

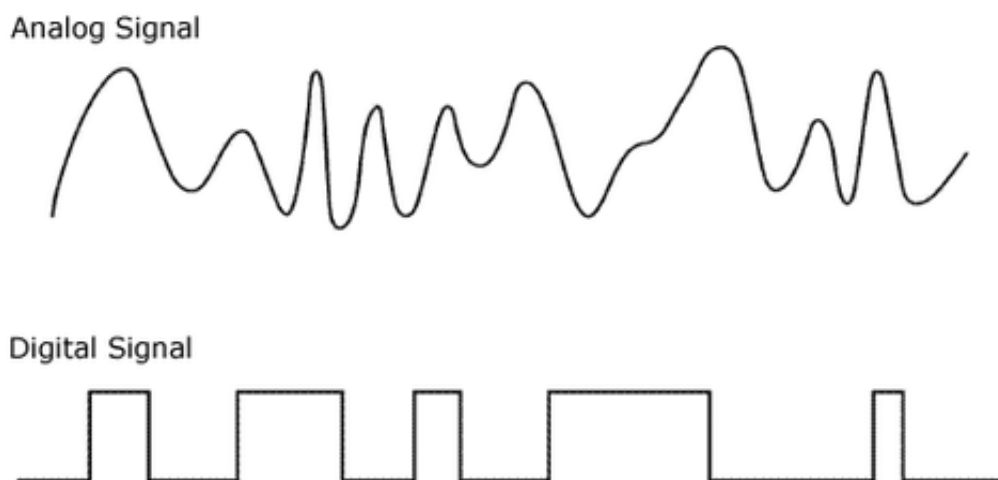
Raspberry Pi Sensors

Author: Nick Lee

Raspberry Pi is a digital computer, and a digital computer only knows two numbers: 1 and 0.

A lot of physical quantities, e.g. temperature, humidity, wind speed, etc., are smooth-varying numbers. They can take on values such as 25.6, 98.76, or 33.3333333333. These are called analog quantities.

For Raspberry Pi to understand analog quantities, a sensor has to convert those analog quantities to digital signals.



There are many digital "languages", e.g. 1-wire, I2C, and SPI. You have to configure Raspberry Pi to speak the sensor's "language" before they can communicate.

Enable

```
$ sudo raspi-config
```

```
↳ Interfacing Options
```

```
↳ 1-Wire ?
```

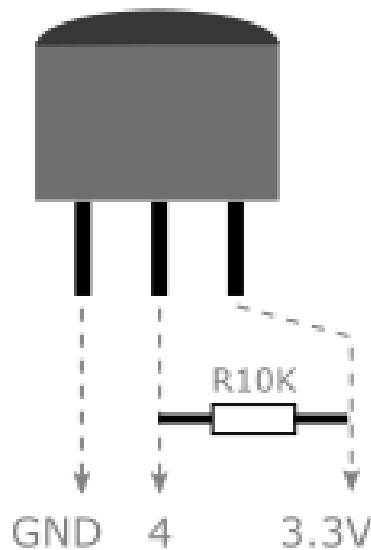
```
↳ I2C ?
```

```
↳ SPI ?
```

DS18B20

temperature sensor

This is a 1-wire sensor. Wire it up according to the diagram below. You need a 10K pull-up resistor.



```
$ cd /sys/bus/w1/devices
$ ls
```

What do you see?

Install sensor package

```
$ sudo pip3 install sensor
```

```
$ python3
```

```
>>> from sensor.DS18B20 import DS18B20
>>> ds = DS18B20('28-ZZZZZZZZZ')
>>> ds.temperature()
```

What do you see? How do you access different units of the temperature?

```
>>> t = ds.temperature()
>>> t.C
28.937
>>> t.F
84.0866
```

Check I2C bus

```
$ i2cdetect -y 1
```

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:				--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--								

Nothing is on the I2C bus, because nothing has been attached yet.

SHT20

humidity and temperature sensor

```
$ i2cdetect -y 1
```

What is the sensor's address?

```
$ python3
```

```
>>> from sensor.SHT20 import SHT20
```

```
>>> sht = SHT20(1, ??address??)
```

```
>>> h = sht.humidity()
```

```
>>> h.RH
```

```
>>> t = sht.temperature()
```

```
>>> t.C
```

BMP180

pressure and temperature sensor

```
$ i2cdetect -y 1
```

What is the sensor's address?

```
$ python3
```

```
>>> from sensor.BMP180 import BMP180
```

```
>>> bmp = BMP180(1, ??address??)
```

```
>>> p = bmp.pressure()
```

```
>>> p.hPa
```

```
>>> t = bmp.temperature()
```

```
>>> t.C
```

```
# Look up mean sea level pressure from local observatory.
```

```
# 1009.1 hPa is only for example.
```

```
>>> a = p.altitude(msl=1009.1)
```

```
>>> a.m
```


Deal with Analog Signals

All sensors we have used so far convert analog quantities to a digital format, so Raspberry Pi can read the numbers readily. **What if we get a sensor that gives analog signals?**

For example, a **TMP36** temperature sensor outputs a varying voltage proportional to the temperature measured.

