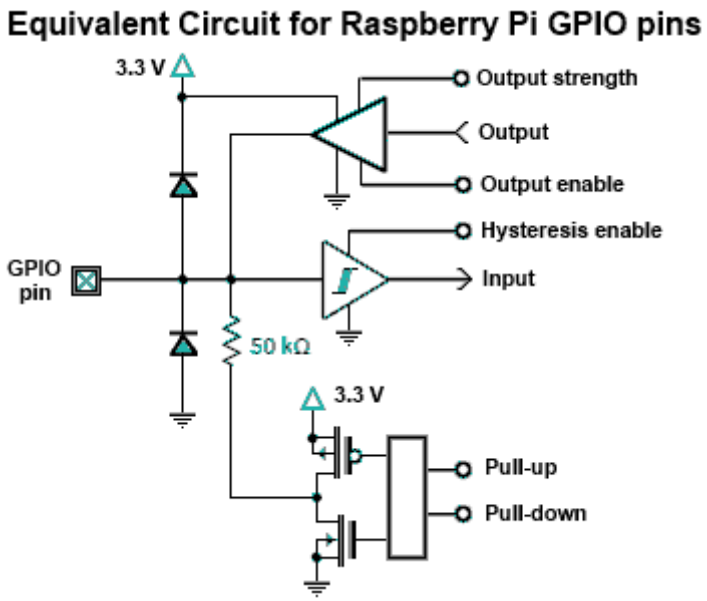
Part 75
-
Interfacing

Version: 2021-02-28

Interfacing with Raspberry Pi

The equivalent circuit of a GPIO pin from the Raspberyy Pi looks like this



You should keep the following limitations in mind when using the GPIO pins

- These are 3.3 volt logic pins. A voltage near 3.3 V is interpreted as a logic one while a voltage near zero volts is a logic zero. A GPIO pin should never be connected to a voltage source greater than 3.3V or less than 0V, as prompt damage to the chip may occur as the input pin substrate diodes conduct. There may be times when you may need to connect them to out-of-range voltages – in those cases the input pin current must be limited by an external resistor to a value that prevents harm to the chip. I recommend that you never source or sink more than 0.5 mA into an input pin.
- To prevent excessive power dissipation in the chip, you should not source/sink more current out of the pin than its programmed limit. So, if you have set the current capability to 2 mA, do not draw more than 2 mA from the pin.
- Never demand that any output pin source or sink more than 16 mA.
- Current sourced by the outputs is drawn from the 3.3 V supply, which can supply only 50 mA maximum. Consequently, the maximum you can source from all the GPIO outputs simultaneously is less than 50 mA. You may be able to draw transient currents beyond that limit as they are drawn from the bypass capacitors on the 3.3 V rail, but don't push the envelope!
- There isn't a similar limitation on sink current. For sink current, the pertinent limitation is that imposed by maximum chip power dissipation. Even so, you can safely sink up to 16 mA each into any number of GPIO pins simultaneously. In the worst case, the output pins (if configured to the 16mA high current drive capability) have a maximum output low voltage of about 0.4 V, and and their internal circuitry dissipates only 6.4 mW worst case. Even sinking 16 mA into 16 pins simultaneously would produce only 0.1024 W, that is, about a tenth of a watt. However, depending on the source of the current, transient sink currents may make demands on the board's bypass capacitors, so you may not want to switch all the outputs to sink the maximum current synchronously, if you require fast, clean transitions.
- Do not drive capacitive loads. Do not place a capacitive load directly across the pin. Limit current into any capacitive load to a maximum transient current of 16 mA. For example, if you use a low pass filter on an output pin, you must provide a series resistance of at least 3.3V/16mA = 200 Ω.

