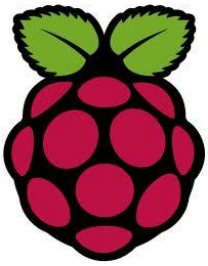
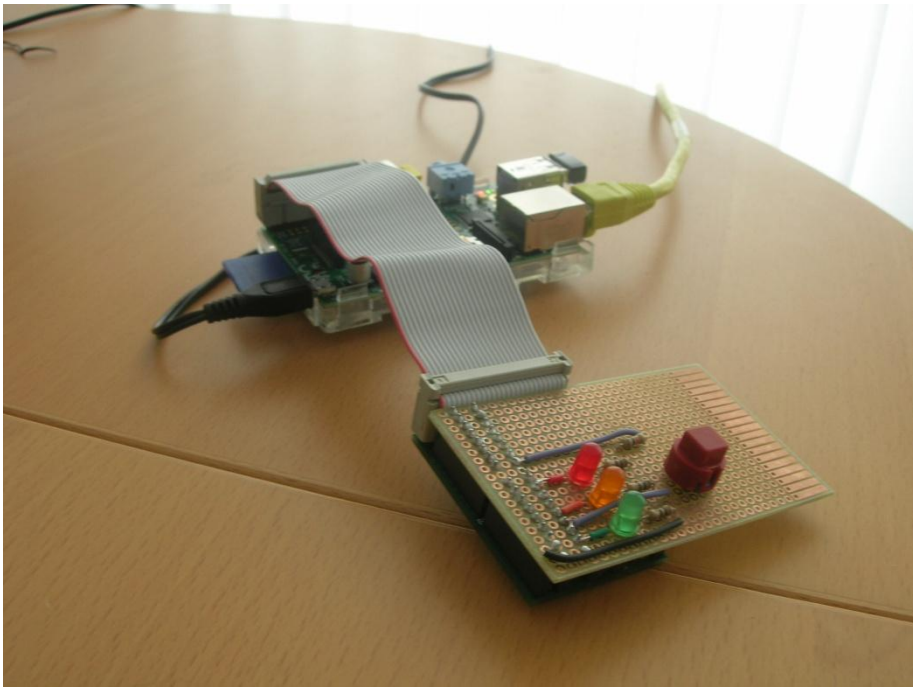


# Raspberry PI Traffic Lights $I^2C$

## Project Notes




## Bob Rathbone Computer Consultancy

www.bobrathbone.com


16th of August 2013

# Contents


Introduction ..... 3

Raspberry PI computer ..... 3

Parts List ..... 4

The MCP23017 expander board ..... 5

The MCP23017 chip ..... 5

MCP2317 and Ciseco Board signals ..... 6

The I2C interface ..... 7

GPIO I2C interface ..... 8

Wiring details ..... 9

Wiring LEDs ..... 9

Switch wiring ..... 9

Wiring list ..... 9

Exercise 1 – Get one LED working ..... 10

Wire up the green LED ..... 10

Install the smbus libraries ..... 10

Flashing LED program ..... 11

Exercise 2 – Get all LEDs working ..... 12

Wire up the red and orange LEDs ..... 12

Create a program for the standard traffic lights sequence. ..... 12

Excercise 3 Pedestrian Crossing ..... 13

Wire the push button ..... 13

Run the Pedestrian Crossing program ..... 13

The completed board ..... 14

Appendix A – Program listing ..... 16

A.1 The flashing I2C LED program – i2cLed.py ..... 16

A.2 The Traffic Lights program – i2cTrafficLights.py ..... 16

A.3 The Pedestrian Crossing program – i2cPedestrianCrossing.py ..... 17

Appendix B Licences ..... 19

Appendix C Resistor Colour codes ..... 20

Glossary ..... 21

# Introduction

This project has been designed to help students and hobbyists get started with using the MCP230117 I/O expander connected to the GPIO interface on the Raspberry PI which uses the **I²C** interface. The principle hardware required to build the traffic lights using an MCP23017 chip consists of the following components:

- A Raspberry PI computer
- An MCP23017 expander board
- A prototype board
- A red, orange and green LED
- A push to make PCB mounted switche (Pulse type)
- Resistors, single strand wire etc.

## Raspberry PI computer

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the United Kingdom by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

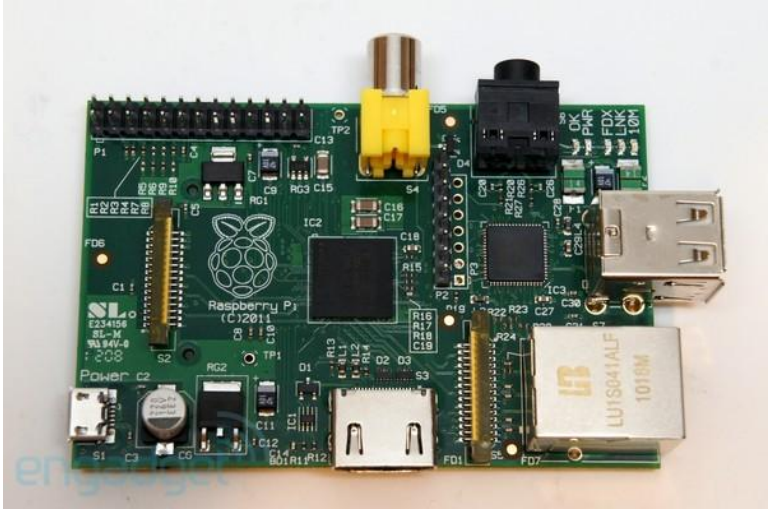


**Figure 1 Raspberry PI Computer**

More information on the Raspberry PI computer may be found here:
http://en.wikipedia.org/wiki/Raspberry_Pi

If you are new to the Raspberry PI try the following beginners guide. http://elinux.org/RPi_Beginners

# Parts List

This project uses the Ciseco “Slice of PI “ MCP23017 I/O expander board available from Ciseco.

