Retrace. . .

GPIO. . . Seems to be more used in the context of microcontrollers and SoC systems, especially when discussing hardware prior to configuration.

Parallel Port. . . Seems to be more used to discuss “a set of GPIO slices.”

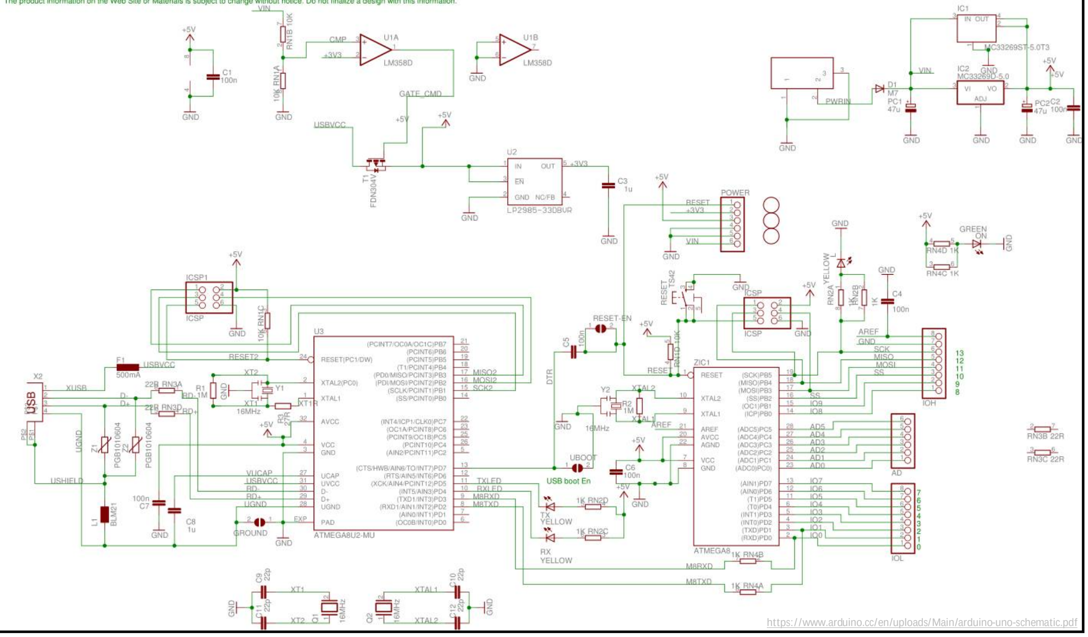
1

The Arduino Uno schematic is “open source.” Anybody can download it and inspect it.

