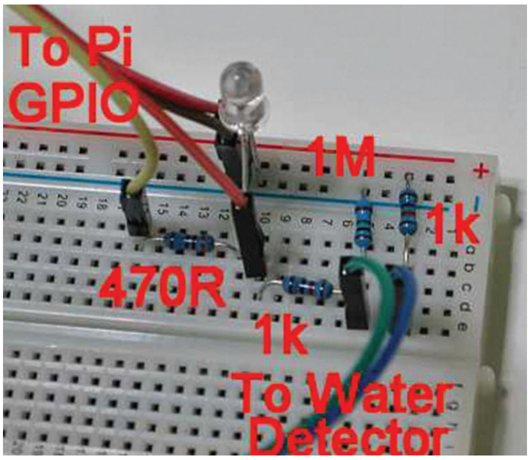
When you have the program on the Pi, you need to run the program in a special way as accessing the GPIO pins requires special permissions as follows, at the command prompt or in a terminal window:

```
sudo python3 waterdetect.py
```

What this command breaks down into

> 'sudo' tells the Pi to run this command with special permissions (like on Windows systems 'Run as Administrator')
>
> 'python3' is the command to run Python3 on the Pi
>
> 'waterdetect.py' is the filename for Python3 to read to run your program

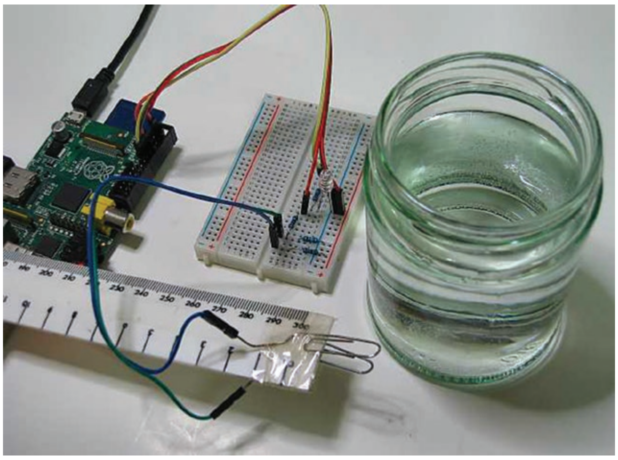
Every time the water detector is activated you will see a message displayed on the display.

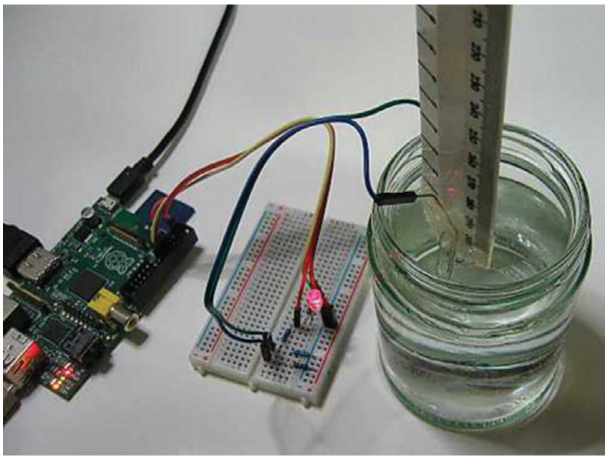


TESTING THE WATER DETECTOR

We have two stages of test that rely on the software running –

1. Touch the two wires of the water detector together and make sure the Pi screen displays a message and the LED turns ON (check wiring and if LED wrong way round).
2. Testing for water
   a. With the detector in the air, there should be no message on Pi screen and LED should be OFF
   b. Dip the detector into a cup/beaker of water, a message should be displayed on the Pi screen and the LED turns ON.





FURTHER EXPERIMENTATION

Now we have a working Water Detector, find out –

- How far in the water does the detector have to be for detection to work?
- If we add other materials to the water (dirt, sand, salt) does this change the depth at which the detector works at?
    - If the depth changes why?
- For detecting rain, what size and shape of detector can you come up with and why?
- What shapes of detector would be better for detecting small water drips and why?

EXTENDING THIS EXAMPLE

From the software you can see a message is displayed when water is detected but we can also do other things at the same time.

Other things you could do to extend the software are –

- Log the detection times to a file on the Pi
- Log the time of detection and how long the water is detected for
- Call a program to send a text message or email that water has been detected
- Change the LED output to a siren
- Detect different water levels on the ruler.

This resource has been created from the winning competition entry for a Raspberry Pi recipe idea by Vladimir Marinov.

