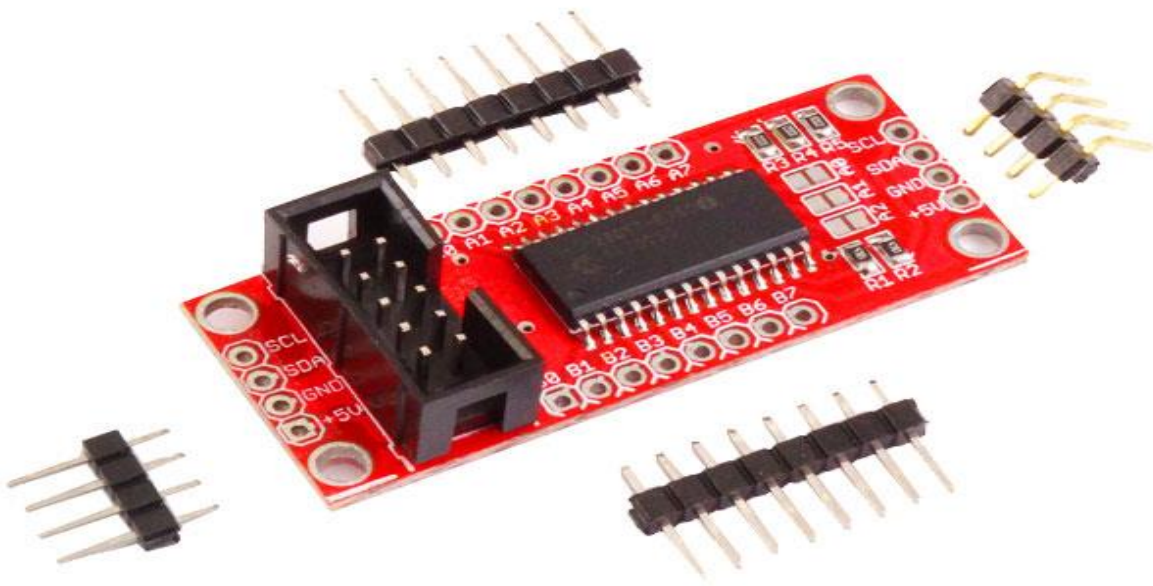




**Research
Design Lab**



I2C Input –Output Port Expander MCP23017

Contents:

| | |
|---------------------------------|---|
| Introduction..... | 3 |
| Features..... | 3 |
| MCP23017 Features..... | 3 |
| Package contains..... | 4 |
| Internal Block connections..... | 4 |
| Working steps..... | 5 |

Introduction:

The IO Expander Board is based on the Microchip MCP23017 Expander Chip. This high-performance IC, allows connection to a range of High-Speed I²C buses, including the standard 100kHz bus, as well as the newer 400kHz and 1.7MHz bus standards.

Features:

- Includes Microchip MCP23017 16-Bit I/O Expander Chip
- Board can be controlled with High-Speed I²C Connection.
- Standard Male Headers for Board Interfacing.
- Suitable for 5V Systems.
- Three hardware address pins to allow up to eight devices on the bus