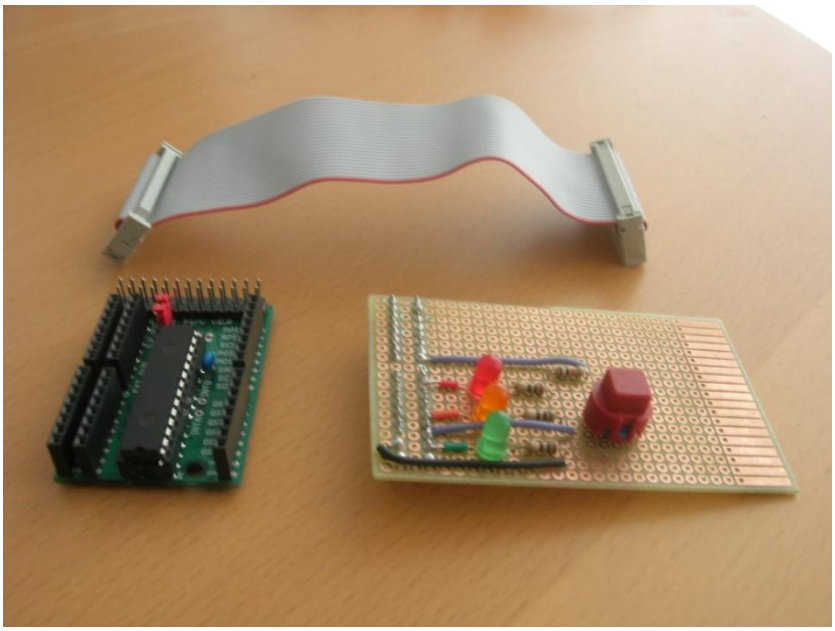


Figure 2 The components for the I²C traffic lights board

The following table shows the parts list for the Raspberry PI **I²C** Traffic Lights.

| Part | Description | Supplier |
|---|---|---|
| **Raspberry PI** | Raspberry PI credit card computer | Farnell or RS |
| **4 Gigabyte SD card** | For the operating system | Any computer or photographic store |
| **5mm LEDs** | One red, one green and one orange/yellow LED. Optional one white LED for the first exercise. | Tandy or Farnell Element 14 |
| **MCP23017 expander board** | "Ciseco "Slice of PI" MCP23017 expander | Ciseco |
| **Prototyping board** | Any small prototyping board | Tandy or Farnell Element 14 |
| **Three 220 Ohm resistors** | In line resistors for LED | Tandy or Farnell Element 14 |
| **One 1K resistor** | For GPIO input switch | Tandy or Farnell Element 14 |
| **Push to make switch** | Either PCB mount or wired | Tandy or Farnell Element 14 |
| **26 way Ribbon Cable** | To connect the I/O expander to the Raspberry PI. | Farnell Element 14 |
| **Single in line male connectors** | Enough for four 8 pin connectors | |
| **Single strand wire** | Various colours | Tandy or Farnell Element 14 |
| **Thin sleeving** | To slip over the LED leads | Tandy or Farnell Element 14 |

# The MCP23017 expander board

There are a number of expander boards available as shown in the following figure. For this project we are using the one shown on the right from Ciseco.

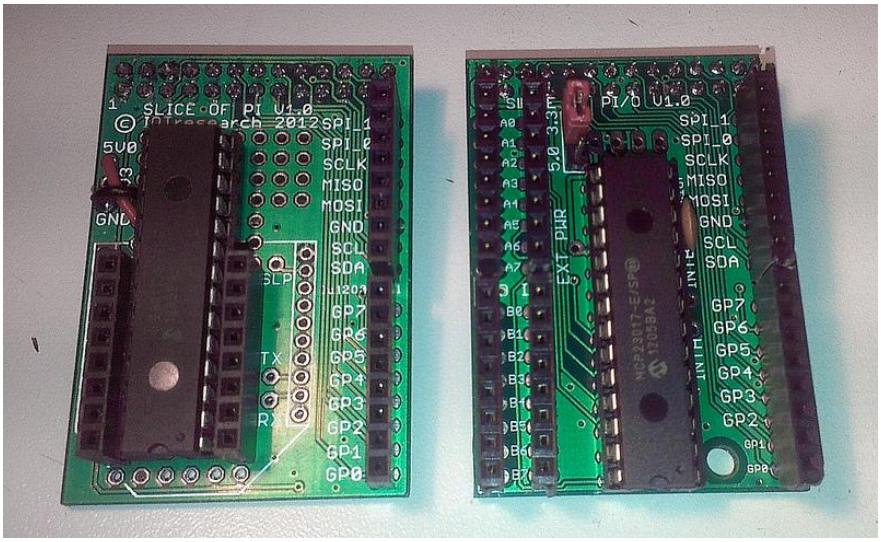



Figure 3 MCP23017 expander boards

Please note in this picture the B0 to B7 outputs are labelled the wrong way round. B7 at the bottom should be B0 and so on.

