I2C

I2C, sometimes IIC or I^2C, stands for inter IC

2-line bus, clock (SCL) and data (SDA)

Devices individually addressable

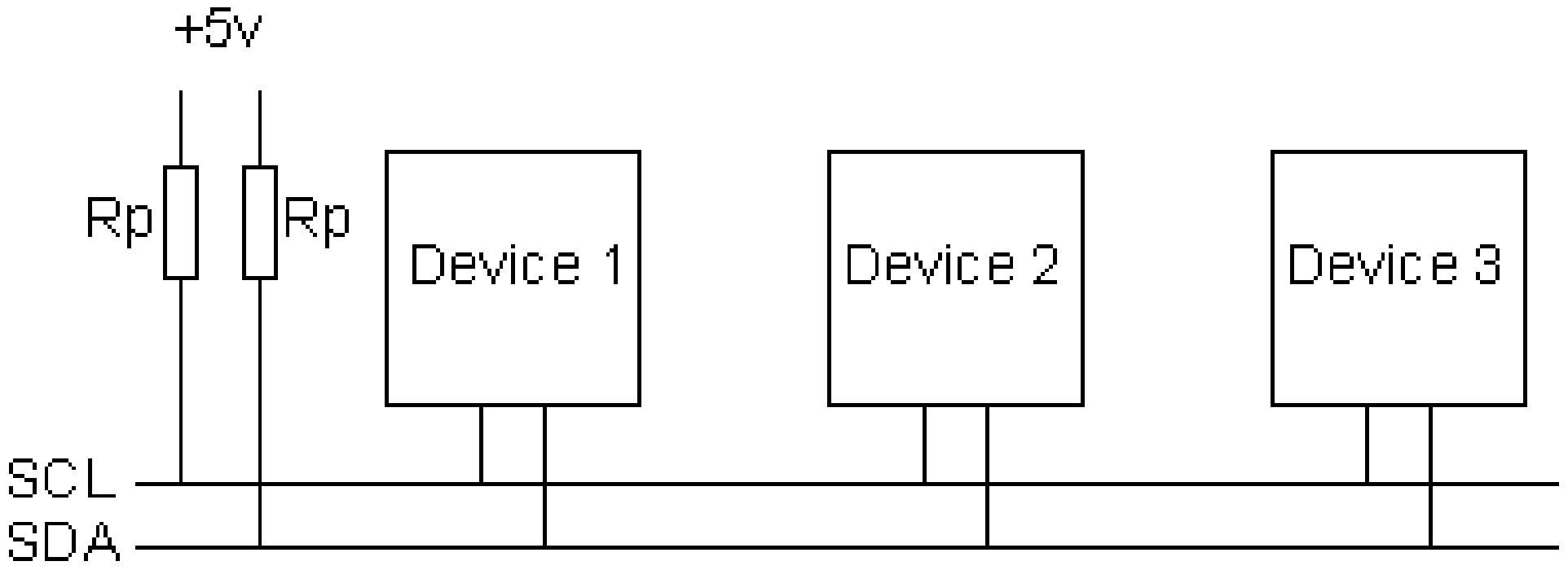
Not sensitive to clock speed

No bus power contention

Sources

- http://www.robot-electronics.co.uk/i2c-tutorial
- http://www.ti.com/lit/an/slva704/slva704.pdf