Arduino™ UNO Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

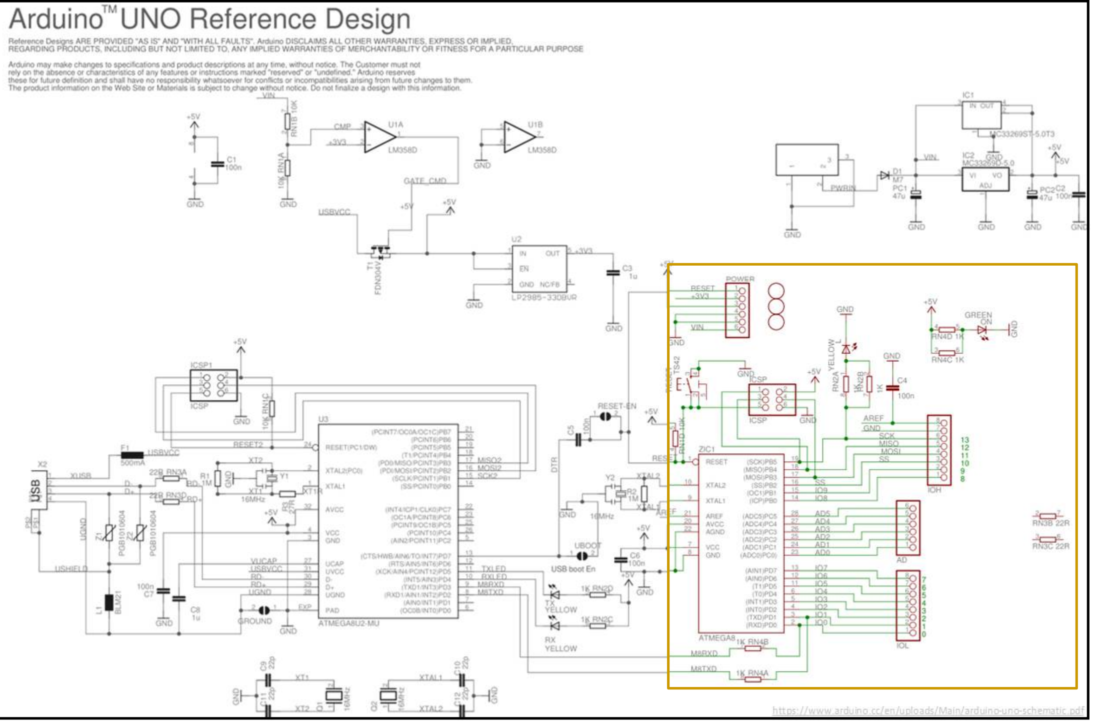


https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

2

The Arduino Uno schematic is "open source." Anybody can download it and inspect it.

Let's zoom in on the ports.



Arduino™ UNO Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

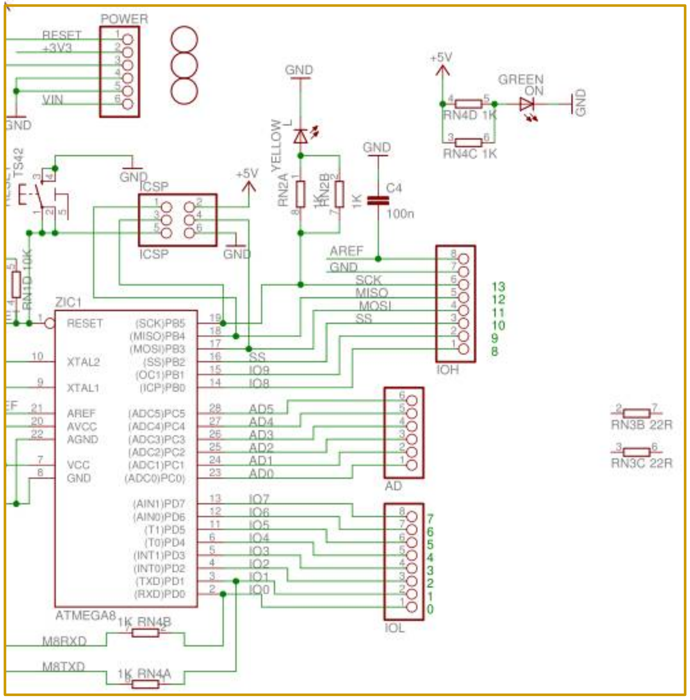
The Arduino Uno schematic is "open source." Anybody can download it and inspect it.

Let's zoom in on the ports.

Arduino's "Digital I/O pin 13" is The ATmega's "(SCK)PB5" pin.

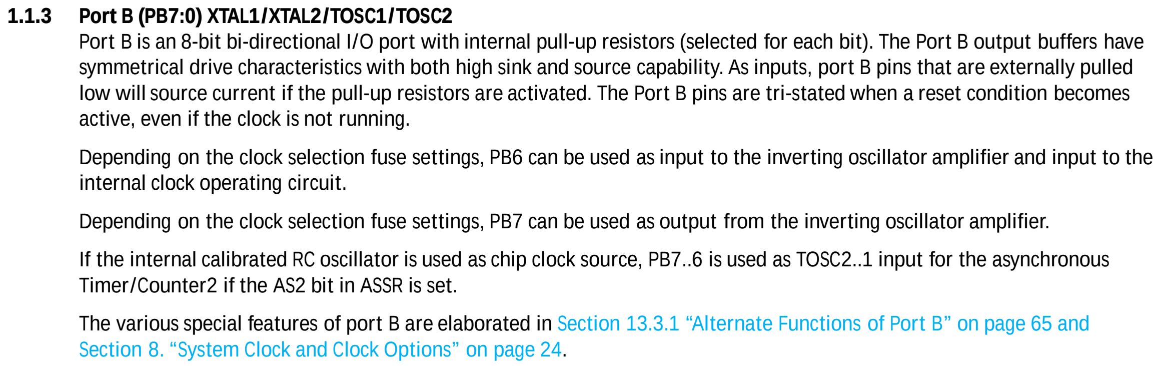
This pin can be configured as part of an 8-bit parallel port, "Port B" in the ATmega's datasheet, or as the clock output of a "Serial Peripheral Interface" (SPI) port