## The MCP23017 chip

The following diagram shows the pin outs for the MCP23017

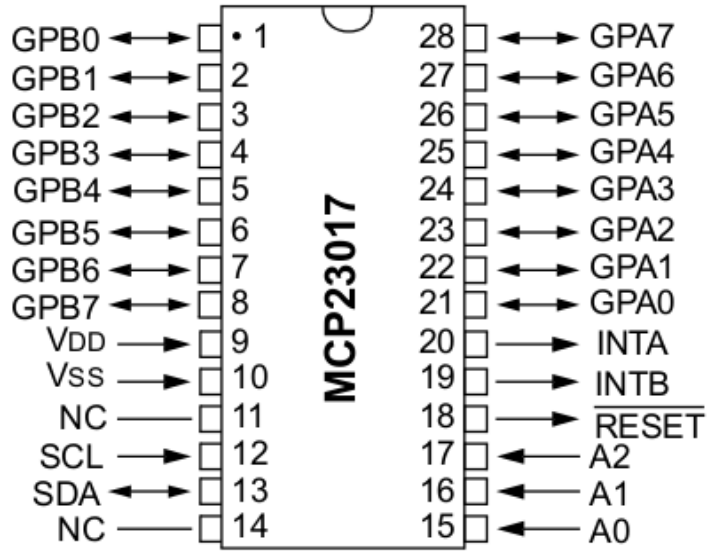



Figure 4 The MCP23017 chip

There are two output banks A and B of 8 pins each. The **I²C** interface consists of a data (SDA) and clock (SCL). The chip has up to eight addresses by biasing the A0, A1 and A2 lines. The full

specification for the MCP23017 chip can be found at:
http://ww1.microchip.com/downloads/en/devicedoc/21952b.pdf

## MCP2317 and Ciseco Board signals

The MCP2317 chip has 16 input/output pins as shown in the following table. This shows the hex ,decimal or binary values that have to be written to the MCP2317 chip enable the outputs. The banks A and B are addressed by 0x12 and 0x13 respectively (See example program listings in appendix A).

| I/O | MCP23017 | Pin | Ciseco Board | Bank | Hex | Decimal | Binary |
|---|---|---|---|---|---|---|---|
| **1** | GPA0 | 1 | A0 | 0x12 | 0x01 | 1 | 00000001 |
| **2** | GPA1 | 2 | A1 | 0x12 | 0x02 | 2 | 00000010 |
| **3** | GPA2 | 3 | A2 | 0x12 | 0x04 | 4 | 00000100 |
| **4** | GPA3 | 4 | A3 | 0x12 | 0x08 | 8 | 00001000 |
| **5** | GPA4 | 5 | A4 | 0x12 | 0x10 | 16 | 00010000 |
| **6** | GPA5 | 6 | A5 | 0x12 | 0x20 | 32 | 00100000 |
| **7** | GPA6 | 7 | A6 | 0x12 | 0x40 | 64 | 01000000 |
| **8** | GPA7 | 8 | A7 | 0x12 | 0x80 | 128 | 10000000 |
| **9** | GPB0 | 21 | B0 (B7) | 0x13 | 0x01 | 1 | 00000001 |
| **10** | GPB1 | 22 | B1 (B6) | 0x13 | 0x02 | 2 | 00000010 |
| **11** | GPB2 | 23 | B2 (B5) | 0x13 | 0x04 | 4 | 00000100 |
| **12** | GPB3 | 24 | B3 (B4) | 0x13 | 0x08 | 8 | 00001000 |
| **13** | GPB4 | 25 | B4 (B3) | 0x13 | 0x10 | 16 | 00010000 |
| **14** | GPB5 | 26 | B5 (B2) | 0x13 | 0x20 | 32 | 00100000 |
| **15** | GPB6 | 27 | B6 (B1) | 0x13 | 0x40 | 64 | 01000000 |
| **16** | GPB7 | 28 | B7 (B0) | 0x13 | 0x80 | 128 | 10000000 |

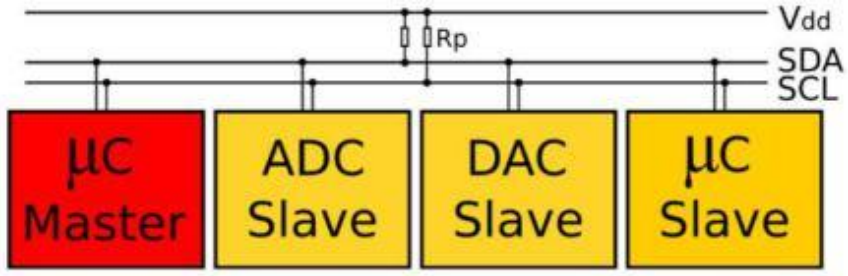
Note: The first batch of Ciseco expander boards have B0-B7 labelled the wrong way round. The numbers in brackets are the incorrect labelling. Watch out for this. Later batches of this board should be correct.

If for example you wish to enable I/O 9 (GPB0) and 10 (GPB1) together you must enable bank 1 and then either add the decimal values together. 1 + 2 = 3 = Hex 0x3 = binary 00000011.

