Accessing IO pins of port expander(MCP23017) interfaced with R-Pi

e-Yantra Team

June 11, 2016

Contents

1 Objective **3**

2 Prerequisites **3**

3 Hardware Requirement **3**

4 Software Requirement **3**

5 Theory and Description **4**
5.1 MCP23017 4
5.2 Interfacing MCP23017 and detecting its address 4
5.3 16x2 LCD Display 5

6 Experiments **8**
6.1 Interfacing an LED and a Switch to R-Pi using MCP23017 IC 8
6.2 Interfacing an LCD to an R-Pi using MCP23017 IC 11

7 Appendix **17**
7.1 Raspberry Pi 2 Pin-out Diagram 17
7.2 MCP23017 datasheet 17
7.3 Pinouts of MCP23017 17

8 References **18**

1 Objective

In this tutorial we will learn to access IO pins of a port expander (MCP23017) interfaced with an R-Pi and will also perform some experiments with LED's, switches and LCD interfaced with this IC.

2 Prerequisites

- Python programming skills
- Interfacing an MCP23017 IC with an R-Pi should be known

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B+)
2. MCP23017
3. Power adapter
4. Connecting wires
5. LED's
6. 16x2 LCD display (I will be using model JHD162A)
7. Push button
8. Resistors (330 ohms)
9. Potentiometer (10k ohms)
10. Bread board

4 Software Requirement

1. PyScripter (version 2.7 or above)
2. Mobaxterm (for windows user)

5 Theory and Description

5.1 MCP23017

It is a 16 bit I/O expander with serial interface. Some of its features are:

- 16-bit remote bidirectional I/O port (I/O pins default to input)
- High-speed I2C interface (100 kHz, 400 kHz, 1.7 MHz)
- Three hardware address pins to allow up to eight devices on the bus
- Two pins to communicate with a master controller i.e. SCL and SDA

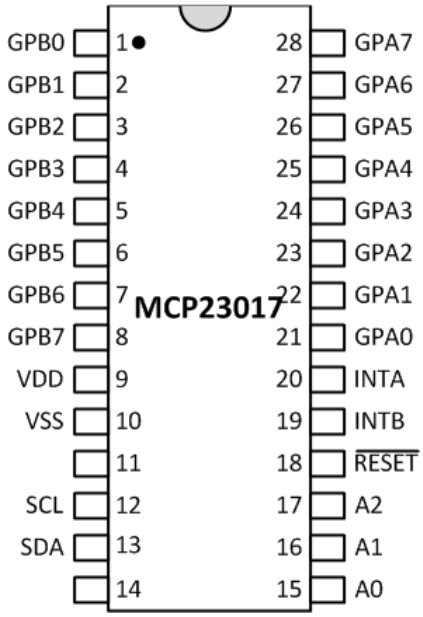

Figure 1: [5]

For further information you refer the Appendix section.

5.2 Interfacing MCP23017 and detecting its address

In order to start programming the port expander(MCP23017) , we need to interface the IC with R-Pi in the following way:

- Connect the VDD pin and RESET on MCP23017 to 3.3v pin on an R-Pi

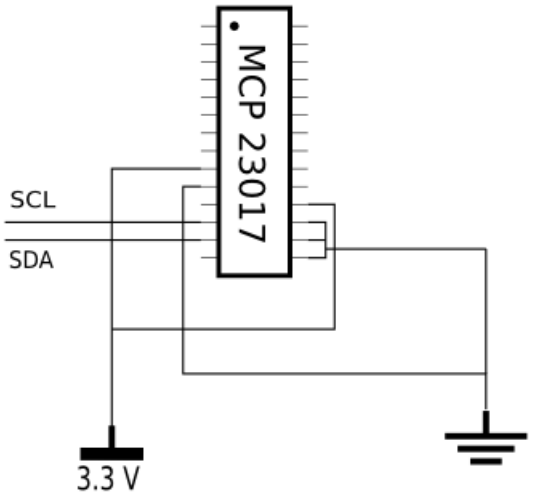

Figure 2: [6]

- Connect the VSS pin on MCP23017 and A0,A1,A2 pin to GND pin on an R-Pi(If you are using 1 port expander IC then we connect the 3 address pins A0, A1 and A2 pins to GND)
- Connect the SCL and SDA pins from the MCP23017 IC to the respective pins on R-Pi

Please refer the Appendix section before making the necessary connections.