Configuring that pin as an output gives up the possibility of a hardware-supported SPI port.



https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

The description of the “Parallel Port B” in the ATmega’s datasheet



1.1.3 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting oscillator amplifier.

If the internal calibrated RC oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

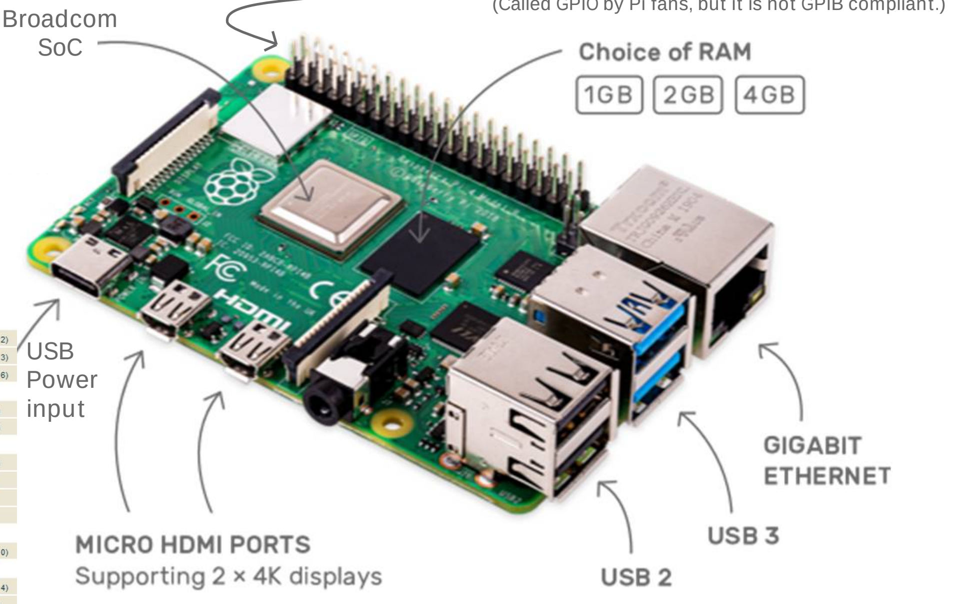
The various special features of port B are elaborated in Section 13.3.1 “Alternate Functions of Port B” on page 65 and Section 8. “System Clock and Clock Options” on page 24.

There is a ton of detail available for the Arduino products!
The datasheet is essential for making sense of the schematic.

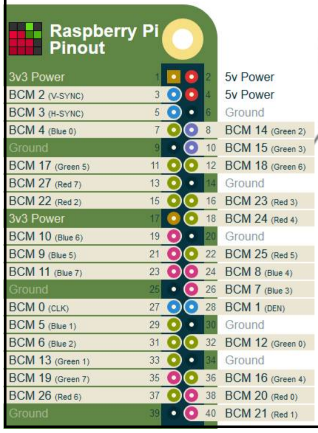
5

Where is this parallel I/O hardware on the Raspberry Pi?