# The I2C interface

The **I²C** (Inter-Integrated Circuit, referred to as I-squared-C, I-two-C, or IIC) is a multi-master serial single-ended computer bus invented by Philips used for attaching low-speed peripherals to a motherboard, embedded system, cell-phone, or other electronic device.

The following diagram shows a typical architecture for an I2C system. There is always a master device , in this project this is the Raspberry PI. There can be up to eight slaves (Addressed by A0, A1 and A2). In this project the slave device is the Ciseco MCP23017 expander board.




- **SCL (Serial Clock) line** is the clock line, the clock is generated by the master and drives the communication in both directions, and this line is an input to all slaves.
- **SDA (Serial Data) line** is the master data output, slave data input, and it caries data from the master to the slave.
- **$V_{dd}$ (Power) line** is a share 5V power line.
- **$R_p$** are 1.8K pull-up resistors
- 128 possible addresses with 16 reserved addresses giving you 112 devices max.

Figure 5 The I2C interface

For more information on the **I²C** interface see the following Wikipedia article:

http://en.wikipedia.org/wiki/I%C2%B2C

# GPIO I2C interface

As well as being able to use the GPIO pins directly for input and output some of these have a secondary function as shown in the diagram below. For the I2C interface pins 3 and 5 are used are data (SDA) and clock (SCL) respectively.

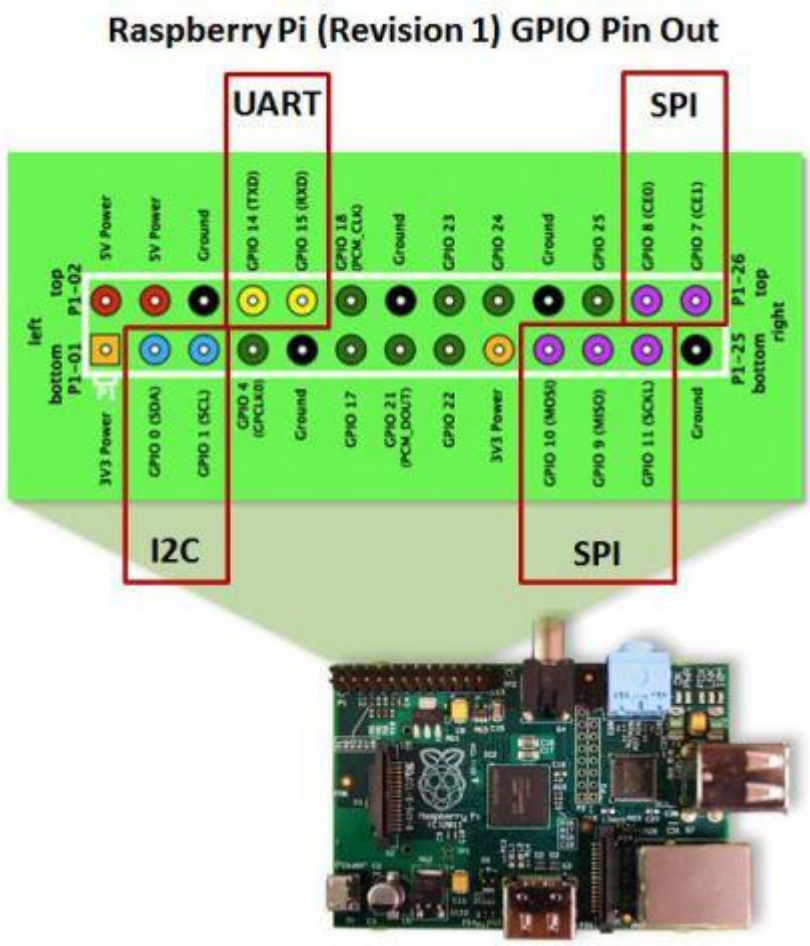



Figure 6 GPIO Pin Wiring for the

Note: On a revision 2 board GPIO 22 is now designated as GPIO 27. For more information on the GPIO see: http://elinux.org/RPi_Low-level_peripherals

# Wiring details

## Wiring LEDs

LEDs have polarity and must be wired correctly to work. The diagram shows the polarity of a typical LED. The longer lead is the positive (+) connection and connects to the Anode (The smaller terminal inside the LED)

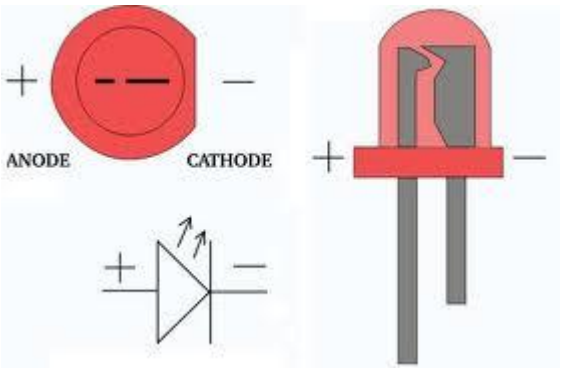



Figure 7 LED polarity

Also the LED must be wired in series with a resistor to limit the current, typically 220 Ohms is OK. Failure to do this will cause the LED to burn brightly for a short while then burn out.