Simple I2C bus

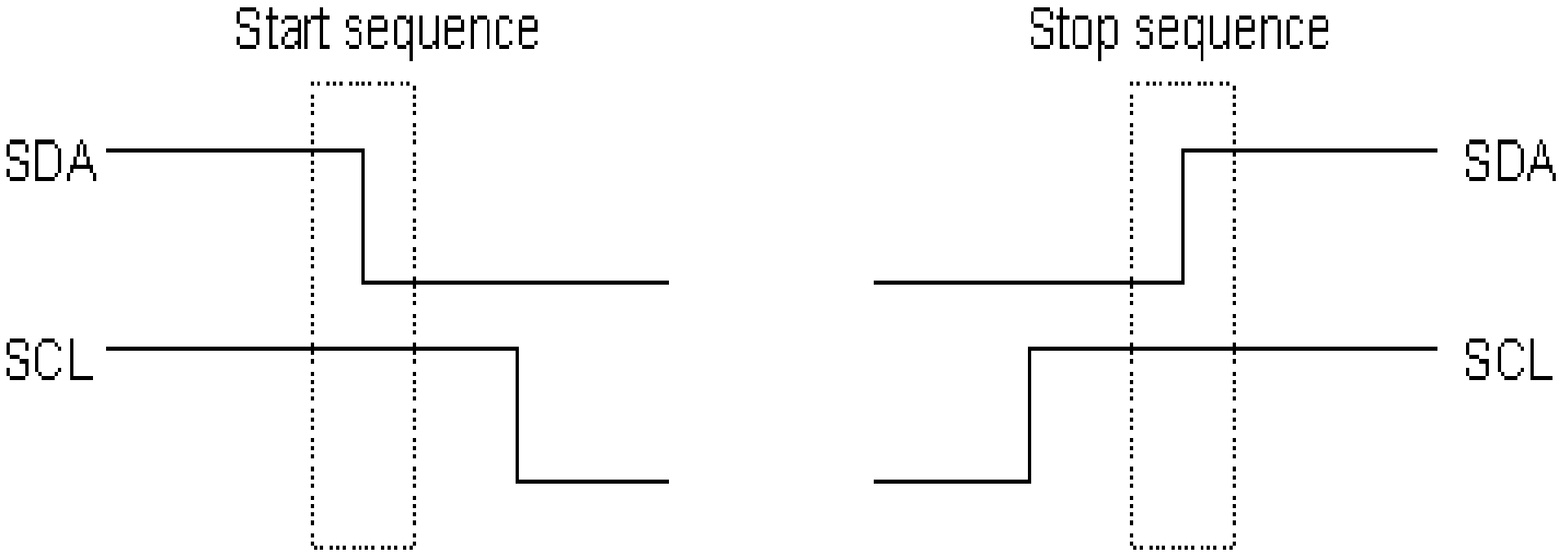


Pull up resistors

Pi has 1.8kΩ pull-up resistors on the SCL and SDA lines – do not add more, and watch for pull-up resistors on break-out boards

I2C Bus signals

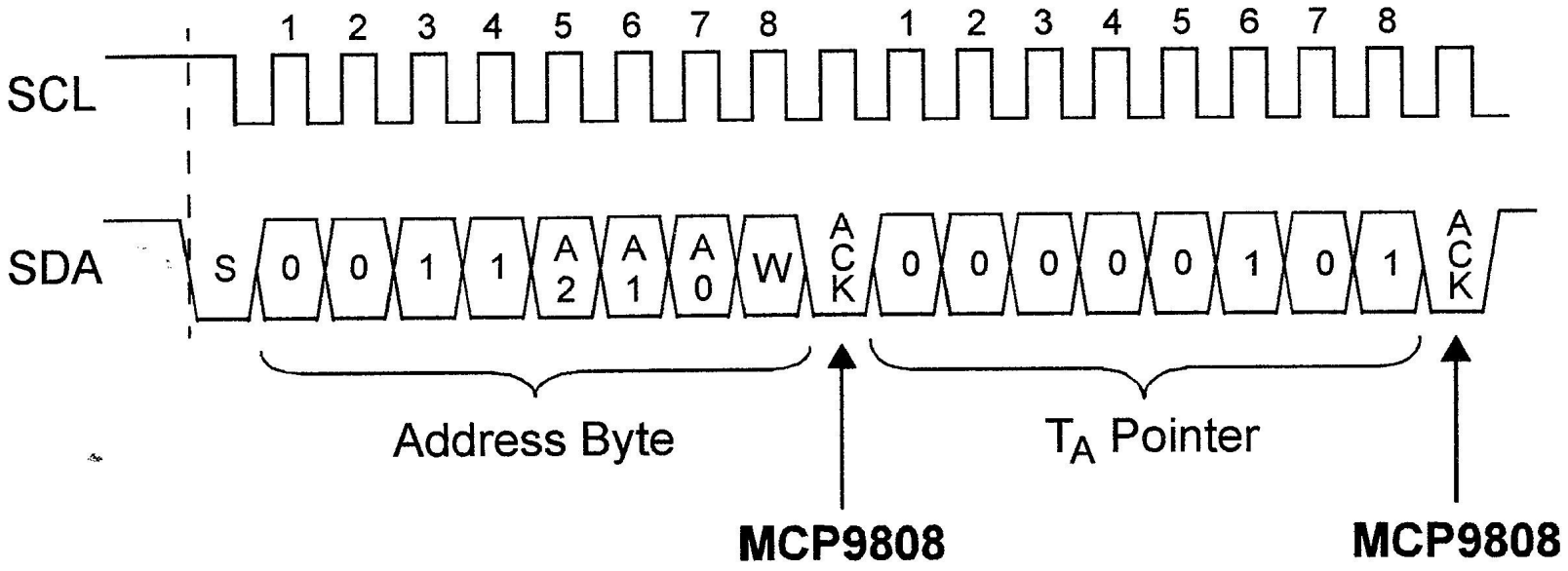
Start and stop are unique, the only place where SDA can change while SCL is high