Parallel I/O header
(Called GPIO by Pi fans, but it is not GPIB compliant.)



https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
https://pinout.xyz/pinout/dpi#



Raspberry Pi Pinout

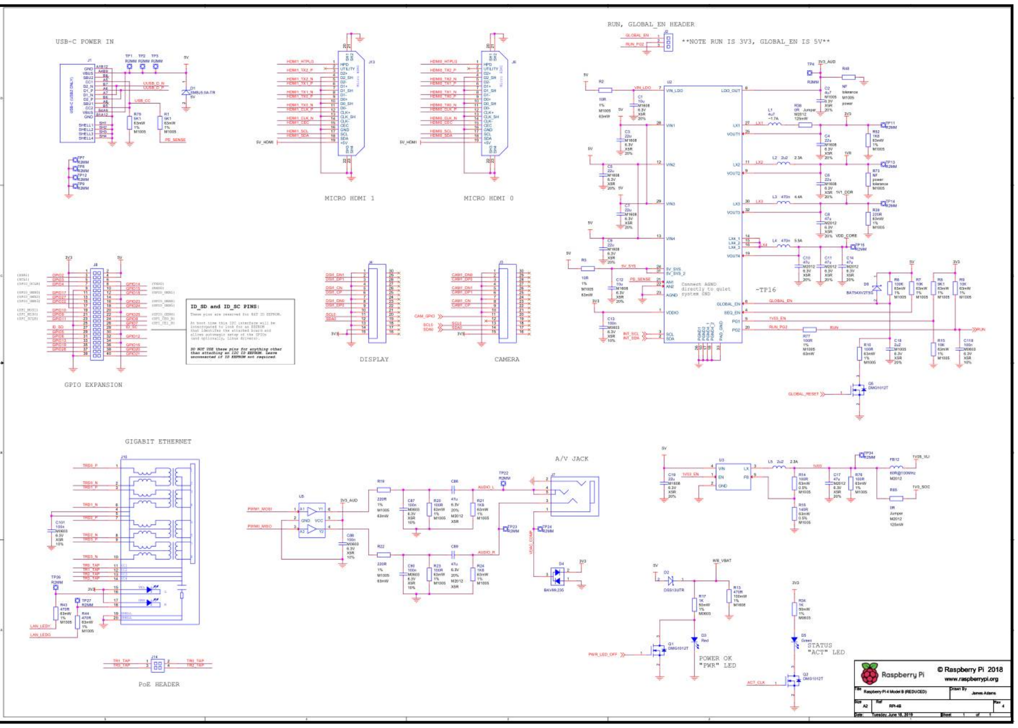
3v3 Power	1	2	5v Power
BCM 2 (V-SYNC)	3	4	5v Power
BCM 3 (H-SYNC)	5	6	Ground
BCM 4 (Blue 0)	7	8	BCM 14 (Green 2)
Ground	9	10	BCM 15 (Green 3)
BCM 17 (Green 5)	11	12	BCM 18 (Green 6)
BCM 27 (Red 7)	13	14	Ground
BCM 22 (Red 2)	15	16	BCM 23 (Red 3)
3v3 Power	17	18	BCM 24 (Red 4)
BCM 10 (Blue 6)	19	20	Ground
BCM 9 (Blue 5)	21	22	BCM 25 (Red 5)
BCM 11 (Blue 7)	23	24	BCM 8 (Blue 4)
Ground	25	26	BCM 7 (Blue 3)
BCM 0 (CLK)	27	28	BCM 1 (DEN)
BCM 5 (Blue 1)	29	30	Ground
BCM 6 (Blue 2)	31	32	BCM 12 (Green 0)
BCM 13 (Green 1)	33	34	Ground
BCM 19 (Green 7)	35	36	BCM 16 (Green 4)
BCM 26 (Red 6)	37	38	BCM 20 (Red 0)
Ground	39	40	BCM 21 (Red 1)

6

The schematic for the Raspberry Pi is also said to be open source. But look closely. The schematic is incomplete. Only the connectors and power connections are documented. ☹

The datasheet for the Broadcom processor is not available.

It is hard to understand the schematic since we do not understand the Broadcom SoC.