## Switch wiring

One side of the switch connects to the +3.3 Volt supply. The other connects via a 1K resistor to the GPIO input (GP6 on the Ciseco board and pin 22 on the GPIO header)

## Wiring list

The following table shows the GPIO Traffic Lights wiring.

| Ciseco Board | Component wiring |
|---|---|
| **Gnd** | Ground 0 Volts |
| **B0 (B7)** | Green Led --> 100 Ohm --> Gnd |
| **B3 (B4)** | Orange Led --> 100 Ohm -->Gnd |
| **B6 (B1)** | Red Led --> 100 Ohm --> Gnd |
| **B1 (B6)** | Input for switch (3.3v) for the Pedestrian Lights program |
| **A7** | A7 --> 10K Ohm --> GND (Pull down resistor)<br>B1 -->Switch --> A7 |

Note: The first batch of Ciseco expander boards have B0-B7 labelled the wrong way round. The numbers in brackets are the incorrect labelling. Watch out for this. Later batches of this board should be correct.

Wire one side of the switch to the B1 output. Wire the other side of the switch to the A7 input. Also wire a 1KΩ resistor from the A7 side of the switch to GND. Also wire this same side of the switch to the 0V pin via a 10KΩ resistor.

# Exercise 1 – Get one LED working

This section assumes that you have already built the MCP23017 expander board from Ciseco.

## Wire up the green LED

Please refer to Figure 9 Completed prototype board - wiring side on page 14.

1. Break off four 8 pin blocks of the Single In-Line male connectors.
2. Plug the four blocks of pins (long pins) into the two left hand rows of female connectors. This ensures the correct spacing.
3. Place the left hand side of the prototype board over the 32 short pins with the copper tracks side uppermost.
4. Carefully solder all 32 pins to the board.
5. Remove the prototype board from the MCP23017 expander board.
6. Take the green LED and one 100 Ω resistor and insert it into the copper track side of the board with the Anode (longer lead) closest to the B0 (B7) connection. The (B7) reference is because the early boards were mislabelled.
7. Wire and solder the anode of the green LED to the B0 connection (labelled B7).
8. Wire and solder one side of the 100Ω resistor to cathode of the LED.
9. Wire and solder the other end of the resistor to the first earth pin on the left hand side of the board (the black wire in diagram).

We are now ready to test the board with the flashing led program.

## Install the smbus libraries

Before you can run the test programs it is necessary to install the smbus libraries and utilities.

Edit **/etc/modules** file and add the following lines to the end of the file. Then save and reboot to enable the hardware I2C driver.

```
i2c-bcm2708
i2c-dev
```

Enter the following commands to add SMBus support (which includes I2C) to Python:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

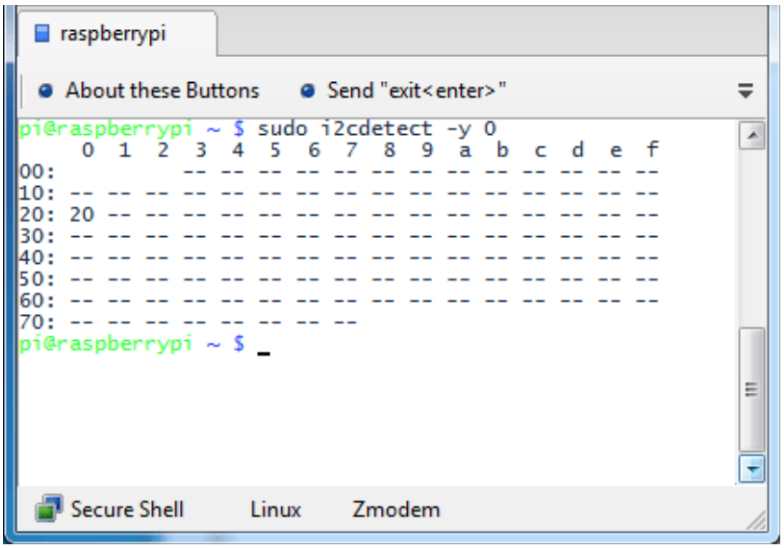
Edit **/etc/modules** file and add the following lines to the end of the file. Then save and reboot to enable the hardware I2C driver.

```
i2c-bcm2708
i2c-dev
```

The i2c-tools isn't strictly required, but it's a useful package since you can use it to scan for any I2C or SMBus devices connected to the Raspberry. If you know something is connected, but you don't know it's 7-bit I2C address, this library has a great little tool to help you find it:

```
$ sudo i2cdetect -y 0 (if you are using a version 1 Raspberry Pi)
$ sudo i2cdetect -y 1 (if you are using a version 2 Raspberry Pi)
```

This will search /dev/i2c-0 or /dev/i2c-1 for all address, and if the MCP23017 I/O Expander is correctly connected, it should show up at **0x20.**




Once both of these packages have been installed, you have everything you need to get started accessing I2C and SMBus devices in Python.

## Flashing LED program

1. Start up your raspberry PI and log in.
2. Make a directory called develop/TrafficLights in the pi home directory and change to it

```
$ mkdir -p develop/TrafficLights
$ cd develop/TrafficLights
```

3. Create a file called i2cLed.py using the i2cLed.py listing in Appendix A.1 on page 16.

```
$ nano i2cLed.py
```

4. Run the i2cLed program

```
$ sudo python i2cLed.py
```

5. The green LED should start flashing on and off. If not troubleshoot why it isn't working. Check wiring and code errors.