And listed here are the electrical characteristics

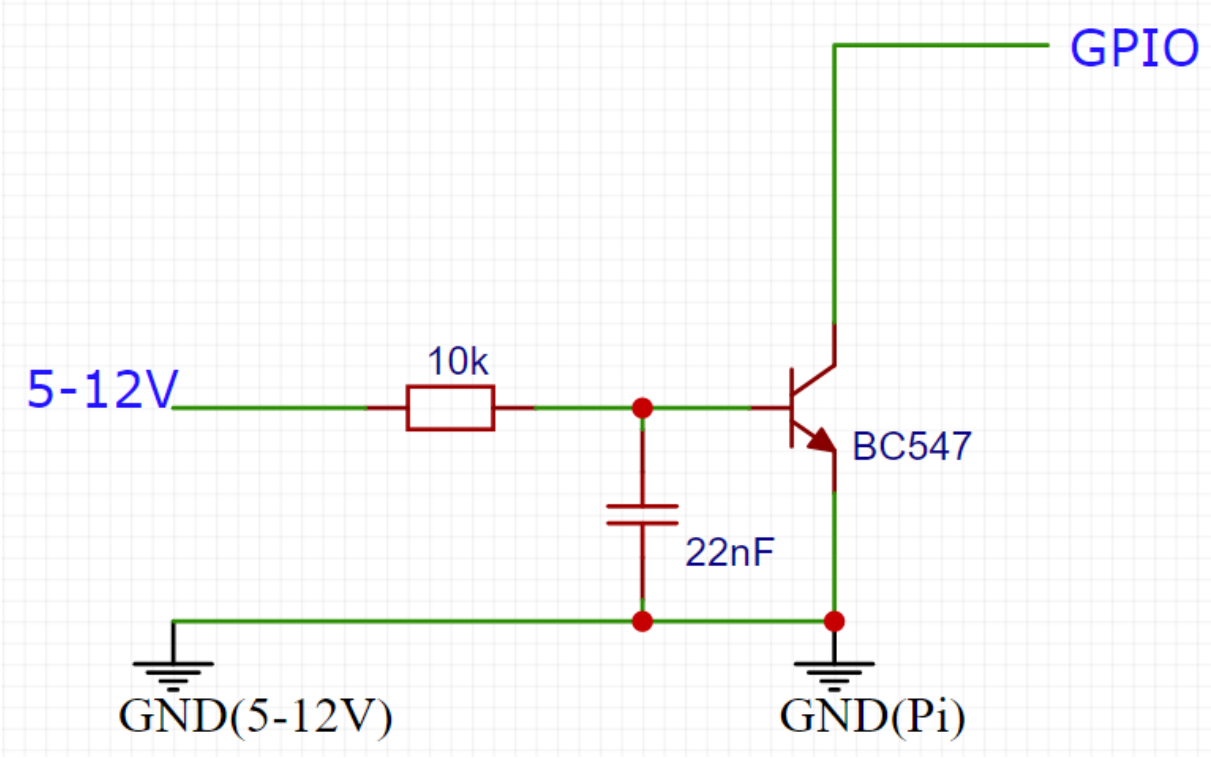
GPIO input/output pin electrical characteristics	
Output low voltage V_{OL}	< 0.40 V < 0.66 V < 0.40 V < 0.40 V
Output high voltage V_{OH}	> 2.40 V > 2.64 V > 2.90 V
Input low voltage V_{IL}	< 0.80 V < 0.54 V < 1.15 V
Input high voltage V_{IH}	> 2.00 V > 2.31 V > 2.15 V
Hystereses	> 0.25 V 0.66 – 2.08 V
Schmitt trigger input low threshold V_{T-}	1.09 - 1.16 V 0.9
Schmitt trigger input high threshold V_{T+}	2.24 - 2.74 V 0.90 V
Pull-up/down resistance	40 – 65 kΩ 100 kΩ
Pull-up/down current	< 50 µA < 28 µA
Pin capacitance	5 pF
Bus hold resistance	5-11 kΩ

In order to interface with the outside world like sensor modules, relays etc, you will need some circuitery to adapt the voltage and current levels from or to the outside towards the Pi. There are frequently called 'levelshifters'. A level shifter will change the voltage level according to the circuits' power lines. E.g. the level shifter will change between 3.3v and 5v.