RASPBERRY PI RECIPES – PRESSURE PLATE

DETECT SOMEONE ENTERING A ROOM

HOW TO DETECT AN INTRUDER ENTERING A ROOM OR A BUILDING

This can be done in many ways, what will be shown here is a method of using a pressure plate, that when somebody steps on it, will complete an electrical circuit. This could be used to sound an alarm, turn a light on or many other methods of signalling this event, however here we will look at using that Pressure Plate to signal to a Raspberry Pi.

When connected to a computer like the Raspberry Pi, we can do more things like -

- Note the time the Pressure Plate was activated
- Sound an alarm
- Turn lights on
- Connect to other parts of security systems to activate other locks or measures
- Using the phone or internet networks to alert others that the Pressure Plate has been activated.

WHAT YOU WILL LEARN

During this recipe you will learn how to control the GPIO (General Purpose Input Output) pins and you will get the chance to create a working pressure plate to detect people entering a room.

You will discover how to make a simple switch and the use of materials as a spring to automatically open a switch.

PARTS REQUIRED

Raspberry Pi Board	Rev 1 or Rev 2
Minimum Peripherals	SD Card, Monitor, Keyboard, Mouse and Power supply Working monitor setup
Optional Peripherals	Network connection may be required for initial setup or later uses
Operating System	Raspbian – tested with 2013-09-25-wheezy-raspbian NOOBS V1.3.2 (November 2013 load Raspbian)
Programming Language	Python 3 (version as supplied with operating system)
Additional Libraries or Modules	RPi.GPIO (V0.5a or higher) and standard python modules as supplied with operating system Any additional libraries/modules will be stated in examples
General Components (suggested for general use and to have to hand)	LED Resistors (1M,100k, 1k, 470R, 330R) Paper Clips (normally fully or half straightened out) Jumper leads (Female to Female) Small Switch
Other Components	Some will require small Breadboard Jumper Leads (Female to Male) These and other items detailed in each recipe

WHAT IS THE DIFFERENCE BETWEEN A PRESSURE PLATE AND A SWITCH?

The Pressure Plate is a special form of switch, the simplest form of switch is two wires without insulation which we touch together to make a connection and take apart to break the connection. However this is too simple for a Pressure Plate, what we need is a large weight (the pressure), in this case the weight of a person, to be enough to operate as a switch, most switches would break if we stood on them. Also we need this to be a large area so we can detect a person's foot.

The Pressure Plate is basically a simple switch, that when someone stands on it will make two sides touch each other making a circuit, and when they step off the Pressure Plate, the two sides open up again, breaking the circuit. The two sides of our Pressure Plate switch will be foil covered cardboard, with some cardboard spacers between them. So when pressed together the tin foil on the two sides touch making the connection. The cardboard acts as a spring so when the weight of a person is removed (step off) they spring apart again, breaking the connection.

MAKING THE PRESSURE PLATE

1. Cut out 2 pieces of cardboard into rectangles both equal size of at least 20cm x 20cm (maximum width 25cm for 30cm wide foil).
2. Cut off 2 lengths of foil 10cm longer than the length of the pieces of cardboard.
3. For each piece of cardboard -

   a. Place the cardboard centrally on foil

   b. Wrap the edges of the foil around the cardboard

   c. Tape down the foil onto the cardboard.
4. Cut extra strips of cardboard that match the width of your foil covered cardboard (about 2cm wide). Cut enough strips to have
   a. a strip at each end
   b. space of 6 to 8 cm between additional strips in the middle
   c. For a 20 x 20cm size 3 strips of 2cm wide is sufficient.
5. Take one piece of foil covered cardboard, place foil fully covered side up on the top.
6. Take extra cardboard strips and place across the width of the cardboard at each end, tape down the strip over the ends of the strips. This is an insultor spacer and spring for the pressure plate.
7. Tape down extra strips across the width of the cardboard with a space of 6-8cm between each strip.
8. Place the other piece of foil covered cardboard (fully foil covered side down) onto the other parts and align, then tape together (leaving at least part of the foil edge visible).
9. Carefully slide a paperclip over the edge of each foil covered cardboard.
10. On the outside of the Pressure Plate lift up the end of each paper clip and slide onto a separate jumper lead.