After making all the necessary connections type the following command on the terminal *sudo i2cdetect -y 1* (to see the device addresses of MCP23017 IC's interfaced)

Note : For a version 1 R-Pi with RAM lesser than 512MB use the command *sudo i2cdetect -y 0*

5.3 16x2 LCD Display

LCD (Liquid Crystal Display) (JHD162A) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special and even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCD means it can display 16 characters per line and there are 2

such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The *Command* register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The *Data* register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. [4] **Pin**

Diagram

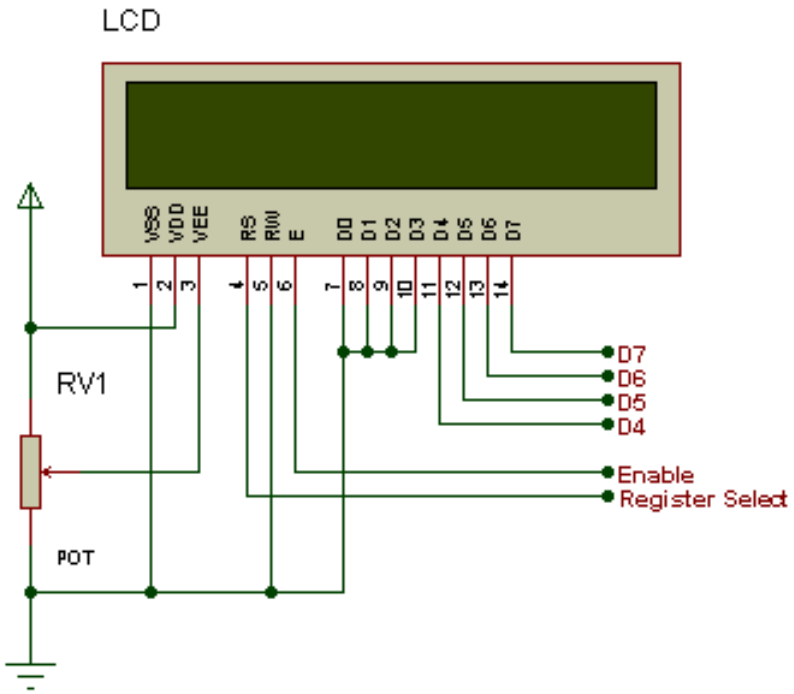

Figure 3: [5]

Pin Description

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	VEE
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7		DB0
8		DB1
9		DB2
10	8-bit data pins	DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight VCC (5V)	Led+
16	Backlight Ground (0V)	Led-

Ref: [4]

6 Experiments

In order to program the MCP23017 chip (using I2C protocol) in Python you must install the **smbus** package. Once you have installed the package you can now start programming the chip.

SMBus protocol commands:

- SMBus Read Byte: i2c_smbus_read_byte_data() This reads a single byte from a device, from a designated register.
- SMBus Write Byte: i2c_smbus_write_byte_data()

  This writes a single byte to a device, to a designated register.