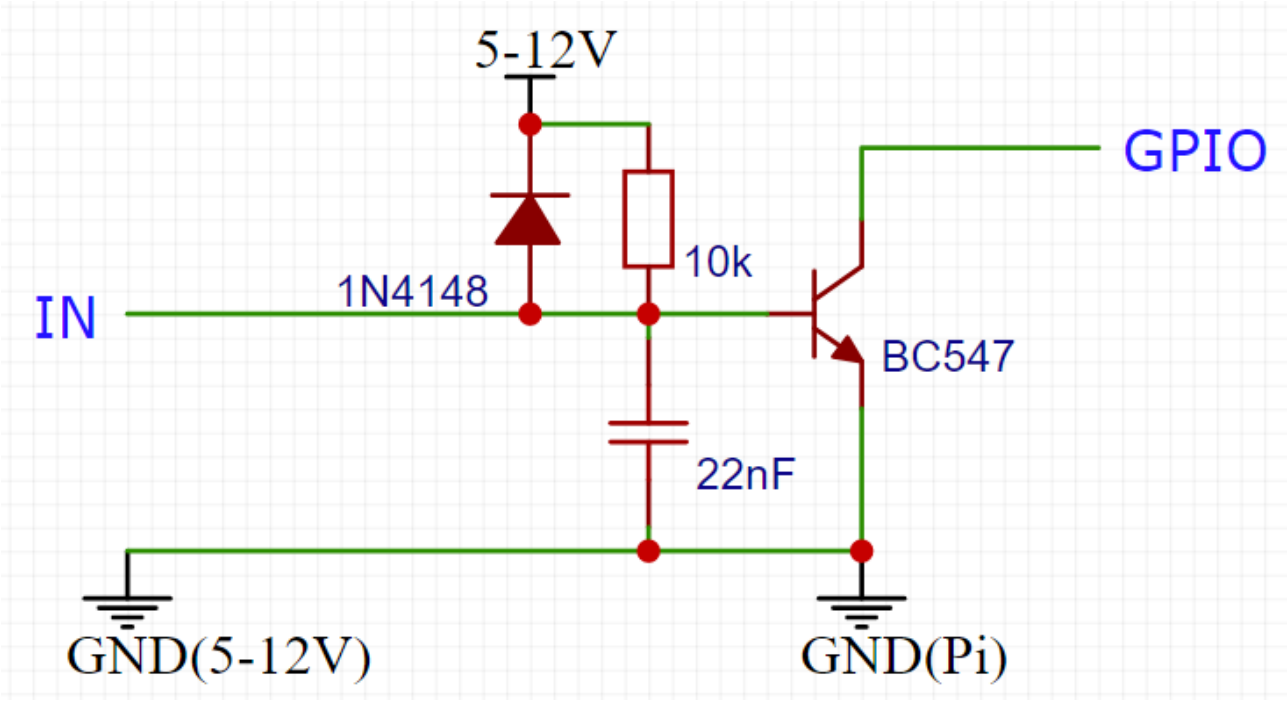
They can be unidirectional, but also bidirectional. I'll list some of the most important interface circuits here.

Unidirectional levelshifters but inverting the logic !!!

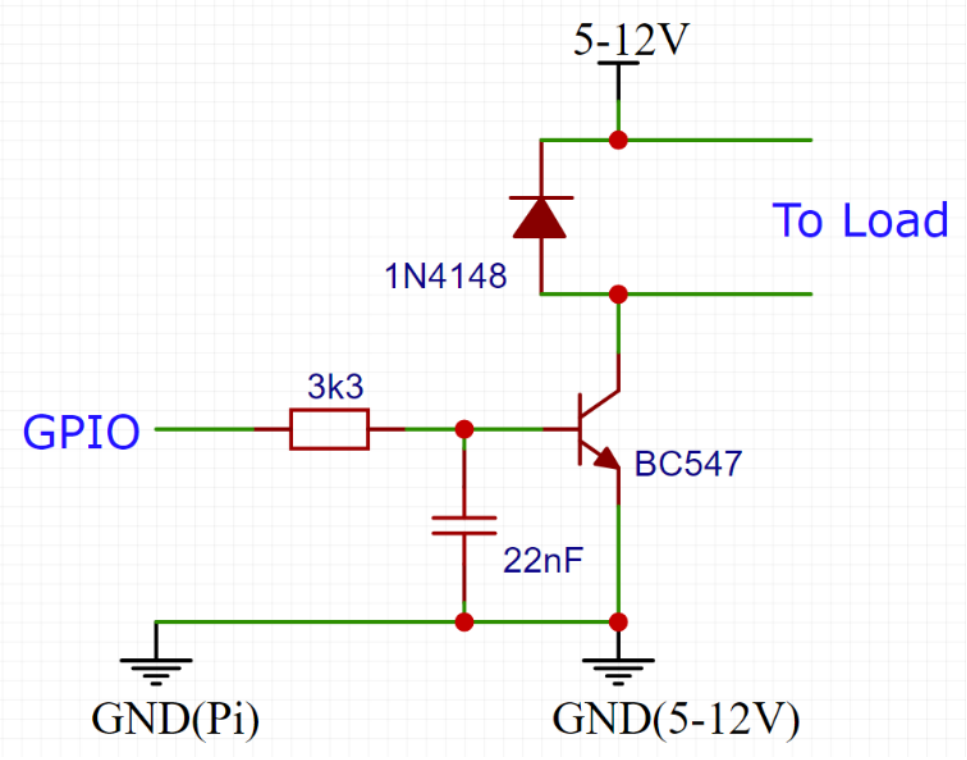
Input voltage provide by input device. Eg: when the outside sensor provides a high level of eg: 10V in the IN, the GPIO pin will see a LOW-level.
Note: make sure to use internal GPIO pull-up circuit.