TESTING THE PRESSURE PLATE

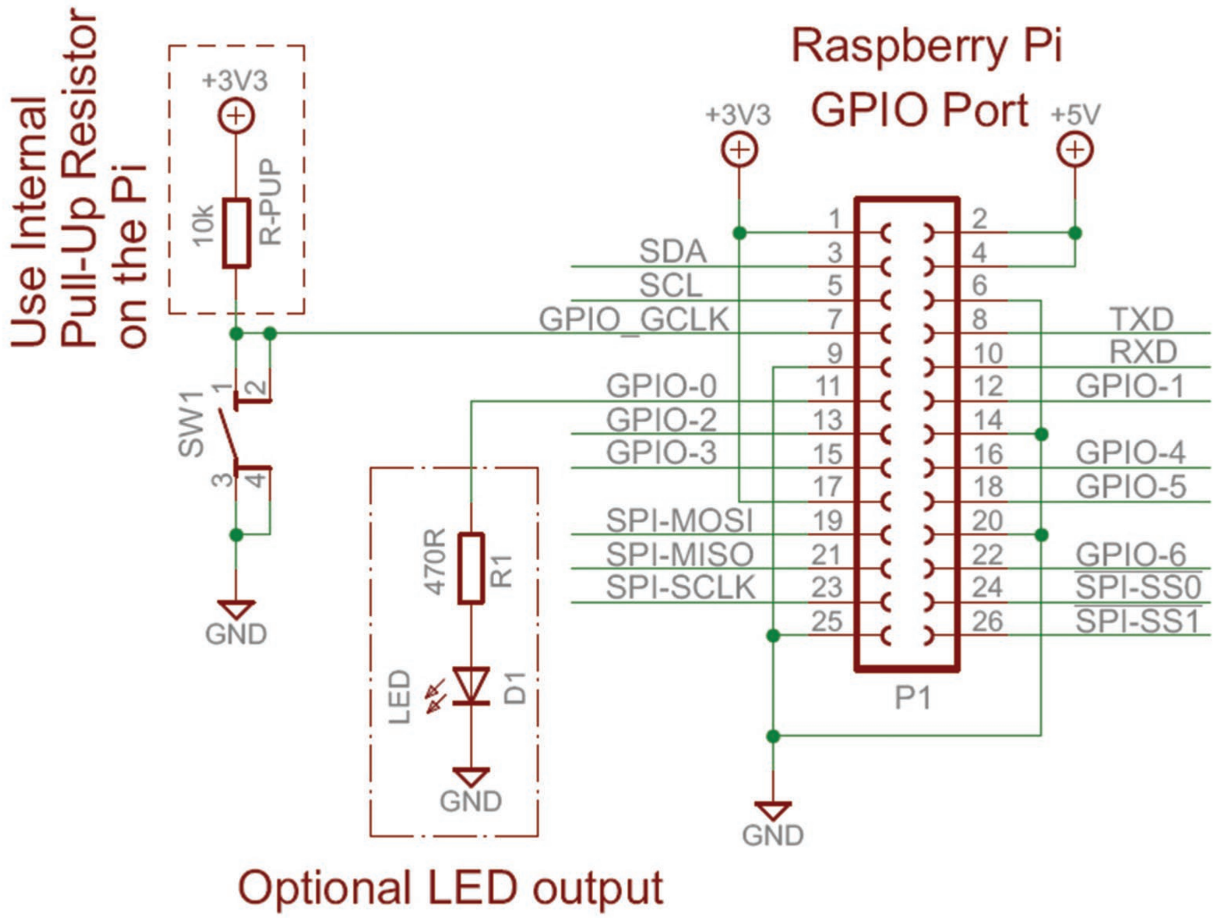
With a continuity tester or a Resistance meter, check that between the other ends of the jumper leads, the resistance is open circuit normally and short circuit when enough pressure is applied to the Pressure Plate.

The reading should go back to open circuit when pressure is removed from the Pressure Plate.

THE CIRCUIT AND CONNECTING THIS TO A PI

You will have noticed that the Pressure Plate has two states – open circuit and short circuit. Short circuit we can connect a Pi GPIO pin to a signal and the Pi will detect that signal which is good. However when open circuit the Pi wont see any signal, and in fact will see almost ANY signal as our wires will act as aerials and any old signal will appear on the Pi giving us false detections.

The clever bit is that the Raspberry Pi has Pull-up resistors inside it which can be enabled by software control. This means we only need connect two wires between the Raspberry Pi and the Pressure Plate – one to GPIO signal and one to GPIO GND.