# Exercise 2 – Get all LEDs working

## Wire up the red and orange LEDs

1. Using the same techniques used in Exercise 1 wire up the red and orange LEDs to B3 (B4) and B6 (B1) respectively on the Ciseco board.
2. Wire the two 100 resistors between GND and cathodes of the red and orange LEDs.
3. Amend the i2cLed.py program to test the red and orange LEDs.

## Create a program for the standard traffic lights sequence.

1. Copy the i2cLed.py program to a new file called TrafficLED.py

```
$ cp i2cLed.py i2cTrafficLights.py
```

2. Now amend the i2cTrafficLights.py file so that it conforms to the UK traffic light sequence. Start with the red light on. If you get stuck then refer to listing in A.2 on page 16.
3. Run the program to test it

```
$ sudo python i2cTrafficLights.py
```

4. The program should cause the LEDs to flash in the standard UK traffic light sequence. If not debug the problem.

# Excercise 3 Pedestrian Crossing

## Wire the push button

1. Insert the push button into the copper side of the board to the right of the LEDs.
2. Wire and solder one side of the switch to the B1 (B6) input.
3. Wire and solder one side of the 1KΩ resister to the other side of the switch.
4. Solder the other side of the resistor to GND (Connect it to the other resistor GND end)
5. Wire the other end of the resistor to the B1 (B6) side of the switch.

## Run the Pedestrian Crossing program

1. Create a program called i2cPedestrianLights.py from the listing in A.3 on page 17.
2. Run the i2cPedestrianLights.py program.

```
$ sudo python i2cPedestrianLights.py
```

3. The lights should go to green
4. Press the button.
5. The light should go to amber, then red and then the amber should flash 5 times before going back to green again.

# The completed board

The following picture shows the completed Traffic Lights board with pedestrian crossing push button. A 26 way ribbon cable is required to connect the MCP23017 expander board to the Raspberry PI. Note that the components are on the copper track side.

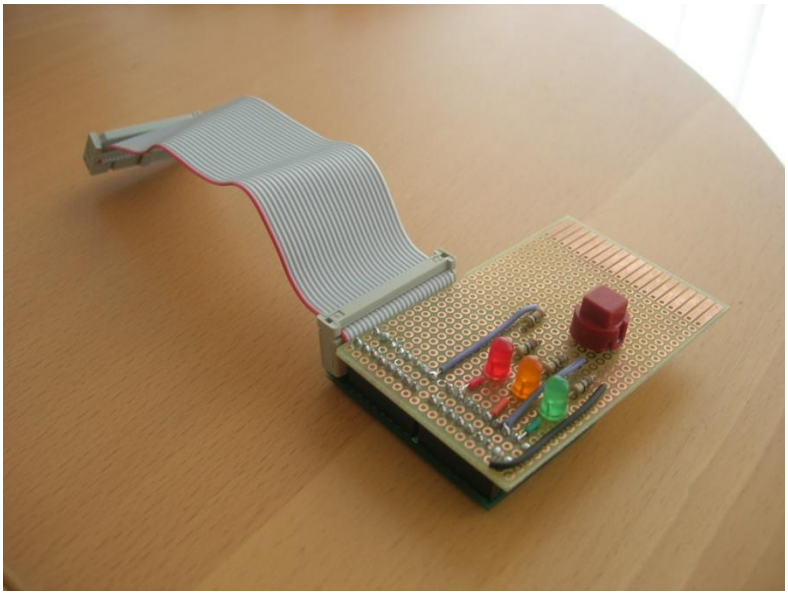


Figure 8 Completed boards - component side

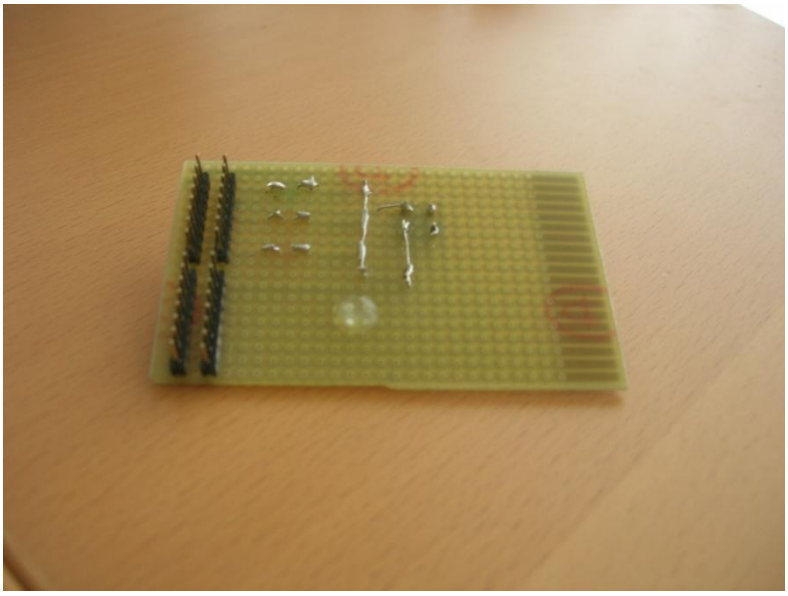


Figure 9 Completed prototype board - wiring side

The following picture shows a single LED plugged into the B0 connector. All of the pins on the left side of the board are GND. On the right is a 20 Single In-Line (SIL20) connector. These connectors can be broken off in blocks of 8 for the prototype connection or in threes to connect a single LED (remove the centre pin with some pliers).