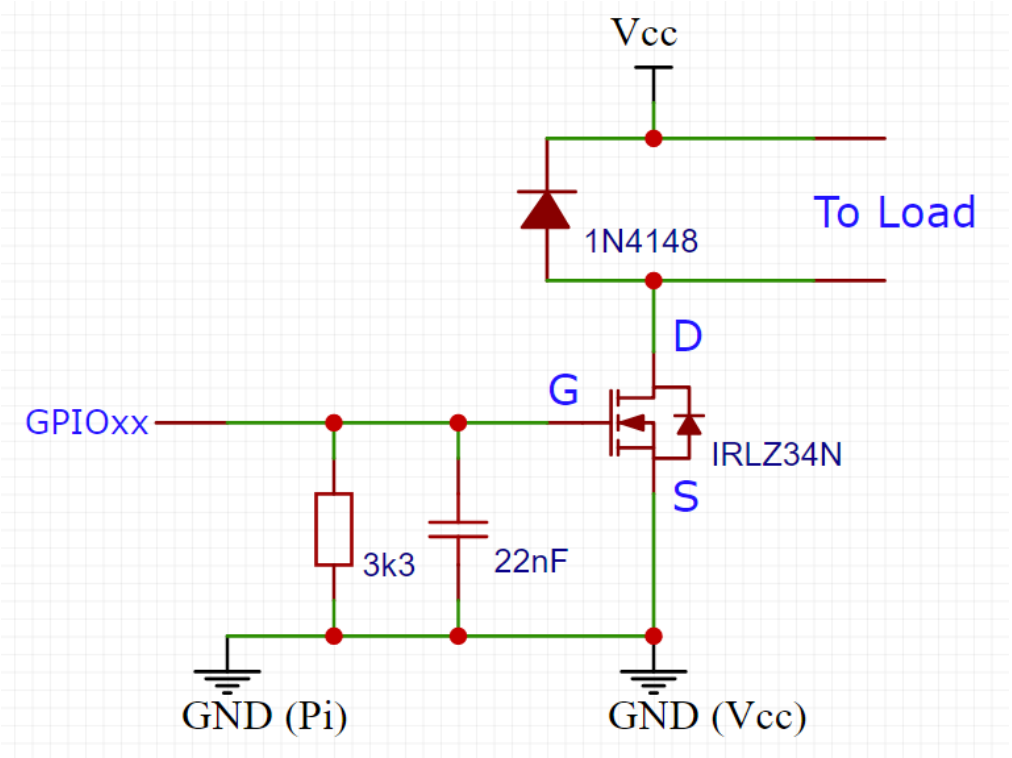
Input voltage provided by levelshifting circuit. Eg: sensor has an open collector or relay output. When the outside sensor pulls the IN line down, the GPIO pin will see a HIHG-level.
Note: make sure to use internal GPIO pull-up circuit.



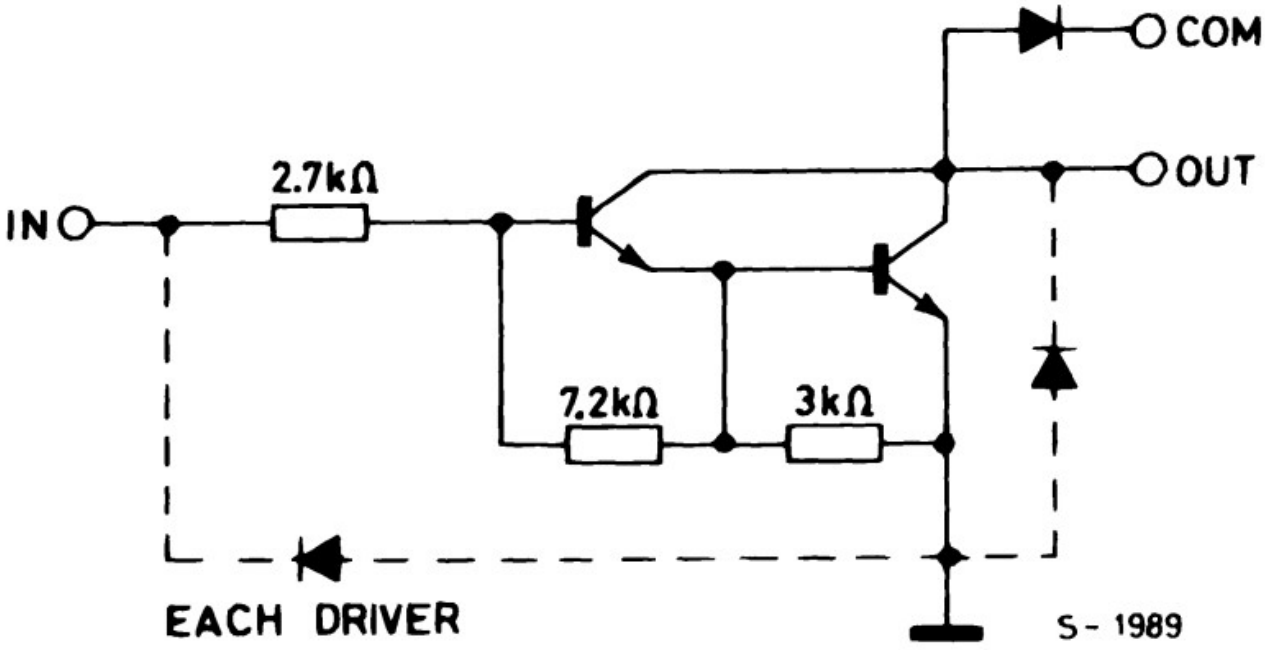
When the GPIO is set to HIGH, the LOAD will be powered.
Note: make sure to use internal GPIO pull-up circuit. This circuit can handle up to 100mA.
Using another transitors like a darlington eg. TIP132, more current can be handled.