So when we wire up a switch to a computer like Raspberry Pi, we add a resistor as in this circuit diagram, so when the switch is OFF (Pressure Plate is open circuit) the Pi will see nearly 3V3, equivalent to '1' when read by software. When the switch is ON (our pressure plate is short circuit) the Pi sees GND (0V), equivalent to '0' when read by software. The current through the switch is limited by the pull-up resistor to avoid short circuiting the 3V3 power to GND.

Take the jumper leads connected to the pressure plate, and connect them to the GPIO pins 7 and 9 as shown in the circuit diagram. As this is a switch it does not matter which way round. You might find it better to use longer wires between the Pi and the Pressure Plate so the Pressure plate can be on the floor while the Pi is on a desk or surface.

USING SOFTWARE TO DETECT THE PRESSURE PLATE

The following is a Python3 program to continuously monitor the GPIO input the Pressure Plate is connected to and display a message when the Pressure Plate is activated (someone steps on it). Save this program as *pressure.py*. The comments in the program explain the stages of the program, please read through it thoroughly.

The lines starting with # are comments and describe what is happening, all your programs should have comments to help others understand your program, and even you when you open the file in two weeks to see what is in the program. This way you can find which bits are useful or as an example for your next program.

```
#!/usr/bin/python3
# Pressure Pad Detection
# Example to loop 'forever' (Use CTRL-C to exit) doing these steps
#  1 read a pressure pad switch via the GPIO
#  2 If pressure pad switch ON (reads as 0)
#            display time stamped message on screen
#            IF connected turn a LED ON otherwise turn a LED OFF
# When
#  LED connected to GPIO bit 0 (pin 11 of P1 connector)
#   in this case a '1' output turns LED ON
#  Pressure pad switch connected to GCLK pin (pin 7 of P1 connector)
#       and GND (pin 9 of P1 connector)
#
# PC Services Pressure Pad Example
# November 2013

# import time library for delays and time stamps
import time

# Import the necessary library to access the GPIO library
import RPi.GPIO as gpio

# First configure the port number method to match GPIO connector pins
gpio.setmode( gpio.BOARD )

# Set input pin GPIO_CLK on pin 7 is connected to Pressure Pad Switch
# enable internal pull up resistor
gpio.setup( 7, gpio.IN, gpio.PUD_UP )

# Set output pin for LED off on pin 11, and initial value is OFF
gpio.setup( 11, gpio.OUT, 0 )

# Now loop forever (until someone types CTRL-C)
while( 1 ):
      # Read the input pin
      pad = gpio.input( 7 )

      # Is it ON (ON = 0)
      if pad == 0 :
            # Display message as pressure pad is on
            print(  "{} - Pressure Pad activated".format( time.asctime() ) )
            # set the output to '1' to turn LED on
            gpio.output( 11, 1 )
      else :
            # Otherwise set the output to '0' to turn LED off
            gpio.output( 11, 0 )

      # now wait a short while, to make sure we can see a real Pressure pad event
      time.sleep( 0.2 )
```

RUNNING THE PROGRAM

When you have the program on the Pi, you need to run the program in a special way as accessing the GPIO pins requires special permissions as follows, at the command prompt or in a terminal window:

```
sudo python3 pressure.py
```

What this command breaks down into

'sudo' tells the Pi to run this command with special permissions (like on Windows systems 'Run as Administrator')

'python3' is the command to run Python3 on the Pi

'pressure.py' is the filename for Python3 to read to run your program

Every time the Pressure Plate is activated you will see a message displayed on the display.

FURTHER EXPERIMENTATION

Now you have a working Pressure Plate, try and find out what the minimum weight is for the Pressure Plate to activate and where it must be placed.

EXTENDING THIS EXAMPLE

From the software you can see a message is displayed when the Pressure Plate is activated but we can also do other things at the same time, the easiest addition you can do without changing the software is to attach an LED as in the previous diagram and that will light when the Pressure Plate is activated.

Other things you could do to extend the software are –

- Log the activation times to a file on the Pi
- Log the time of activation and how long the Pressure Plate is activated for
- Call a program to send a text message or email that the Pressure Plate is activated
- Change the LED output to a siren.

TO USE THE LED OUTPUT

Connect Jumper Leads as follows

1. Pin 11 of GPIO connector to one side of 470R Resistor
2. Other side of 470R Resistor to Anode of LED
3. Cathode of LED to Pin 25 of GPIO connector.

If the LED does not light, check your wiring is connected to correct pins and firmly, then also try swapping the LED round.

This resource has been created from the winning competition entry for a Raspberry Pi recipe idea by Elliot Fenwick.