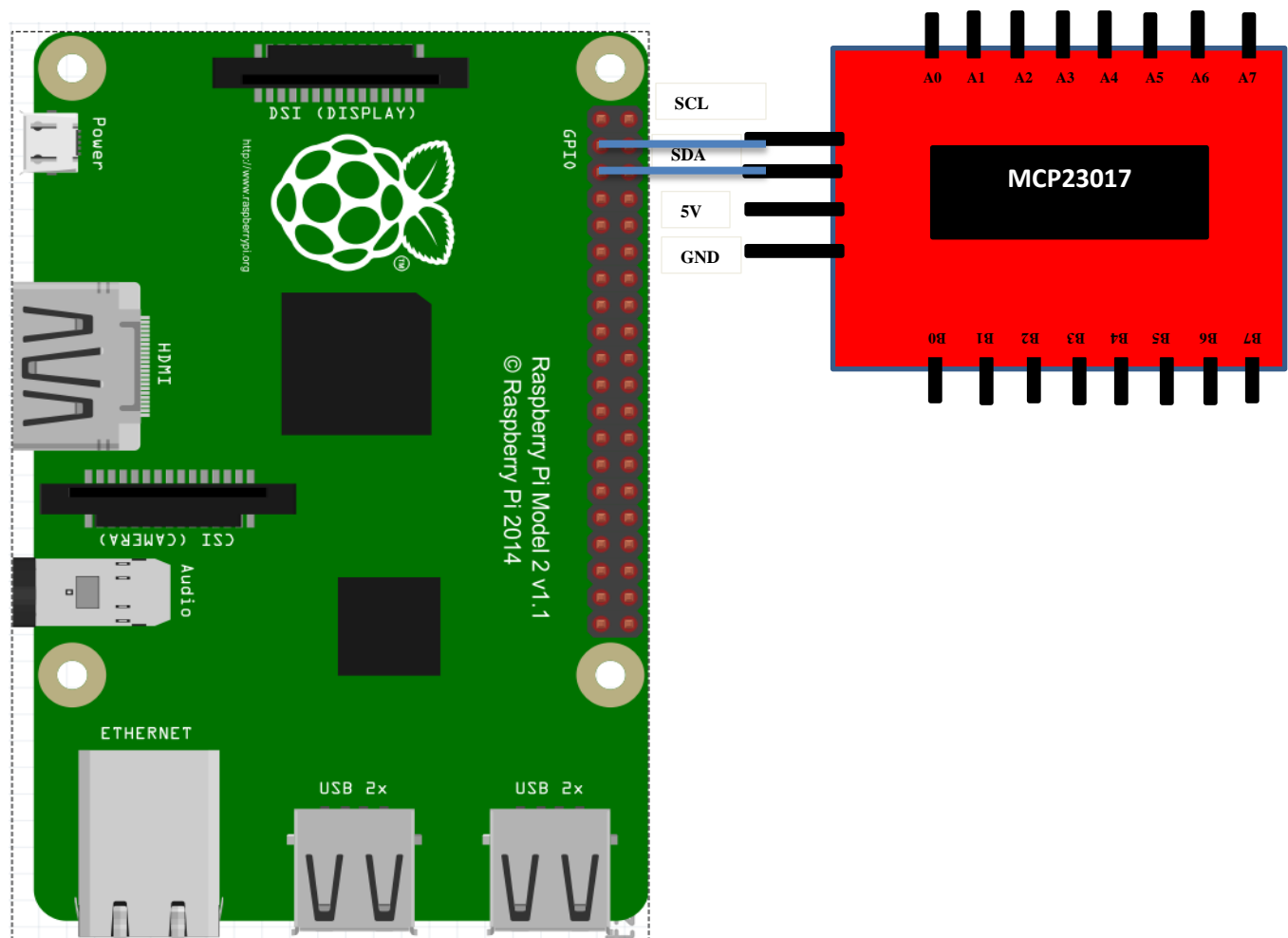
MCP23017 Features:

- 16-bit remote bidirectional I/O port - I/O pins default to input
- High-speed I2C™ interface (MCP23017) - 100 kHz - 400 kHz - 1.7 MHz
- Three hardware address pins to allow up to eight devices on the bus
- Configurable interrupt output pins - Configurable as active-high, active-low or open-drain
- INTA and INTB can be configured to operate independently or together
- Configurable interrupt source - Interrupt-on-change from configured register defaults or pin changes
- Polarity Inversion register to configure the polarity of the input port data
- External Reset input
- Low standby current: 1 μ A (max.)
- Operating voltage: - 1.8V to 5.5V @ -40°C to +85°C - 2.7V to 5.5V @ -40°C to +85°C - 4.5V to 5.5V @ -40°C to +125°C

Package Contains:

- IO Expander Board MCP23017 IC+ Male Burgestick

Internal Block connections:

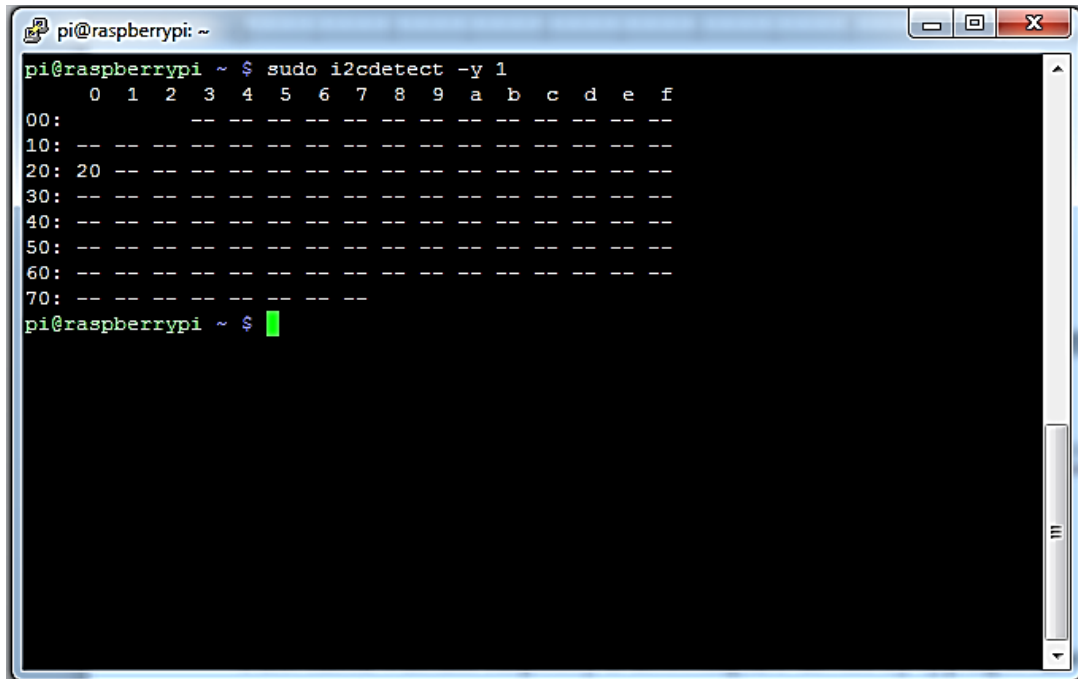


Working Steps:

Steps 1:

First check whether raspberry is detecting I2C device by typing following command.

Sudo i2cdetect -y 1



```
pi@raspberrypi: ~  
pi@raspberrypi ~ $ sudo i2cdetect -y 1  
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20: 20  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
pi@raspberrypi ~ $
```

Step 2:

Configure a port which u want to use by using following command.

Sudo i2cset -y 1 0x20 0x00 0x80

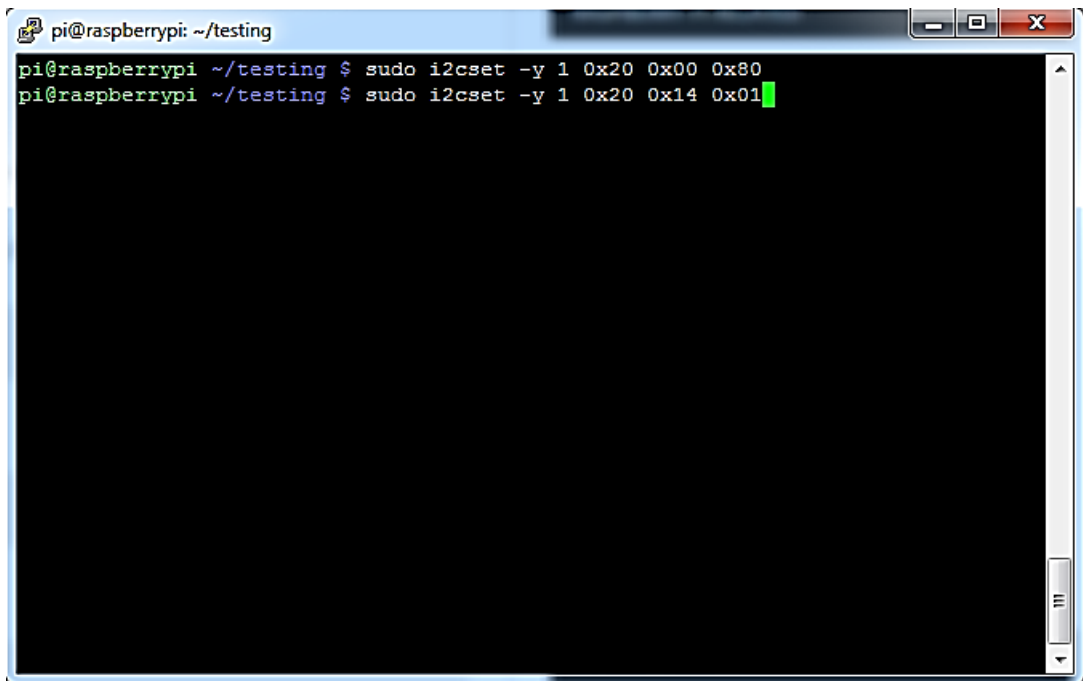
A terminal window titled 'pi@raspberrypi: ~/testing' with a black background. The command 'sudo i2cset -y 1 0x20 0x00 0x80' is entered and highlighted with a green cursor.

```
pi@raspberrypi: ~/testing
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x00 0x80
```