Instead of using classic transistors, you can use MOSFETs as well.
Using MOSFET 2N7000 will allow you to drive up Vcc to 60V and 100mA (@ Vgs = 3.3V) while using a IRLZ34N will even boost the current to 10A (@ Vgs = 3.3V)



If you need several GPIO pins to be interfaced, you might consider using an IC instead of discrete components
Eg: the ULN2003 has 7 of these drivers onboard capable of driving 50 V and 500mA per driver.

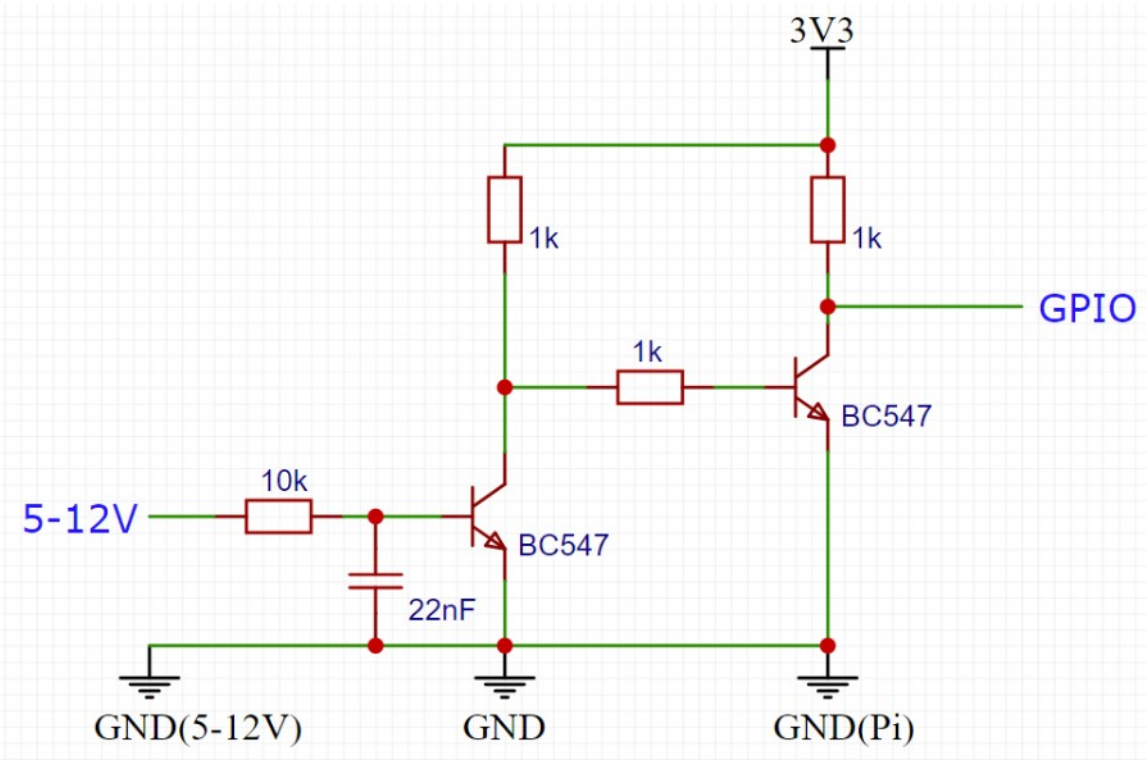


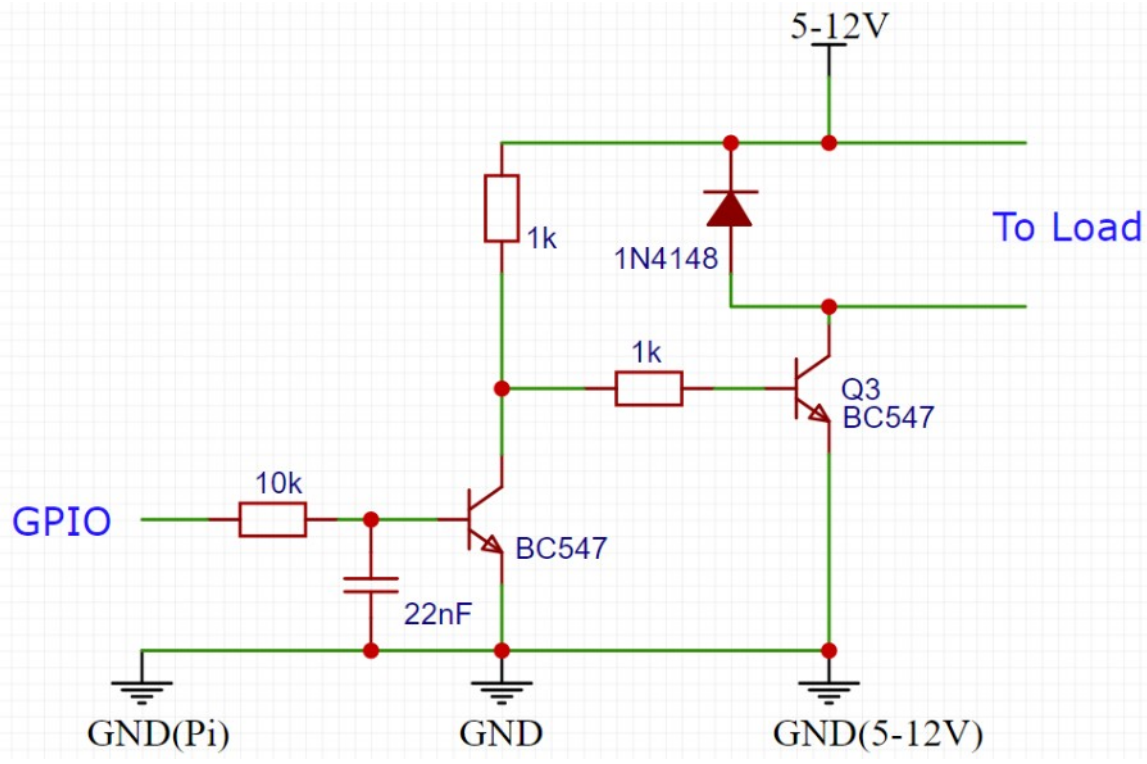
Some additional information:

- The 22nF capacitor is used to suppress all high frequenty signals that might disturb the circuit. Hence also using the GPIO internal pull-up resistor is used for the sam e reason however is much more effective for low frequency signals eg: 50 or 60Hz from the electrical net.
- The 1N4148 multipurpose diode is used as a 'fly-back'-suppressor when inductive loads are used, 'killing' the reverse voltage when switching off inductive devices like relays.
- All resistors are 1/4W