Writing to an I2C device

- Master sends start sequence
- Master sends one byte with device address, and read/write bit set low (write)
- Slave acknowledges (bit 9 pulled low)
- Master sends device register number
- Slave acknowledges
- Master sends data bytes
- Master sends stop sequence

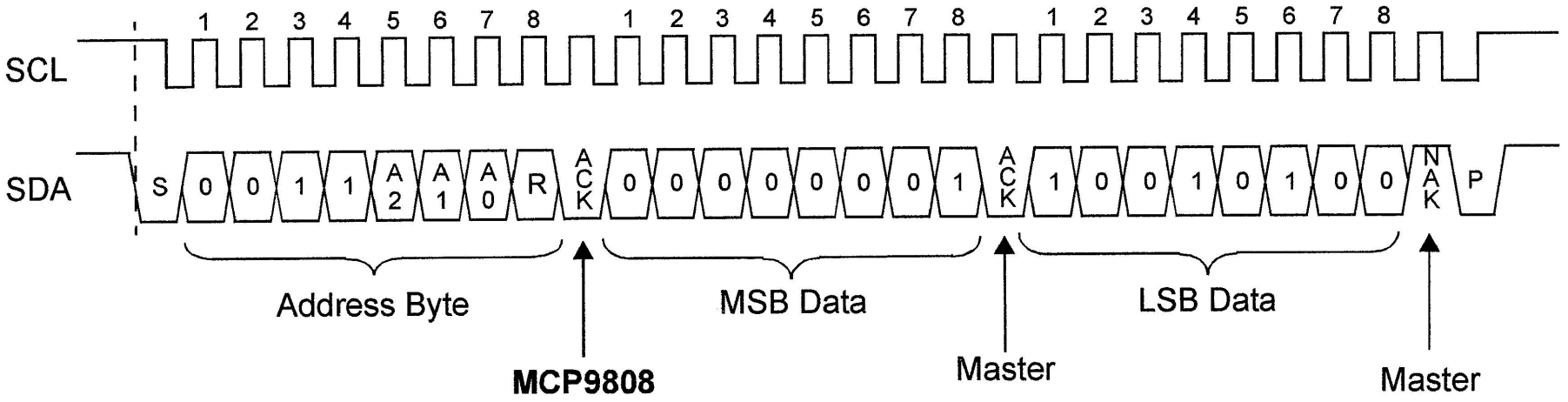
Writing to an I2C device



Reading from an I2C device

- Master writes to device and register but sends no data
- Master sends a start signal
- Master sends device address with the read/write bit set high (ie read)
- Slave acknowledges
- Slave sends data bytes
- Master acknowledges (except last byte)
- Master sends stop sequence