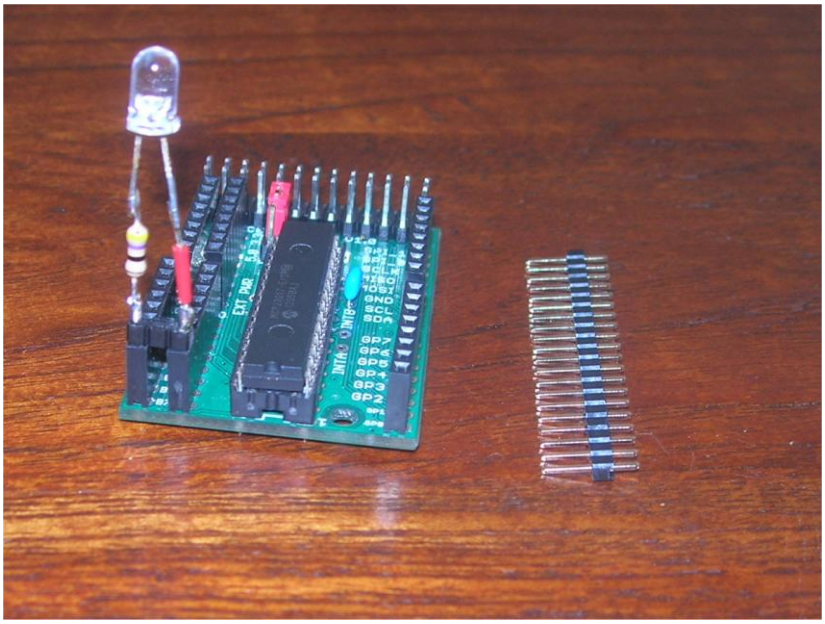



Figure 10 Single LED experiment

# Appendix A – Program listing

## A.1 The flashing I2C LED program – i2cLed.py

```
#! /usr/bin/python

# A simple Python command line tool to control an MCP23017 I2C IO Expander
# Flash a single LED
# Bob Rathbone www.bobrathbone.com
# Adapted from code by Nathan Chantrell http://nathan.chantrell.net

# GNU GPL V3

# SK Pang Electronics June 2012

import smbus
import sys
import getopt
import time
#bus = smbus.SMBus(0) # Revision 1 boards
bus = smbus.SMBus(1)  # Revision 2 boards

address = 0x24 # I2C address of MCP23017 (Can vary - use i2cdetect -y 1)
bus.write_byte_data(address,0x00,0x00) # Set all of bank A to outputs
bus.write_byte_data(address,0x01,0x00) # Set all of bank B to outputs

def set_led(data,bank):
  #print "set_led bank=" + str(bank) + " data=" + str(data)
  if bank == 1:
   bus.write_byte_data(address,0x12,data)
  else:
   bus.write_byte_data(address,0x13,data)
  return

# Handle the command line arguments
def main():
   a = 0
delay = 0.5
while True:
     led = 1
     bank = 0
     set_led(led,bank)
     time.sleep(delay)
     set_led(0,bank)
     time.sleep(delay)

if __name__ == "__main__":
   main()
```

## A.2 The Traffic Lights program – i2cTrafficLights.py

```
#! /usr/bin/python

# A simple Python progam using an MCP23017 I2C IO Expander
# Traffic Light sequencer
# Bob Rathbone www.bobrathbone.com
# Adapted from code by Nathan Chantrell http://nathan.chantrell.net
# License GNU GPL V3

import smbus
import sys
import getopt
```

```
import time
#bus = smbus.SMBus(0)
bus = smbus.SMBus(1)

address = 0x20 # I2C address of MCP23017
bus.write_byte_data(0x20,0x00,0x00) # Set all of bank A to outputs
bus.write_byte_data(0x20,0x01,0x00) # Set all of bank B to outputs

def set_led(data,bank):
  if bank == 1:
   bus.write_byte_data(address,0x12,data)
  else:
   bus.write_byte_data(address,0x13,data)
  return

# Run the traffic light sequence
def main():
   green_led = 1
   amber_led = 8
   red_led = 64
   bank = 0

   while True:
       set_led(green_led,bank)
       time.sleep(3)
       set_led(amber_led,bank)
       time.sleep(3)
       set_led(red_led,bank)
       time.sleep(3)
       set_led(red_led | amber_led,bank)
       time.sleep(3)

if __name__ == "__main__":
   main()
```

