Star Letter: An FET Buffer Stage for GPIO Access

In response to the In Control article from Issue 4, Clive Tombs shares his own example of connecting to GPIO pins.

Introduction

Following on from the issue 4 article on transistors, I would like to describe my use of the 2N7000 Enhancement FET. I used this device only because I had some on hand from previous projects. Other types could be better suited as I will explain later.

Their use provides some interesting behaviours to buffer circuits which may prove beneficial in some applications.

The data sheet can be found here: http://pdf1.alldatasheet.com/datasheet-pdf/view/2842/MOTOROLA/2N7000.html

Now, the FET's Gate is, in simplistic terms, insulated from the Source and Drain connections. Only the voltage relative to the Source (Vgs) is important. Once again I state in simplistic terms. Even if the GPIO pin is configured as an INPUT with the Pi's own Pull Up or Down resistors active, the FET will change state due to the extremely high input impedance of the FET.

From the data-sheet it can been seen that at around Vgs of 2.5v at room temperature the device starts to conduct. By 3.3v it can certainly operate an LED or small relay. As I stated above other FETs may be more suitable in their Vgs characteristics.

Now consider the following application: Test all inputs at start-up. Very simple code can be written to test all used inputs at start-up. By pulling the inputs up then down and testing for the condition in software and visually for an LED flash one can verify both the wiring and the buffer FET. This may seem trivial, but if the LED were replaced with the start circuit for some equipment

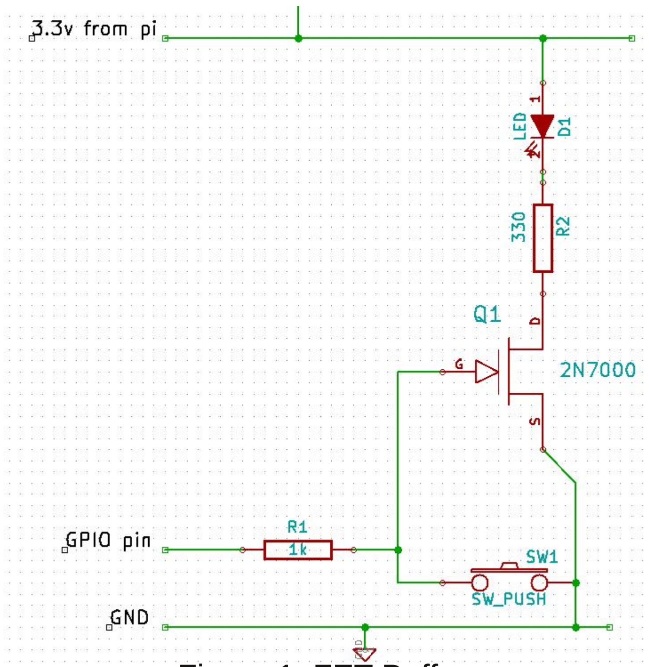

Figure 1: FET Buffer

which must be started in a correct sequence, this code would eliminate the FET as a source of error. As a maintenance engineer I like diagnostics to make my life easier!

It also has the advantage that one GPIO can be used for both input and output with, in Fig 1's case, a visual indication of button press too.

This is my first ever stab at a Python script. It is bound to be very inelegant, but it just about does what we need. It has been tested in Python 3 only. Try running it with a finger on the button to simulate an input being stuck.

Of course one could arrange the switch to pull the input up. That way the LED would not be on all the time. Script adjustments will be necessary.

With a change in resistor values the FET status can remain unchanged if the button is pressed when the GPIO is set as output.
Eg: if R1 is 330Ω and the switch is connected through about 4k7Ω the Vgs will still be in excess of 3.0v with the button pressed if GPIO pin is output set high.

2N7000s are available for 10p each. Other, superb devices are now available. Some like the 2SK4043LS can switch pulses of 80A with as little as 2.5v Vgs. A single transistor could never do that as driven by the PI. And the 2SK3018, a surface mount device designed for small Vgs conditions like here in the PI.

```
#input test with visual indication
import RPi.GPIO as GPIO
import time

# change to BCM GPIO numbering
GPIO.setmode(GPIO.BCM)

# (pull_up_down be PUD_OFF, PUD_UP or
# PUD_DOWN, default PUD_OFF)
GPIO.setup(4, GPIO.IN,
pull_up_down=GPIO.PUD_UP)

# test for pin able to go high
if GPIO.input(4):
    print ('Input True Good')
    time.sleep(0.2)
    GPIO.setup(4, GPIO.IN,
         pull_up_down=GPIO.PUD_DOWN)
else:
    print ('Fault Input - pin4')
    time.sleep(1)
    quit()

# test for input able to go low
if GPIO.input(4):
    print ('Faulty Input - pin4')
    time.sleep(1)
    quit()
else:
    print ('Input False Good')
    time.sleep(1)

# if it gets to here, inputs' states
# are both achievable
print ('Inputs tested Good')

# commence the button demo
print ('Press the Button')

while True:
    # set pin high and
    # wait for button press
    GPIO.setup(4, GPIO.IN,
            pull_up_down=GPIO.PUD_UP)

    # button pressed
    if not GPIO.input(4):
        print ('button pressed')
        time.sleep(1)

        # button released
        if GPIO.input(4):
            print ('button released')
            flash = 20
            GPIO.setup(4, GPIO.OUT)

            # flash until
            while flash > 0:
                GPIO.output(4, True)
                time.sleep(0.1)
                GPIO.output(4, False)
                time.sleep(0.1)
                flash -= 1

            print ('press button')
```

There is a lot to be said for the FET in this application.

Clive Tombs

Editors Note

We love hearing from our readers. If you have a comment on an article or a cool Raspberry Pi related tip to share, please send it to us and we'll try to get it in an upcoming issue.

DID YOU KNOW?

The "In Control" series in issues 2, 3 and 4 is a great place to start and learn how to use the GPIO. If you have not started yet but want to have a go, there have been some updates to the RasPi GPIO Python library that you need to know before starting.

1) The RasPi GPIO library can now be easily installed with:

```
$sudo apt-get install python-rpi.gpio
```
or
```
$sudo apt-get install python3-rpi.gpio
```

2) Add the following line to each program:
```
import RPi.GPIO as GPIO
```