6.1 Interfacing an LED and a Switch to R-Pi using MCP23017 IC

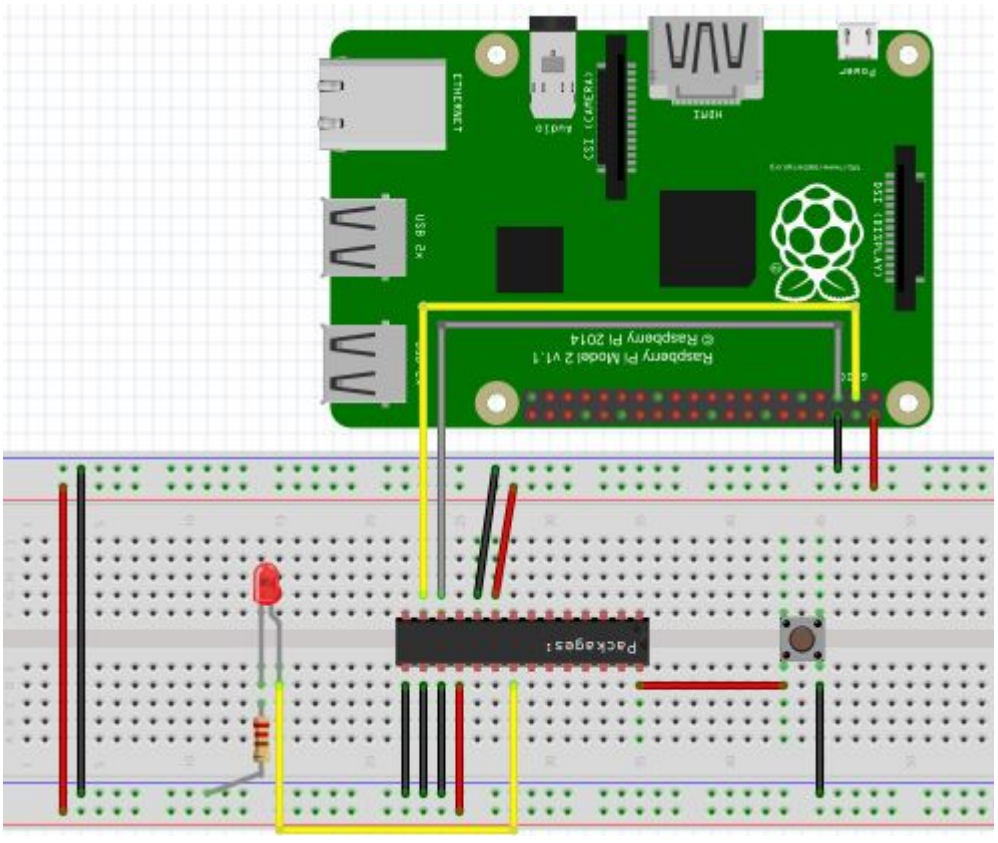

Figure 4: [2]

As shown in the figure:

- Pin 9 (VDD) is connected to 5V
- Pin 10 (VSS) is connected to Ground
- Pin 12 (SCL) is connected to Pin 5 on the Pi GPIO
- Pin 13 (SDA) is connected to Pin 3 on the Pi GPIO
- Pin 18 (Reset) should be set high for normal operation so we connect this to 5V
- Pins 15, 16 & 17 (A0-A2) determine the number assigned to this device. We are only using one device so we will give it a binary zero by setting all three of these pins to 0 (ground)
- Led is connected to GPA0 and switch is connected to GPA7

Code

```
# LED connnected to GPA0
# Push button to GPA4
import smbus # module to access i2c based interfaces
import time

#define I2C connections
bus = smbus.SMBus(1)
DEVICE = 0x20 # 0x20 is address of slave MCP23017 IC on bus.
              #Using this address bus communicates with MCP23017 IC Devic
              #This address in set by setting the A0,A1,A2 pins of IC to
IODIRA = 0x00
OLATA  = 0x14
GPIOA  = 0x12


# all bits of IODIRA register are set to 0 meaning GPA pins are outputs
bus.write_byte_data(DEVICE,IODIRA,0x00)

# Set all 7 output bits of port A to 0
rw = bus.read_byte_data(DEVICE,GPIOA) & 0x00;
bus.write_byte_data(DEVICE,OLATA,rw)

try:
 while True:
  input = bus.read_byte_data(DEVICE,GPIOA) #read status of GPIO register
                                           #switch status
  if input & 0x80 == 0x80:                 # switch pressed i.e. input =
      rw = bus.read_byte_data(DEVICE,GPIOA) & 0x01
      bus.write_byte_data(DEVICE,OLATA,rw)
      time.sleep(1)
      # Set all bits to zero
      bus.write_byte_data(DEVICE,OLATA,0)

except KeyboardInterrupt:
       pass
       bus.write_byte_data(DEVICE,OLATA,0) # in case of keyboard
                                           # interrupt set port A pins to
```