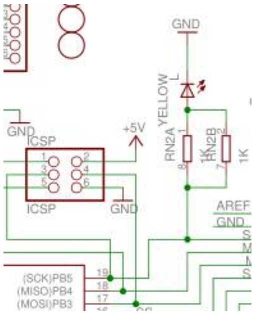
7

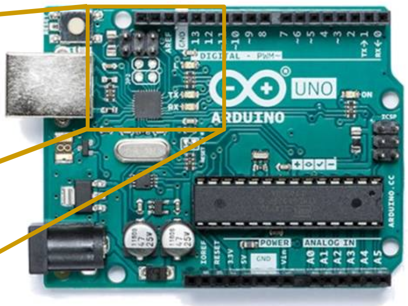
Let's get back to something substantive.
<u>How does the "Blink" program use parallel port pin 13 on the Arduino Uno</u>?

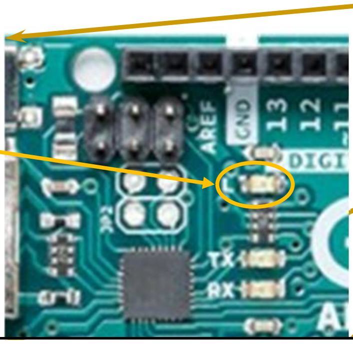
```
/*Blink, sans the header comments

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}


// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```







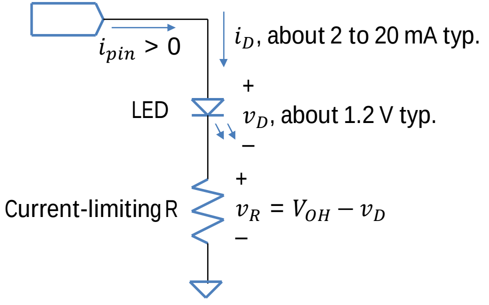
Here is the Pin 13 LED

8

Output pins are not very powerful. On Raspberry Pi, $|i_{pin}| \le$ 16 mA and 50 mA total current for all I/O pins. V_{DD} – 3.3 V
The above currents are absolute maximums. Actually $|i_{pin}| \le$ 2 mA is recommended.
Arduino Uno: $|i_{pin}| \le$ 40 mA and 500 mA total current for all I/O pins. V_{DD} = 5.0 V
I/O pins might not operate at the voltages your I/O device expects. On R-Pi, V_{pin} = 0 or 3.3 V (nominally).

Positive logic—send logic-1 to turn LED (or whatever) on

Port Pin—output high, V_{OH} = ~3.3 V or ~5.0 V



$i_{pin} > 0$

i_D, about 2 to 20 mA typ.

LED

+ v_D, about 1.2 V typ. –

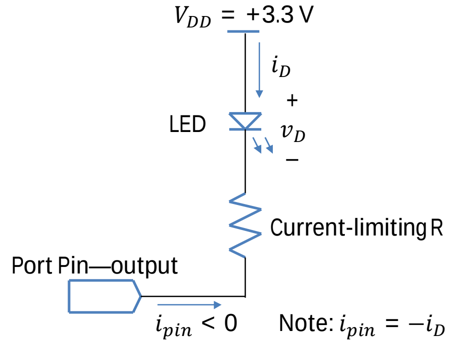
Current-limiting R

+ $v_R = V_{OH} - v_D$ –

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

$$R = \frac{V_{OH} - v_D}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V}}{2\text{ mA}} = 1.05\text{ k}\Omega \simeq 1.1\text{ k}\Omega$$

Negative logic—send logic-0 to turn LED (or ?) on



V_{DD} = +3.3 V

i_D

LED

+ v_D –

Current-limiting R

Port Pin—output

$i_{pin} < 0$

Note: $i_{pin} = -i_D$

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

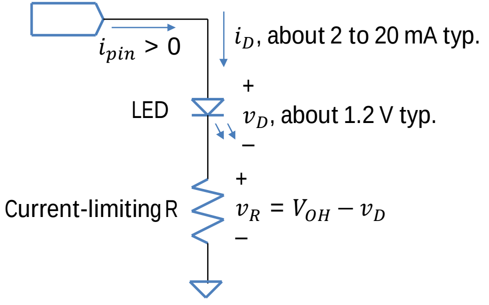
$$R = \frac{V_{DD} - v_D - V_{OL}}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V} - 0}{5\text{ mA}} = 1.05\text{ k}\Omega$$

9

Output pins are not very powerful. On Raspberry Pi, $|i_{pin}| \le$ 16 mA and 50 mA total current for all I/O pins. V_{DD} – 3.3 V
The above currents are absolute maximums. Actually $|i_{pin}| \le$ 2 mA is recommended.
Arduino Uno: $|i_{pin}| \le$ 40 mA and 500 mA total current for all I/O pins. V_{DD} = 5.0 V
I/O pins might not operate at the voltages your I/O device expects. On R-Pi, V_{pin} = 0 or 3.3 V (nominally).

Positive logic—send logic-1 to turn LED (or whatever) on

Port Pin—output high, V_{OH} = ~3.3 V or ~5.0 V