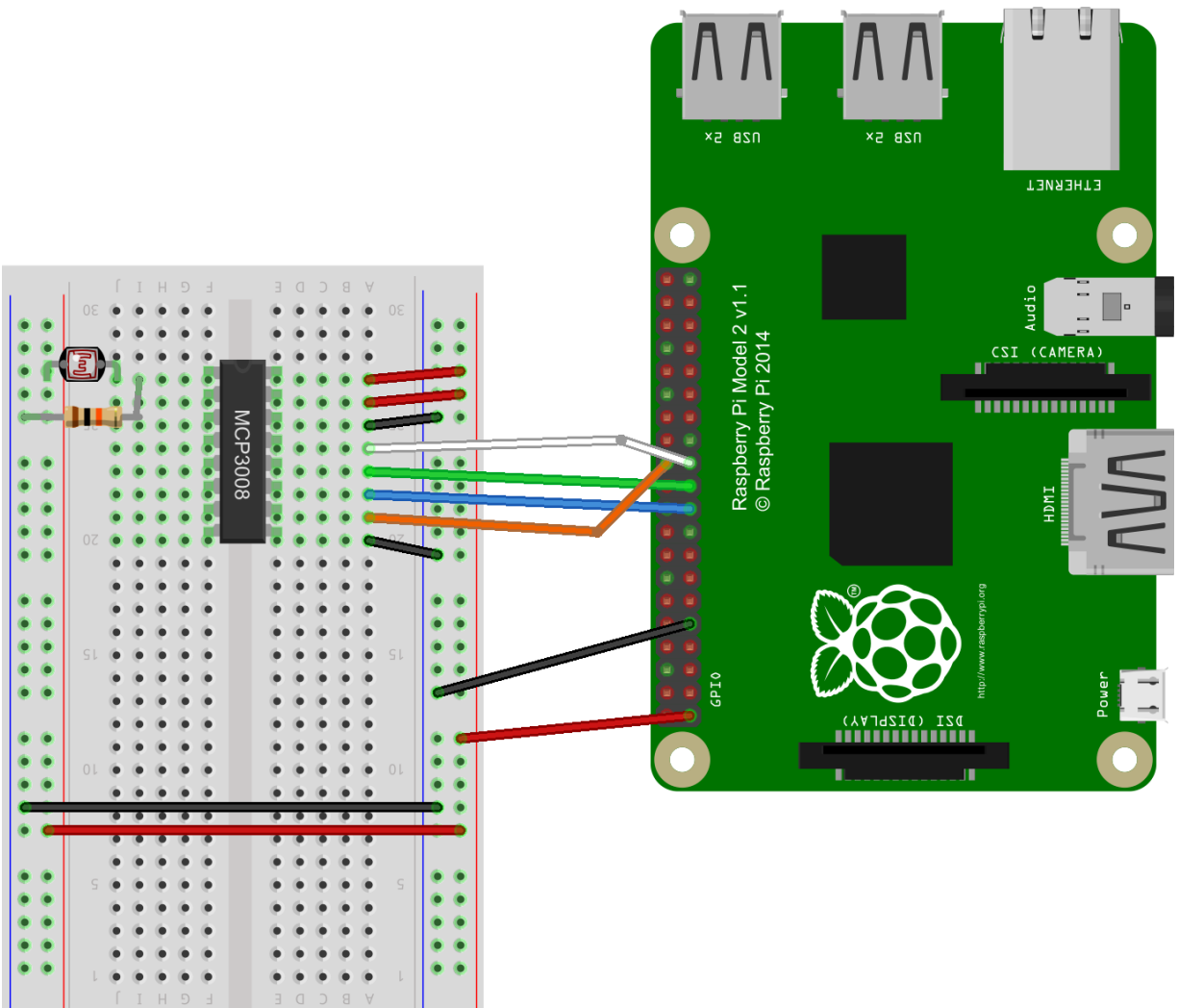
Another example is a **photoresistor**, whose resistance changes with light intensity.

How could Raspberry Pi read these non-digital values?

The solution is to insert an **analog-to-digital converter (ADC)** between the sensor and the Pi.



Photoresistor and MCP3008

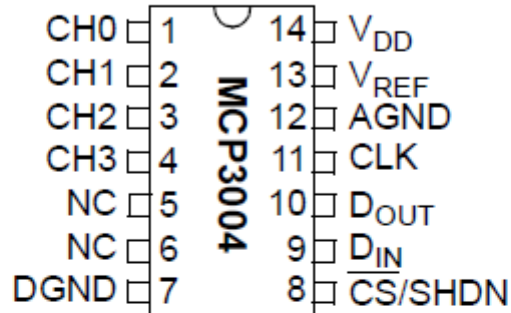


fritzing

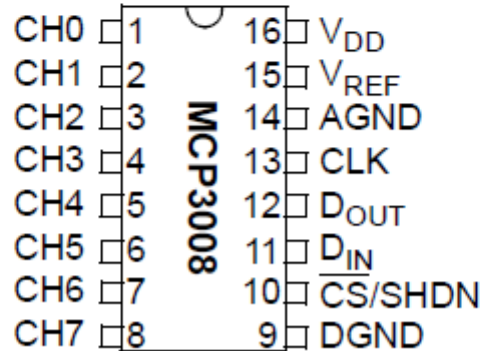
- the ADC uses a chip select of 0
- the reference voltage is 3.3V
- the photoresistor and 10K resistor form a voltage divider, and the result is tapped by channel 0
- as the photoresistor's resistance changes, channel 0's voltage also changes.

Package Types

PDIP, SOIC, TSSOP



PDIP, SOIC























The pinouts of MCP3004 and MCP3008 are very similar. The sensor package, while explicitly supporting MCP3004, can be used to read MCP3008 as well.

```
$ python3
```

```
>>> from sensor.MCP3004 import MCP3004
>>> mcp = MCP3004(bus=0, addr=0, vref=3.3)
>>> mcp.voltage(0) # read channel 0
```

Varying the light intensity, you should notice the voltage changes too.

Raspberry Pi B+ J8 Header

<i>Pin#</i>	<i>NAME</i>		<i>NAME</i>	<i>Pin#</i>
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 1.1
16/07/2014

<http://www.element14.com>