Reading from an I2C device

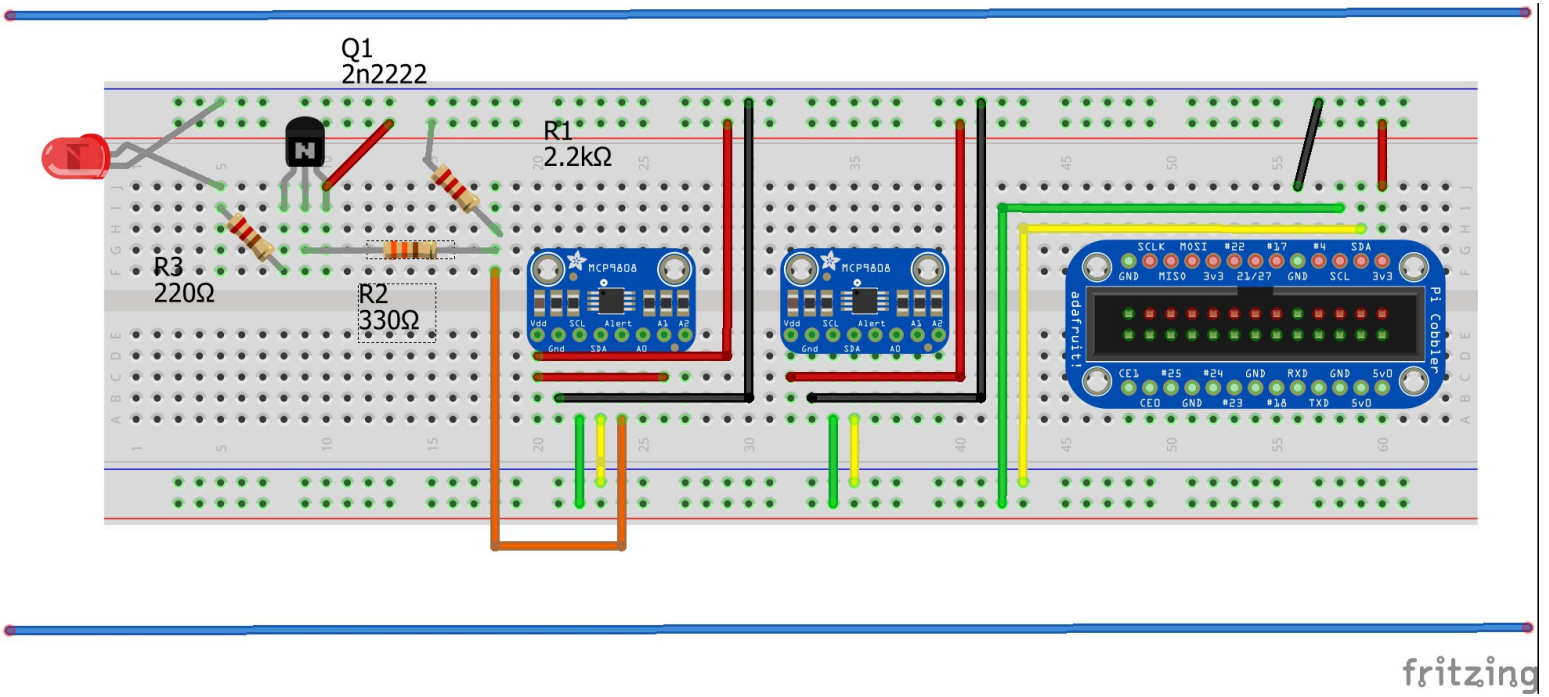


My System

Pi 2 B 512MB memory

Raspbian Jessie kernal version 4.4

Two 9808 temp sensor boards on I2C bus



Pi setup

(You may need to install I2C tools

$ sudo apt-get install i2c-tools ◄)

I2C will need to be enabled

$ sudo raspi-config ◄

then Interfacing Options / I2C enable automatic loading

(Previously this was under Advanced Options)

