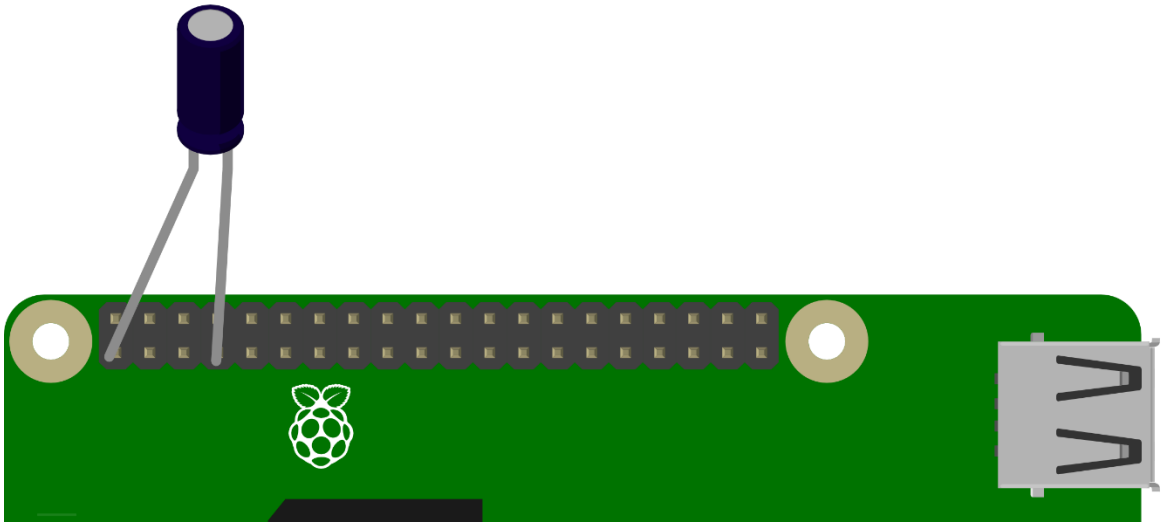
Shakey Shakey. A game all about shaking

RASPBERRY JAM

3.3V	1	2	5V		
2 SDA1	3	4	5V		
3 SCL1	5	6	GND		
4 GCLK	7	8	14	TXD0	
GND	9	10	15	RXD0	
17	11	12	18		
27	13	14	GND		
22	15	16	23		
3.3V	17	18	24		
10 MOSI	19	20	GND		
9 MISO	21	22	25		
11 SCLK	23	24	8	CE0_N	
GND	25	26	7	CE1_N	
ID_SD	27	28	ID_SC		
5	29	30	GND		
6	31	32	12		
13	33	34	GND		
19	35	36	16		
26	37	38	20		
GND	25	40	21		

Getting started

1. Before turning on the Raspberry Pi wire up one of the Tilt sensors. It does not matter wh way around the wires go. Connect them to 3.3V (pin 1) and GPIO4 (pin 7)



2. Time to connect the power to the Raspberry Pi and boot up.

3. Run Python 3 and open a new file [File]-[New].

   Save the file and call it shakey.py

   You could use any name but it makes it easier for the workshop if everyone is the same

Code to read the sensor

1 Let's start coding

```
# Starting code to read the tilt sensor and increase score
# import the RPi.GPIO library so you can control the GPIO pins on the Raspberry Pi
import RPi.GPIO as GPIO

# Set up to use BCM numbering
GPIO.setmode(GPIO.BCM)

# Set GPIO 4 as input with pulldown resistor
GPIO.setup(4,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

#set the score to 0 at the start
score=0

# while True means do forever
while True:
    # if connection made increase score by 1 and display it
    if GPIO.input(4):
        score=score +1
        print("Your score is: " + str(score))
```

We are using the RPi.GPIO library to read GPIO 4. This is a more full featured library.

After saving the program and running it what happens when you shake the sensor?