## A.3 The Pedestrian Crossing program – i2cPedestrianCrossing.py

```
#! /usr/bin/python

# A simple Python command line tool to control an MCP23017 I2C IO Expander
# Traffic Light sequencer
# Bob Rathbone www.bobrathbone.com
# Adapted from code by Nathan Chantrell http://nathan.chantrell.net

# Licence GNU GPL V3

import smbus
import time
import atexit

# Define bus - 0 for revision 1 boards or 1 for rev 2
#bus = smbus.SMBus(0)
bus = smbus.SMBus(1)

# Define the MCP23017 expander banks
bankA = 0x13
bankB = 0x12

# Define the address of the expander
address = 0x20 # I2C address of MCP23017

bus.write_byte_data(0x20,0x00,0x00) # Set all of bank A to output
bus.write_byte_data(0x20,0x00,0x80) # Set  A7 of bank A to input
```

```
bus.write_byte_data(0x20,0x01,0x00) # Set all of bank B to outputs

# Register exit routine
def finish():
       set_led(0,bankA)
       set_led(0,bankB)
       print("Program stopped")

atexit.register(finish)

# Write a byte to either bank A or B
def set_led(data,bank):
  ##print "bank "+ str(bank) + " data=" + str(data)
    bus.write_byte_data(address,bank,data)
    return

# Main routine for traffic light sequence
def main():
   # Make pin A7 high so we have +3.3v available for the switch
   plus3 = 2

   # Define the LEDs - For green led also make A7 high
   green_led = 1 + plus3
   amber_led = 8
   red_led = 64

   while True:
       set_led(plus3,bankA)
       set_led(green_led,bankA)
       print "Green"
       buttonPressed = False
       while not buttonPressed:
           inp = bus.read_byte_data(address,bankB)
           if inp > 0:
               buttonPressed = True
               print "Button pressed value=" + str(inp)
           else:
               time.sleep(0.25)

       set_led(amber_led,bankA)
       print "Amber"
       time.sleep(3)
       set_led(red_led,bankA)
       print "Red"
       time.sleep(3)

       # Flash the amber light
       count = 5
       print "Flash Amber"
       while count > 0:
           set_led(amber_led,bankA)
           time.sleep(0.5)
           set_led(0,bankA)
           time.sleep(0.5)
           count = count-1

# Tell the program where to start
if __name__ == "__main__":
   main()

# End
```

# Appendix B Licences

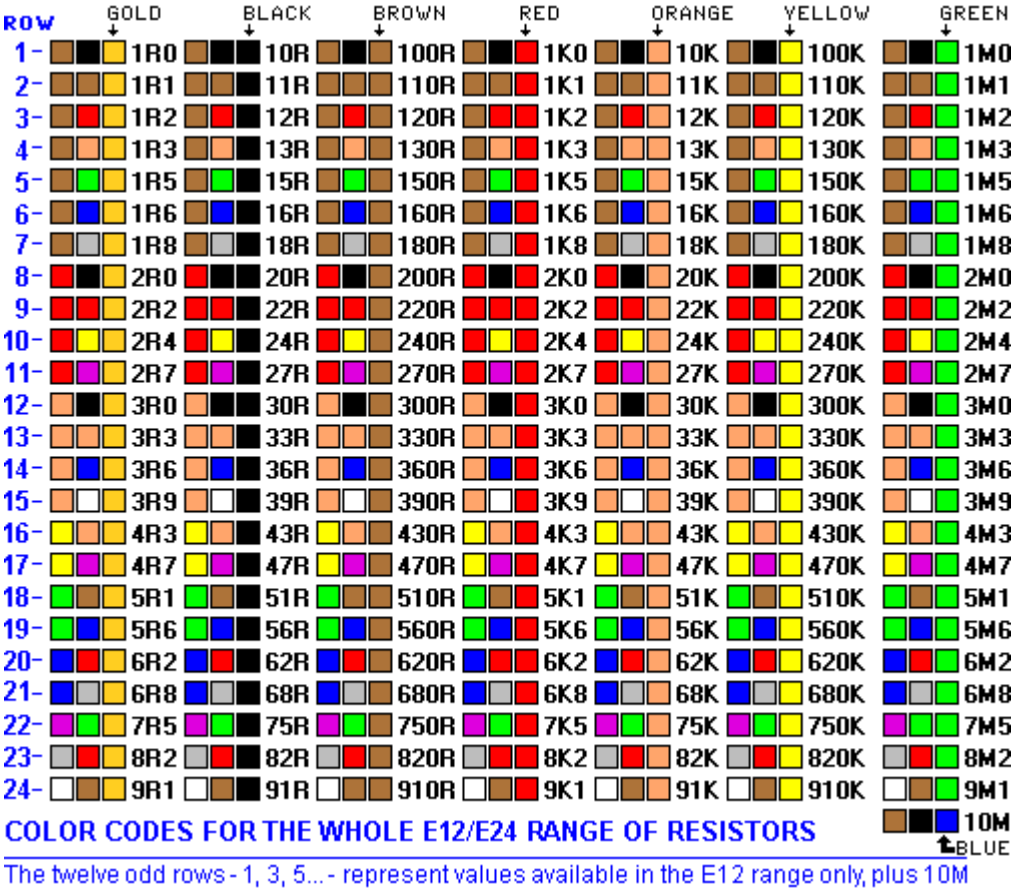
The software and documentation for this project is released under the GNU General Public Licence.

The GNU General Public License (GNU GPL or GPL) is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to use, study, share (copy), and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project.

The GPL grants the recipients of a computer program the rights of the Free Software Definition and uses *copyleft* to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a *copyleft* license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. GPL was the first *copyleft* license for general use.

See http://www.gnu.org/licenses/#GPL for further information on the GNU General Public License.

# Appendix C Resistor Colour codes



# Glossary

| | |
|---|---|
| **GPIO** | General Purpose Input Output |
| **I²C** | Inter-Integrated Circuit – From Philips Corporation |
| **SMBus** | System Management Bus (See I²C) |