6.2 Interfacing an LCD to an R-Pi using MCP23017 IC

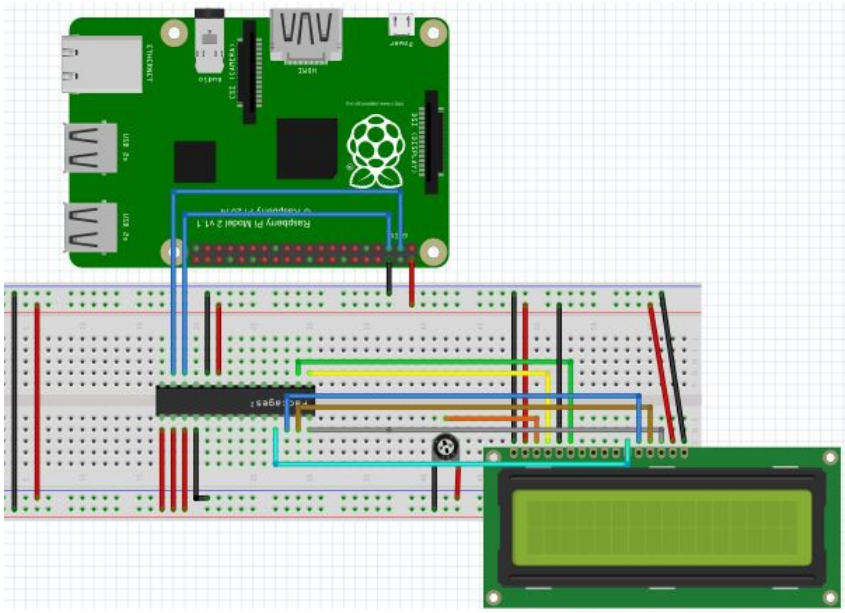

Figure 5: [2]

As shown in the figure:

- Pin 9 (VDD) is connected to 5V (Red)
- Pin 10 (VSS) is connected to Ground (Black)
- Pin 12 (SCL) is connected to Pin 5 on the Pi GPIO
- Pin 13 (SDA) is connected to Pin 3 on the Pi GPIO
- Pin 18 (Reset) should be set high for normal operation so we connect this to 5V (Red)
- Pins 15, 16 & 17 (A0-A2) determine the number assigned to this device. We are only using one device so we will give it a binary zero by setting all three of these pins to 0 (ground) (Black)
- RS and Enable pins of the LCD are connected to GPB0 and GPB1 respectively.
- R/W pin is Grounded(As it has to be used only for write operations).
- Data pins D7,D6,D5 and D4 are connected to GPA7,GPA6,GPA5 and GPA4 respectively.