Oh, so it doesn't just give you one score each time you shake it. It gives you loads… What's happening?
What is it is the little metal ball inside that is making the connection is bouncing, so more than one connection is being detected. Computers work so fast that it can detect this multiple connections.

So, we need to sort that out with a bit of a change to the program.

2 Update your program as follows. New lines/code are highlighted in **blue** to fix the fact you get lots of scores for each shake

```
# Starting code to read the tilt sensor and increase score
# Adding in slight pause to do de-bounce

# import the RPi.GPIO library so you can control the GPIO pins on the Raspberry Pi
import RPi.GPIO as GPIO

# import sleep command that lets you do a pause/wait
from time import sleep

# Set up to use BCM numbering
GPIO.setmode(GPIO.BCM)

# Set GPIO 4 as as input with pulldown resistor
GPIO.setup(4,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

#set the score to 0 at the start
score=0

# while True means do forever
while True:
    # if connection made increase score by 1 and display it
    if GPIO.input(4):
        score=score +1
      # this little wait manages the debounce
        sleep(0.05)
        print("Your score is: " + str(score))
```

Adding a second senor

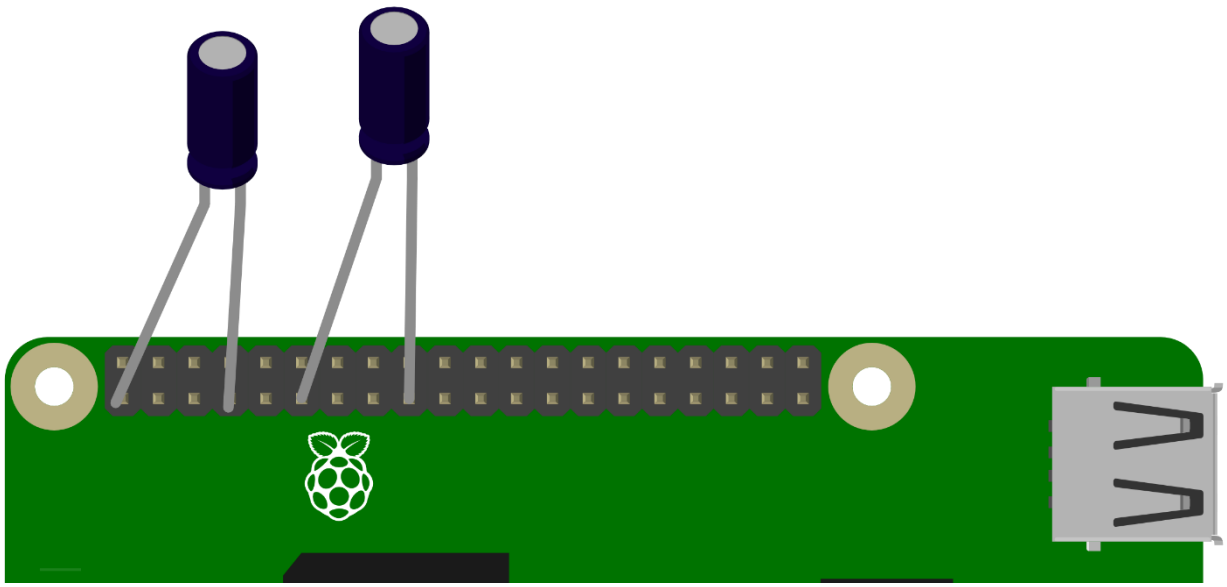
That's better. The important bit is the sleep(0.05). This waits for one twentieth of a second when a shake it detected. This slows down the computer so it does not record the bouncing. Chances are you are not able to shake the sensor 20 times a second.

What you did is add debounce ability to you program. This is often needed with mechanical switches.

Have another go at running the program and shaking the sensor. See how many points you can get.