Step 3:

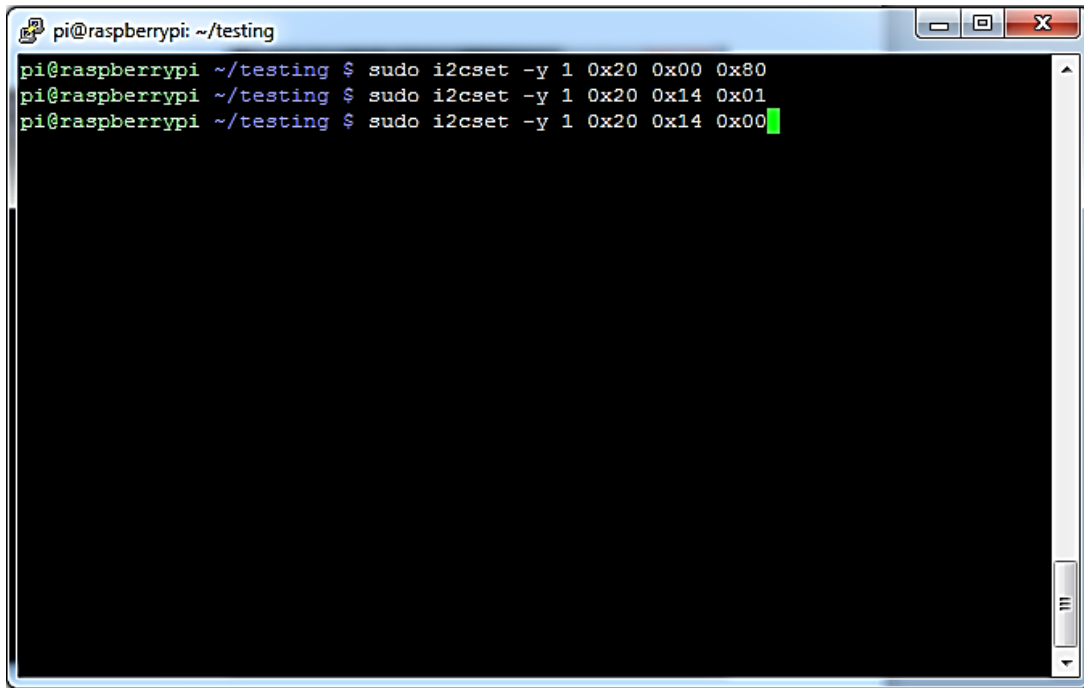
Turn on LED connected to pin7 by using following command.

Sudo i2cset -y 1 0x20 0x14 0x01

A terminal window titled 'pi@raspberrypi: ~/testing' with a black background. Two commands are shown: 'sudo i2cset -y 1 0x20 0x00 0x80' and 'sudo i2cset -y 1 0x20 0x14 0x01'. The second command is highlighted with a green cursor.

```
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x00 0x80
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x14 0x01
```

Turn off led by following command.

Sudo i2cset -y 1 0x20 0x14 0x00

```
pi@raspberrypi: ~/testing
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x00 0x80
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x14 0x01
pi@raspberrypi ~/testing $ sudo i2cset -y 1 0x20 0x14 0x00
```

Python script to count binary 000 to 111 using IO-Expander:

Step 1: open nano editor using command python **filename.py**

Step 2: write a following code in the nano editor as shown below.

```
import smbus

import time

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0

bus = smbus.SMBus(1) # Rev 2 Pi uses 1

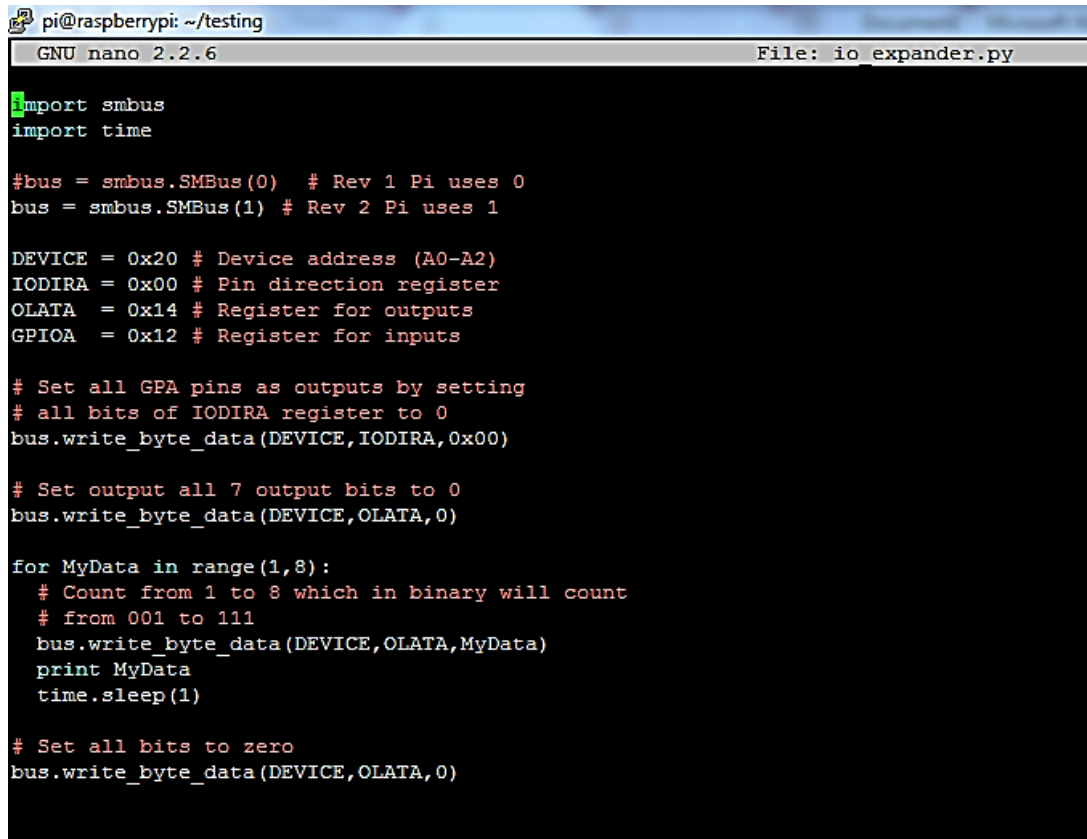
DEVICE = 0x20 # Device address (A0-A2)

IODIRA = 0x00 # Pin direction register

OLATA = 0x14 # Register for outputs

GPIOA = 0x12 # Register for inputs
```

```
# Set all GPA pins as outputs by setting
# all bits of IODIRA register to 0
bus.write_byte_data(DEVICE,IODIRA,0x00)
# Set output all 7 output bits to 0
bus.write_byte_data(DEVICE,OLATA,0)
for MyData in range(1,8):
    # Count from 1 to 8 which in binary will count
    # from 001 to 111
    bus.write_byte_data(DEVICE,OLATA,MyData)
    print MyData
    time.sleep(1)
# Set all bits to zero
bus.write_byte_data(DEVICE,OLATA,0)
```

```
pi@raspberrypi: ~/testing
GNU nano 2.2.6 File: io_expander.py

import smbus
import time

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0
bus = smbus.SMBus(1) # Rev 2 Pi uses 1

DEVICE = 0x20 # Device address (A0-A2)
IODIRA = 0x00 # Pin direction register
OLATA = 0x14 # Register for outputs
GPIOA = 0x12 # Register for inputs

# Set all GPA pins as outputs by setting
# all bits of IODIRA register to 0
bus.write_byte_data(DEVICE,IODIRA,0x00)

# Set output all 7 output bits to 0
bus.write_byte_data(DEVICE,OLATA,0)

for MyData in range(1,8):
    # Count from 1 to 8 which in binary will count
    # from 001 to 111
    bus.write_byte_data(DEVICE,OLATA,MyData)
    print MyData
    time.sleep(1)

# Set all bits to zero
bus.write_byte_data(DEVICE,OLATA,0)
```

Step 3:

Run above code by typing `sudo python filename.py`