Code

```
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*
# 4 : RS (Register Select)
# 5 : R/W (Read Write)        - GROUND THIS PIN
# 6 : Enable or Strobe
# 7 : Data Bit 0              - NOT USED
# 8 : Data Bit 1              - NOT USED
# 9 : Data Bit 2              - NOT USED
# 10: Data Bit 3              - NOT USED
# 11: Data Bit 4
# 12: Data Bit 5
# 13: Data Bit 6
# 14: Data Bit 7
# 15: LCD Backlight +5V**
# 16: LCD Backlight GND

#import

import smbus
import time

#define I2C connections
bus = smbus.SMBus(1)
DEVICE = 0x20
IODIRA = 0x00
IODIRB = 0x01
OLATA = 0x14
OLATB = 0x15



# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.0005
```

```
E_DELAY = 0.0005

def set_high_command(value):
  data_r = bus.read_byte_data(DEVICE,OLATB)
  data_w = data_r | value
  bus.write_byte_data(DEVICE,OLATB, data_w)

def set_low_command(value):
  data_r = bus.read_byte_data(DEVICE,OLATB)
  data_w = data_r & value
  bus.write_byte_data(DEVICE,OLATB, data_w)

def set_high_char(value):
  data_r = bus.read_byte_data(DEVICE,OLATA)
  data_w = data_r | value
  bus.write_byte_data(DEVICE,OLATA, data_w)

def set_low_char(value):
  data_r = bus.read_byte_data(DEVICE,OLATA)
  data_w = data_r & value
  bus.write_byte_data(DEVICE,OLATA, data_w)

def main():
  # Main program block
  bus.write_byte_data(DEVICE,IODIRA,0x00)
  bus.write_byte_data(DEVICE,IODIRB,0x00)
  bus.write_byte_data(DEVICE,OLATA,0x00)
  bus.write_byte_data(DEVICE,OLATB,0x00)


  # Initialise display
  lcd_init()

  while True:

    # Send some test
    lcd_string("Rasbperry Pi",LCD_LINE_1)
    lcd_string("16x2 LCD Test",LCD_LINE_2)

    time.sleep(3) # 3 second delay

    # Send some text
    lcd_string("1234567890123456",LCD_LINE_1)
    lcd_string("abcdefghijklmnop",LCD_LINE_2)
```

```
    time.sleep(3) # 3 second delay

    # Send some text
    lcd_string("RaspberryPi-spy",LCD_LINE_1)
    lcd_string(".co.uk",LCD_LINE_2)

    time.sleep(3)

    # Send some text
    lcd_string("Follow me on",LCD_LINE_1)
    lcd_string("Twitter @RPiSpy",LCD_LINE_2)

    time.sleep(3)

def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)

def lcd_byte_char(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command
  set_high_command(0x01)


  # High bits
  set_low_char(0x0F)

  if bits&0x10==0x10:
    set_high_char(0x10)
  if bits&0x20==0x20:
    set_high_char(0x20)
  if bits&0x40==0x40:
    set_high_char(0x40)
  if bits&0x80==0x80:
    set_high_char(0x80)
```

```
    # Toggle 'Enable' pin
    lcd_toggle_enable()

    # Low bits
    set_low_char(0x0F)

    if bits&0x01==0x01:
      set_high_char(0x10)
    if bits&0x02==0x02:
      set_high_char(0x20)
    if bits&0x04==0x04:
      set_high_char(0x40)
    if bits&0x08==0x08:
      set_high_char(0x80)

    # Toggle 'Enable' pin
    lcd_toggle_enable()

def lcd_byte(bits, mode):
    # Send byte to data pins
    # bits = data
    # mode = True  for character
    #        False for command
    set_low_command(0x0FE)


    # High bits
    set_low_char(0x0F)

    if bits&0x10==0x10:
      set_high_char(0x10)
    if bits&0x20==0x20:
      set_high_char(0x20)
    if bits&0x40==0x40:
      set_high_char(0x40)
    if bits&0x80==0x80:
      set_high_char(0x80)

    # Toggle 'Enable' pin
    lcd_toggle_enable()

    # Low bits
    set_low_char(0x0F)
```

```
    if bits&0x01==0x01:
      set_high_char(0x10)
    if bits&0x02==0x02:
      set_high_char(0x20)
    if bits&0x04==0x04:
      set_high_char(0x40)
    if bits&0x08==0x08:
      set_high_char(0x80)

    # Toggle 'Enable' pin
    lcd_toggle_enable()

def lcd_toggle_enable():
    # Toggle enable
    time.sleep(E_DELAY)
    set_high_command(0x02)
    time.sleep(E_PULSE)
    set_low_command(0xFD)
    time.sleep(E_DELAY)

def lcd_string(message,line):
    # Send string to display

    message = message.ljust(LCD_WIDTH," ")

    lcd_byte(line, LCD_CMD)

    for i in range(LCD_WIDTH):
      lcd_byte_char(ord(message[i]),LCD_CHR)

if __name__ == '__main__':

    try:
      main()
    except KeyboardInterrupt:
      pass
    finally:
      lcd_byte(0x01, LCD_CMD)
      lcd_string("Goodbye!",LCD_LINE_1)
```