That was fun. How about we add a second tilt sensor. Before adding any electronics you have to shutdown the Raspberry Pi as connecting components wrongly can cause the Raspberry Pi to fail.
So, shutdown the Raspberry Pi and disconnect the power cable.

4 Add a second tilt sensor to GPIO 17 (pin 11) and a second 3.3V (pin 17)



5 Adding the code to read the second sensor.

```
# Starting code to read the tilt sensor and increase score
# Adding in slight pause to do de-bounce
# Adding in second tilt sensor

# import the RPi.GPIO library so you can control the GPIO pins on the Raspberry Pi
import RPi.GPIO as GPIO

# import sleep command that lets you do a pause/wait
from time import sleep

# Set up to use BCM numbering
GPIO.setmode(GPIO.BCM)

# Set GPIO 4 as as input with pulldown resistor
GPIO.setup(4,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
# Set GPIO 17 as as input with pulldown resistor
GPIO.setup(17,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

#set the score to 0 at the start
score = 0

while True:
    if GPIO.input(4):
        score =score +1
        sleep(0.05)
        print("Your score is: " + str(score))

        # stay in this while loop until GPIO17 is High
        while not GPIO.input(17):
            sleep(0.05)
```

Not a massive change. You can see where we add the second tilt sensor on GPIO17. Then the lines at the end.

The following line is interesting as it uses a ‘not’ statement. This means when it’s the opposite. So, this line say while sensor 17 is not high (connected) stay in the loop.

What this means is after you activate sensor on GPIO 4 it waits until you activate the sensor on GPIO 17 before it will detect GPIO 4 again. So, you have to shake the sensors one at a time. Handy you have two hands to hold the sensors so you can share one in one hand and the second sensor in your other hand.

Give it a go and see what happens. How fast are you?

This is all good fun but it’s really a game. With a game you have to be able to tell who’s done the best. The program so far goes on forever. What we need is to give the program a time limit so we can see who shakes the sensors the most in a specific time.

To achieve this we will need to add a timer. So, the next update to the code has a fair few changes.

Making it into a game

6 Time to update the code one last time to make this into a game that you can challenge other to play and see who can get the highest score.

```
# Starting code to read the tilt sensor and increase score
# Adding in slight pause to do de-bounce
# Adding in second tilt sensor
# Add timer to make it a game

# import the RPi.GPIO library so you can control the GPIO pins on the Raspberry Pi
import RPi.GPIO as GPIO

# import sleep and time commands that lets you do a pause/wait and get the time
from time import sleep,time

# Set up to use BCM numbering
GPIO.setmode(GPIO.BCM)

# Set GPIO 4 as as input with pulldown resistor
GPIO.setup(4,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
# Set GPIO 17 as a second input with pulldown resistor
GPIO.setup(17,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

#set the score to 0 at the start
score = 0

# Get the current time as a marker
starttime = time()

# Set how long a game is for (15 seconds)
duration = 15

# Set the time the game ends
endtime = starttime + duration

# Give the player time to get ready
print("READY!")
```

```
sleep(1)
print("STEADY!")
sleep(1)
print("GO!!!!!!!")
sleep(0.5)

#While current time is less than start time + the duration (15 seconds)
while time() < endtime:

    # display the score and the amount of time left
    print("Your Score is: " + str(score) + " -  Time left: " +  str(endtime - time()))
    # if connection made increase score by 1
      if GPIO.input(4):
        score =score +1
        sleep(0.05)

        # stay in this while loop until GPIO17 is High or the time runs out
        while (not GPIO.input(17)) and (time() < endtime):
            print("Your Score is: " + str(score) + " -  Time left: " +
str(endtime - time()))

# Final score
print("Your Final Score is: " + str(score))
```

That is a big chunk of code. Now the game lasts for 'duration' seconds so you can check to see who can do the most shakes in 15 seconds.

Try changing the value of duration to change the length of the game.

Challenge – use this input for your own game

Hint: Maybe a game like Angry Birds or Pong could use these as the controller