Unidirectional levelshifters

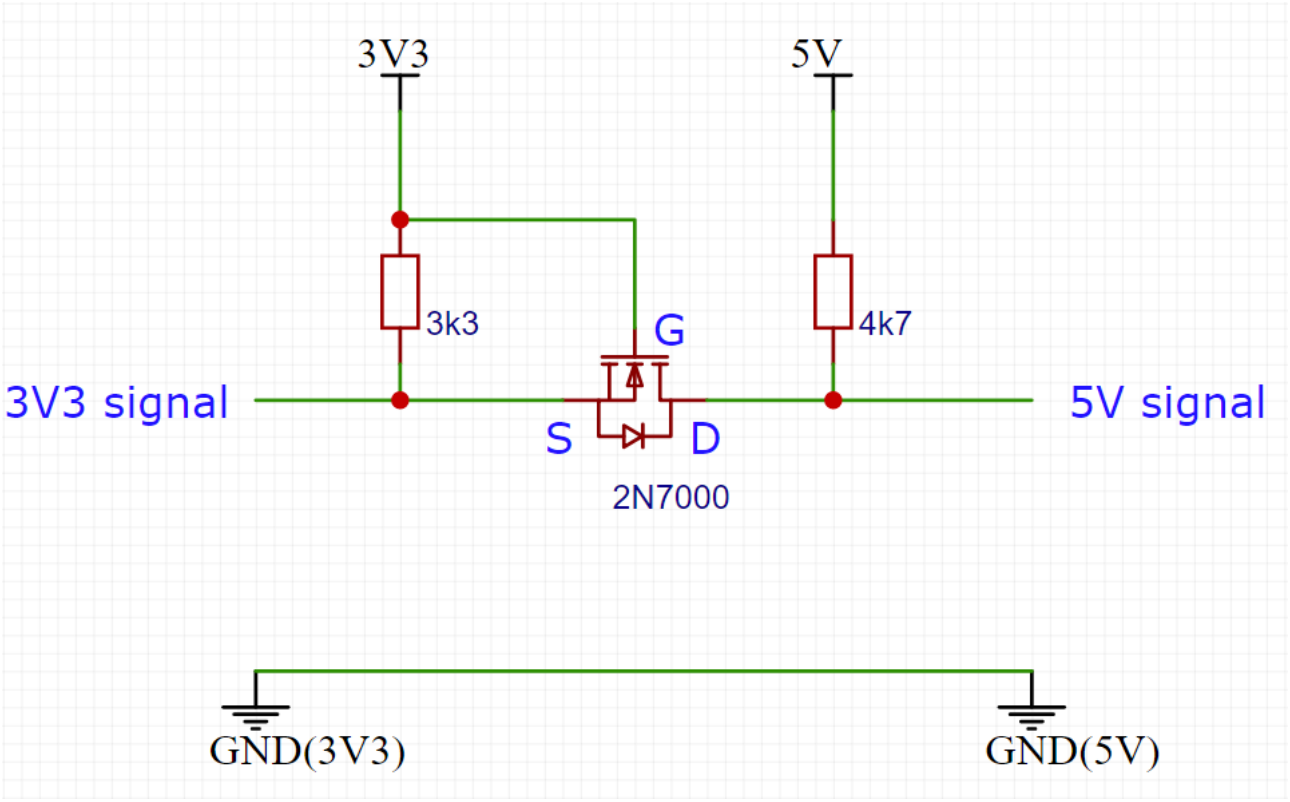
Here you see an input and an output circuit which does not inverted the logic. A HIGH in is a HIGH out.
Note: make sure to use internal GPIO pull-up circuit. This circuit can handle up to 100mA. Using another transitor for Q3 like a darlington eg TIP132, more current can be handled.





Bidirectional level shifter

For sensor circuit modules 5V is the most common voltage. So the interfaces used like I2C, SPI and serial are mostlikey 5V driven/levelled. Also USB is still a 5V protocol. These protocols are often bidirectional. The solution to stepping down and stepping up a voltage is to use a bi-directional level shifter. It's a very simple set up that uses a MOSFET and two resistors. The schematic is below.



Note that the Gate is always connected to the LOWER voltage. Any devices that share a signal line but are on two different voltages can use this set up. The level shifter shown above is adapted for I2C devices working on different voltages. The resistors would then take the values of the corresponding pull-up resistor for the I2C line, usually 4.7K on the 5v side and 3.3K on the 3.3v side. This will provide a suitable current of 1mA. Be careful here, don't have two sets of pull-up resistors on the same line, if you already have them installed somewhere else then don't include them again.
10K resistors on either side work well with serial data.

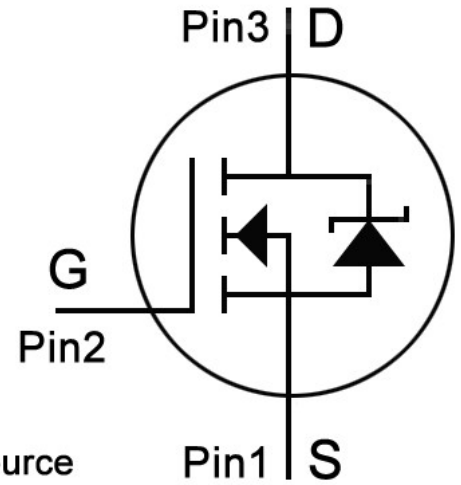
Again, if several GPIO ports need to be adjusted, consider using an IC or modules instead of discrete components.