7 Appendix

7.1 Raspberry Pi 2 Pin-out Diagram

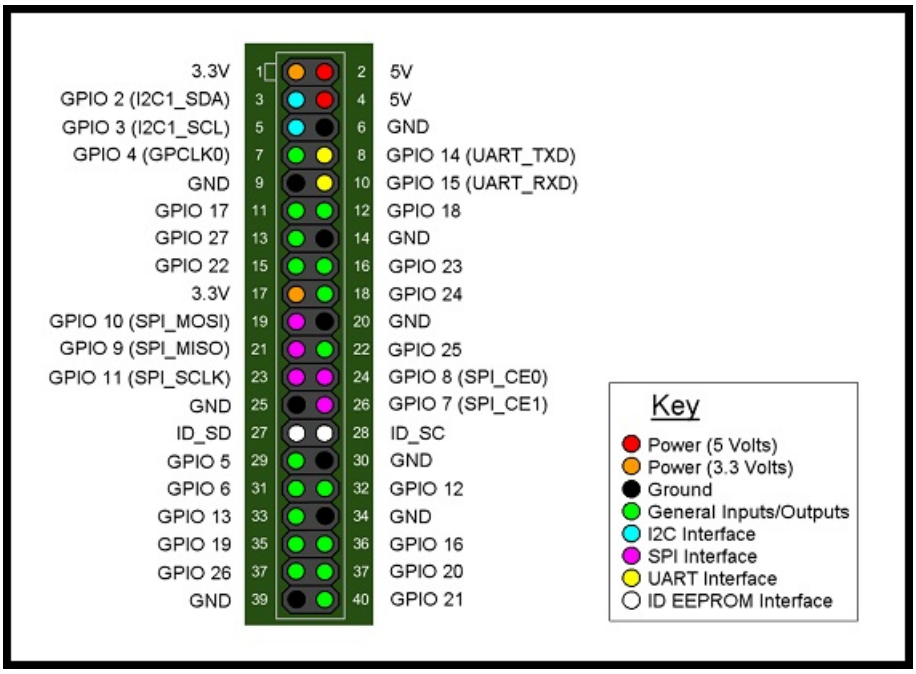

Figure 6: [5]

7.2 MCP23017 datasheet

`http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf`

7.3 Pinouts of MCP23017

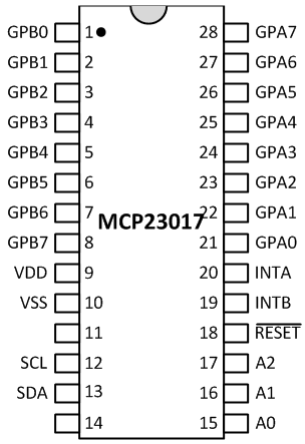

Figure 7: [5]

8 References

1. https://www.kernel.org/doc/Documentation/i2c/smbus-protocol
2. http://dangerousprototypes.com/wp-content/media/2013/04/mcp23017test_bb-600x458.png
3. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
4. http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet
5. http://data.designspark.info/uploads/images/53bc258dc6c0425cb44870b50ab30621
6. https://www.mathworks.com/examples/matlab/4547-add-digital-i-o-pins-to-raspberry-pi-hardware-using-mcp23017