$i_{pin} > 0$

i_D, about 2 to 20 mA typ.

LED

+ v_D, about 1.2 V typ. –

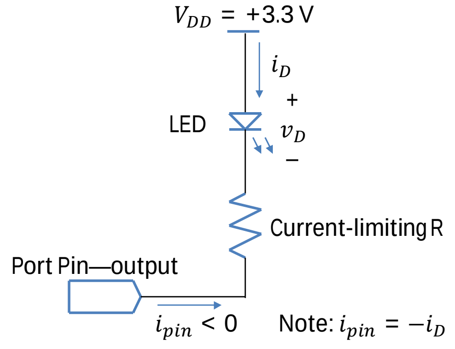
Current-limiting R

+ $v_R = V_{OH} - v_D$ –

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

$$R = \frac{V_{OH} - v_D}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V}}{2\text{ mA}} = 1.05\text{ k}\Omega \simeq 1.2\text{ k}\Omega$$

Negative logic—send logic-0 to turn LED (or ?) on



V_{DD} = +3.3 V

i_D

LED

+ v_D –

Current-limiting R

Port Pin—output

$i_{pin} < 0$

Note: $i_{pin} = -i_D$

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

$$R = \frac{V_{DD} - v_D - V_{OL}}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V} - 0}{2\text{ mA}} = 1.05\text{ k}\Omega$$

10

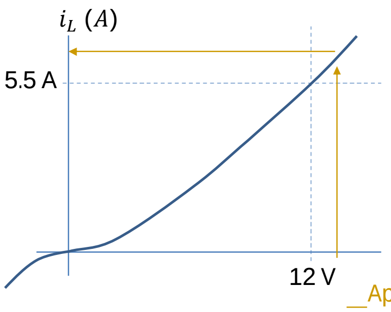
Output pins are not very powerful. On Raspberry Pi, $|i_{pin}| \leq$ 16 mA and 50 mA total current for all I/O pins. $V_{DD} - 3.3$ V
The above currents are absolute maximums. Actually $|i_{pin}| \leq$ 2 mA is recommended.
Arduino Uno: $|i_{pin}| \leq$ 40 mA and 500 mA total current for all I/O pins. V_{DD} = 5.0 V
I/O pins might not operate at the voltages your I/O device expects. On R-Pi, V_{pin} = 0 or 3.3 V (nominally).

Observe that the Arduino has about an order of magnitude more driving power for outputs than the Raspberry Pi.
Every microcontroller or SoC chip will have similar limits.
These two chips happen to represent two good representative cases of the most and least powerful typically marketed.

11

Side-note: The i-v characteristics of typical loads.

"Voltage-driven" load, eg. Incandescent lamp
Car headlight, rated 12 V 65 W, nominal current is 5.5 A
Nominal resistance is 2.18 Ω
Its i-v characteristic is nearly linear.

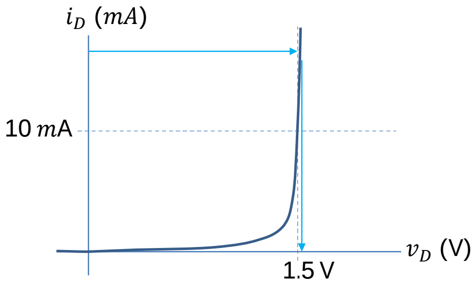


Applied voltage sets the current.

Apply 12 V across the lamp,
let the current be whatever it will be, which is 5.5 A

Suppose you apply 14 V? $i \simeq \frac{14\text{ V}}{2.18\ \Omega}$ = 6.4 A

"Current-driven" load, e.g. an LED
Rated 10 mA, 15 mW, Nominal voltage is 1.5 V
Exponential characteristic,
Resistance is not a useful concept—curve is too warped



Apply 10 mA to the LED, get 1.5 V drop.
Suppose we apply 14 mA to the LED, get
1.52 V drop, not so far different than normal.