Then reboot

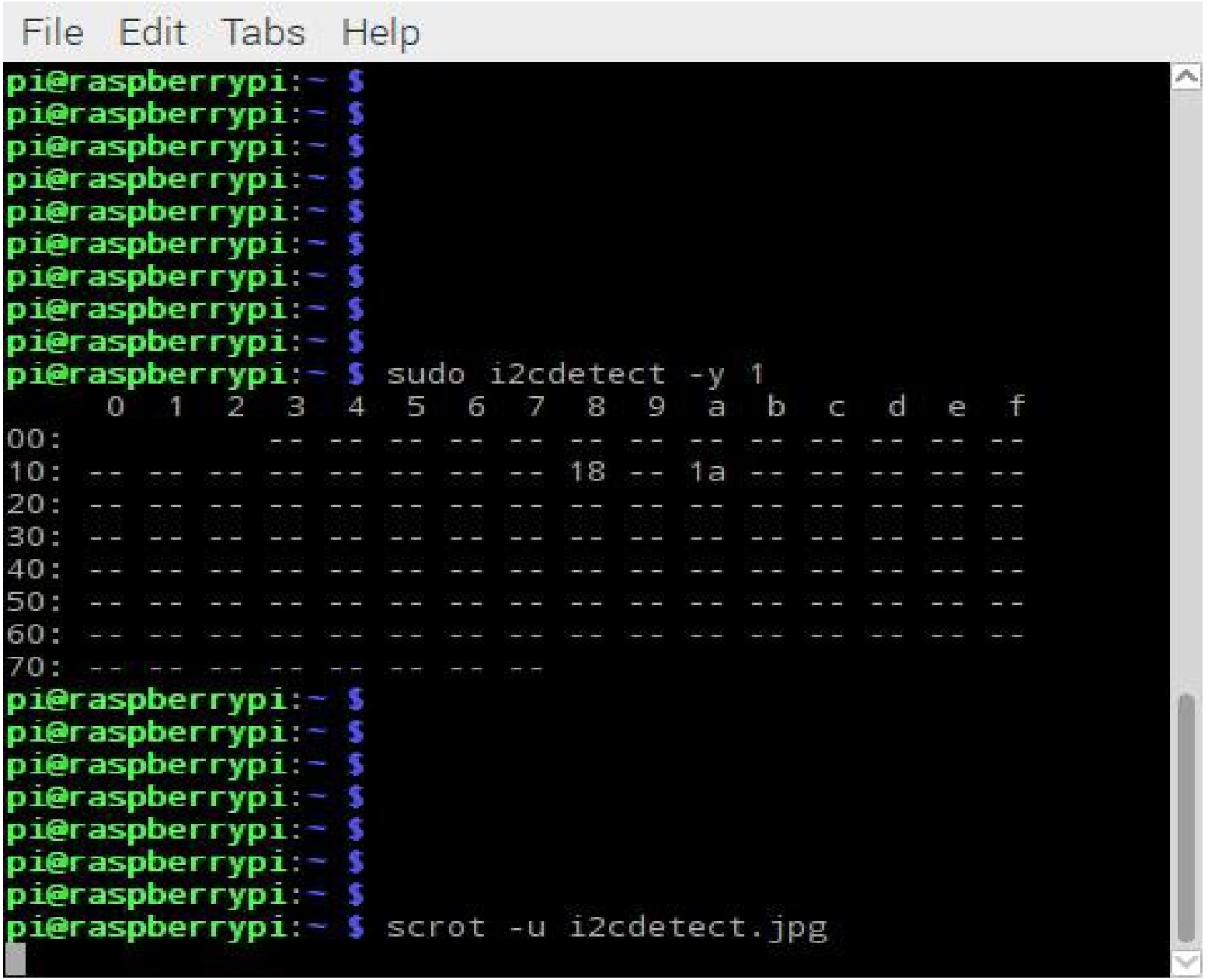
Pi setup

To test the system, connect an I2C module to the Pi and enter;

$ sudo i2cdetect -y 1 ◄

($ sudo i2cdetect -y 0 ◄ on older Pi models)

I2cdetect -y 1



Python - setting up to use I2C

```
# define I2C address of sensor
i2c_addr = 0x18

# import libraries
import smbus as smbus

#configure I2C bus for functions
i2c = smbus.SMBus(1)
```

Python – simple I2C read

```
# Now read from 9808 unit, register 5, and print result


temp = i2c.read_word_data( i2c_addr, 5 )


print ("9808 at address ", hex (i2c_addr),
"   returned ", bin (temp))
```

Massaging the data

- Output data from the 9808 sensor is sent as a two byte word, in big endian order. The Pi assumes little endian format, so the two bytes of data have to be separated and flipped.

- The first 3 bits of the 9808 data are flags which are not needed for temperature measuring, and the fourth bit is a sign bit for temperatures below zero.

98
08 temperature binary word

Bit	value	Bit	value
15	T-crit	07	2^3 ^{O}C
14	T-upper	06	2^2 ^{O}C
13	T-lower	05	2^1 ^{O}C
12	Sign	04	2^0 ^{O}C
11	2^7 ^{O}C	03	2^{-1} ^{O}C
10	2^6 ^{O}C	02	2^{-2} ^{O}C
09	2^5 ^{O}C	01	2^{-3} ^{O}C
08	2^4 ^{O}C	00	2^{-4} ^{O}C

Demo1.py essentials

```
#!/usr/bin/env python
# define I2C address of sensor
i2c_addr = 0x18
# import libraries
import smbus as smbus
import time
#configure I2C bus for functions
i2c = smbus.SMBus(1) # For original Pi use "i2c = smbus.SMBus(0)"
# Now read from 9808 unit, register 5, and print result
while True:
    temp = i2c.read_word_data( i2c_addr, 5 )
    print ("9808 at address ", hex (i2c_addr), " returned ", bin (temp))
    time.sleep(1)
```

Demo1.py output

Python 3.4.2 (default, Oct 19 2014, 13:31:11)

[GCC 4.9.1] on linux

Type "copyright", "credits" or "license()" for more information.