Eg: Texas Instruments TXB0108PWR or Adafruit 8-channel Bi-directional Logic Level Converter, Adafruit

2N7000/2N7002

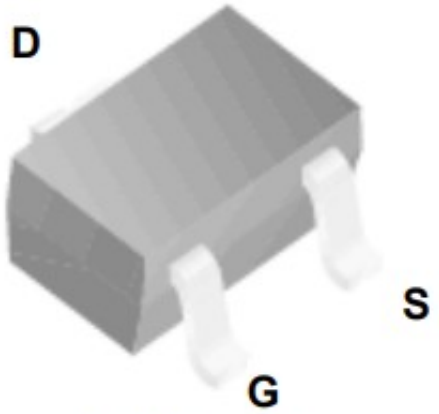
TO-92 Package

N Channel
Mosfet

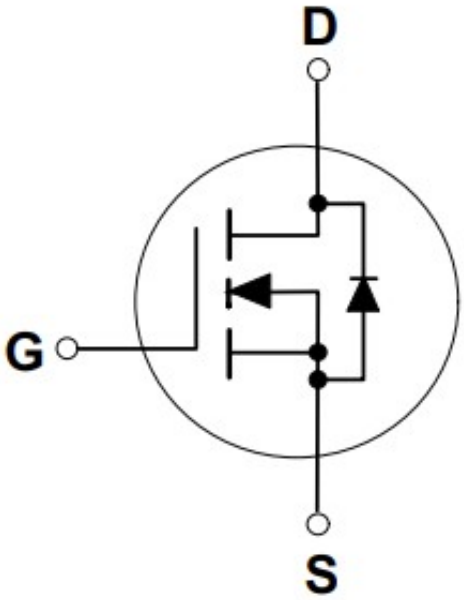


Pin1 S
Pin3 D
G Pin2

S = Source
G = Gate
D = Drain

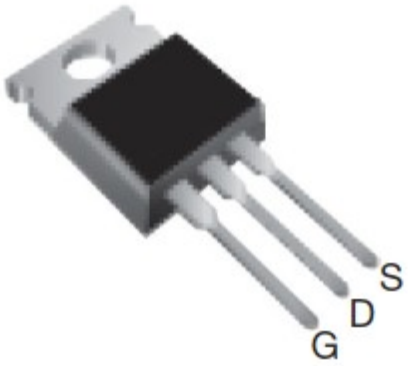


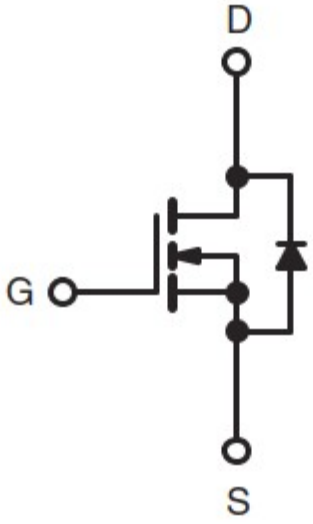
SOT−23
(TO-236AB)
2N7002/NDS7002A



IRLZ34N

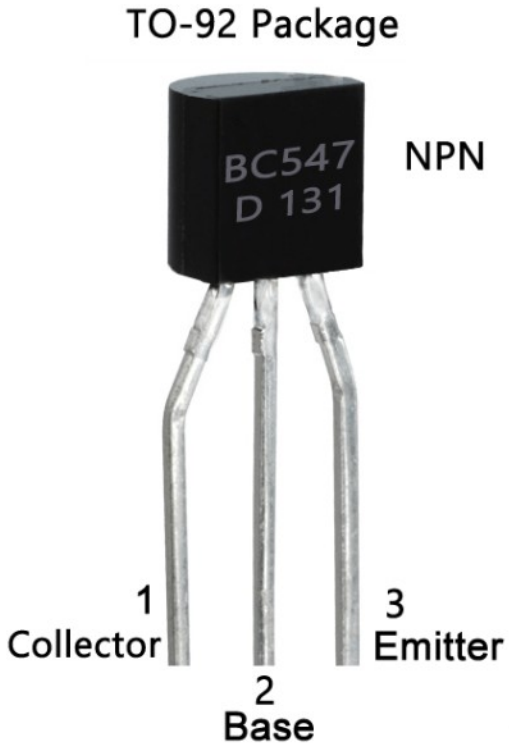
TO-220AB

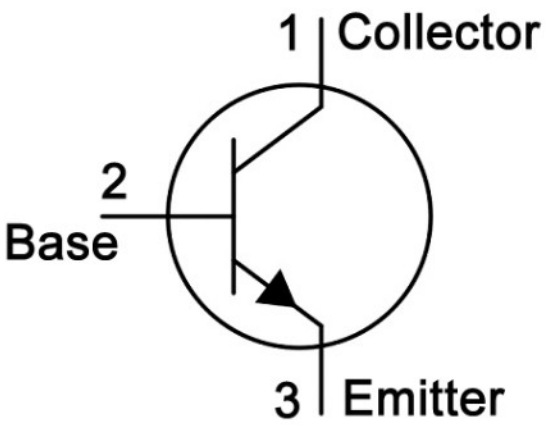




N-Channel MOSFET

BC457





TIP132