Power is well controlled with a current driven strategy,
but not with a voltage-driven strategy!

12

Output pins are not very powerful. On Raspberry Pi, $|i_{pin}| \leq$ 16 mA and 50 mA total current for all I/O pins. V_{DD} – 3.3 V
The above currents are absolute maximums. Actually $|i_{pin}| \leq$ 2 mA is recommended.
Arduino Uno: $|i_{pin}| \leq$ 40 mA and 500 mA total current for all I/O pins. V_{DD} = 5.0 V
I/O pins might not operate at the voltages your I/O device expects. On R-Pi, V_{pin} = 0 or 3.3 V (nominally).

Observe that the Arduino has about an order of magnitude more driving power for outputs than the Raspberry Pi.
Every microcontroller or SoC chip will have similar limits.
These two chips happen to represent two good representative cases of the most and least powerful typically marketed.

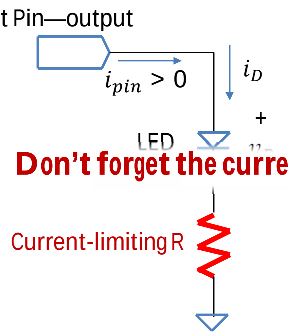
Voltage-driven loads (e.g. light bulbs, small motors, small solenoids) can be driven directly from a pin, if enough current is available from the pin.

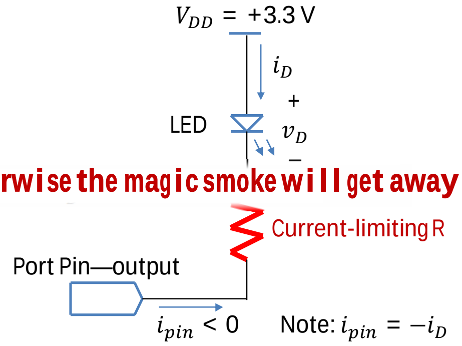
Current driven loads need to be in series with a *current limiting resistor* to set the load current properly.

Output pins are not very powerful. On Raspberry Pi, $|i_{pin}| \leq$ 16 mA and 50 mA total current for all I/O pins. V_{DD} – 3.3 V
The above currents are absolute maximums. Actually $|i_{pin}| \leq$ 2 mA is recommended.
Arduino Uno: $|i_{pin}| \leq$ 40 mA and 500 mA total current for all I/O pins. V_{DD} = 5.0 V
I/O pins might not operate at the voltages your I/O device expects. On R-Pi, V_{pin} = 0 or 3.3 V (nominally).

Positive logic—send logic-1 to turn LED (or whatever) on

Negative logic—send logic-0 to turn LED (or ?) on





Note: $i_{pin} = -i_D$

Don't forget the current limiting resistor, otherwise the magic smoke will get away!

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

$$R = \frac{V_{OH} - v_D}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V}}{2\text{ mA}} = 1.05\text{ k}\Omega \simeq 1.2\text{ k}\Omega$$

The LED has a specified ideal current, i_{Dq}, say 2 mA
The LED will have a typical voltage drop, say v_{Dq} = 1.2 V
The I/O pin has a certain logic voltage for logic-1, V_{OH}

$$R = \frac{V_{DD} - v_D - V_{OL}}{i_D} = \frac{3.3\text{ V} - 1.2\text{ V} - 0}{2\text{ mA}} = 1.05\text{ k}\Omega$$