>>> ========== RESTART==========

>>>

9808 at address 0x18 returned 0b11111111000001

9808 at address 0x18 returned 0b11111111000001

9808 at address 0x18 returned 0b100000011000001

9808 at address 0x18 returned 0b11111111000001

9808 at address 0x18 returned 0b11111111000001

9808 at address 0x18 returned 0b11111111000001

9808 at address 0x18 returned 0b11111111000001

Demo2.py

Demo2 reads the 9808 as does Demo1, strips off the first 3 flags, checks the fourth flag (temp below zero) and strips that off, then calculates the temperature based on the retained data.

Demo2.py output

9808 at address 0x18 returned 0011111111000001

reversed bytes = 1100000100111111

Most significant byte of addr1 is 00000001 after stripping first 3 bits

Most significant byte of addr1 is 00000001 after stripping first 4 bits

19.9375 degrees C

9808 at address 0x18 returned 0100000011000001

reversed bytes = 1100000101000000

Most significant byte of addr1 is 00000001 after stripping first 3 bits

Most significant byte of addr1 is 00000001 after stripping first 4 bits

20.0 degrees C

Demo3.py

Demo3 reads two 9808 units as does Demo2 and processes the data in the same way. The only coding required is to set another address, and duplicate the code using this second address.

- i2c_addr = 0x18
- i2c_addr2 = 0x1a

Demo3.py output

9808 at address 0x18 returned 0100000011000001

reversed bytes = 1100000101000000

Most significant byte of addr1 is 00000001 after stripping first 3 bits

Most significant byte of addr1 is 00000001 after stripping first 4 bits

20.0 degrees C

9808 at address 0x1a returned 0011111111000001

reversed bytes = 1100000100111111

Most significant byte of addr2 is 00000001 after stripping first 3 bits

Most significant byte of addr2 is 00000001 after stripping first 4 bits

19.9375 degrees C

9808 temperature binary word

Bit	value	Bit	value
15	T-crit	07	2^3 ^{O}C
14	T-upper	06	2^2 ^{O}C
13	T-lower	05	2^1 ^{O}C
12	Sign	04	2^0 ^{O}C
11	2^7 ^{O}C	03	2^{-1} ^{O}C
10	2^6 ^{O}C	02	2^{-2} ^{O}C
09	2^5 ^{O}C	01	2^{-3} ^{O}C
08	2^4 ^{O}C	00	2^{-4} ^{O}C

Demo4.py

This sketch reads register 5 and separates out the flags.

The 9808 defaults to 0°C for the various temperature settings – these can be changed and if necessary locked.

At power up, and ambient about 20°C T is above T-crit and T-upper, so these flags are set to 1, and T is also above T-lower, so this flag is set to 0.

Demo4.py output

9808 at address 0x18 data

flag1 = T-crit 1

flag2 = T-upper 1

flag3 = T-lower 0

19.875 degrees C

9808 at address 0x1a data

flag1 = T-crit 1

flag2 = T-upper 1

flag3 = T-lower 0

19.8125 degrees C

Demo5.py

This sketch writes a new temperature settings to T-crit and T-upper and then shows flags and temperature at each 9808 sensor.

On the 9808 the critical temperature is stored in register 4.

The critical code is;

i2c.write_word_data (i2caddr2, 4, 0b0111000000000001)

Demo5.py output

9808 at address 0x1a data

flag1 = T-crit 0

flag2 = T-upper 0

flag3 = T-lower 0

22.0 degrees C

9808 at address 0x1a data

flag1 = T-crit 0

flag2 = T-upper 1

flag3 = T-lower 0

23.0625 degrees C

9808 at address 0x1a data

flag1 = T-crit 1

flag2 = T-upper 1

flag3 = T-lower 0

26.0625 degrees C

Demo5.py continued

Some code also added to activate the alert output when the alert condition is exceeded, and to set the alert condition to high.

Critical code is written to register 1

i2c.write_word_data(i2c_addr2, 1, 0b0000111000000000

Alert output is wired to a LED to switch it on when the alert goes high