Using the Raspberry Pi & EB080 Interface Board

by John Crow July 2012

Abstract

The EB080 allows users to interface a Raspberry Pi device directly with any of the E-blocks range. In this article we are given an overview of the device and how you can use it to drive any E-blocks attached to the device.

Requirements

Hardware:

- Raspberry Pi
- EB080 Interface board

Warning

Points highlighted in red are where you could cause damage to your hardware if care is not taken to ensure connections are correct.

Remember the Raspberry Pi hardware operates at 3.3V, even though the actual power supply required is 5V.
So except for the 5V line to power the board taken from the Matrix expansion board, signals to all the GPIO lines must not exceed 3.3 V, otherwise the processor on the RPi will be destroyed.

Typed Commands
Commands that need to be typed in the terminal window are shown in green.

Introduction:

Unless you have been living in some remote technologically devoid desert island for the last year J you cannot have missed all the publicity and hype surrounding the release and massive popularity of the Raspberry Pi.

For those who have not heard about it, the Raspberry Pi is an ARM based computer that runs a version of Linux from an SD card.
What's so special about it is, it's only the size of a credit card and it costs about £35.
Only now are they starting to become readily available with a short lead time.
(I waited nearly two months for mine to be delivered.)

Matrix have recently produced a small interface board (EB-080) for the Raspberry Pi
(From here on the Raspberry Pi will be referred to as the RPi).

This has 2 standard female 9 pin D-Connectors, as used in the Matrix E-blocks system.
E-Block hardware can then be interfaced directly to the GPIO port on the RPi.

The board comes with a short ribbon cable that allows the GPIO connector on the RPi to connect to the matching connector on the interface board.

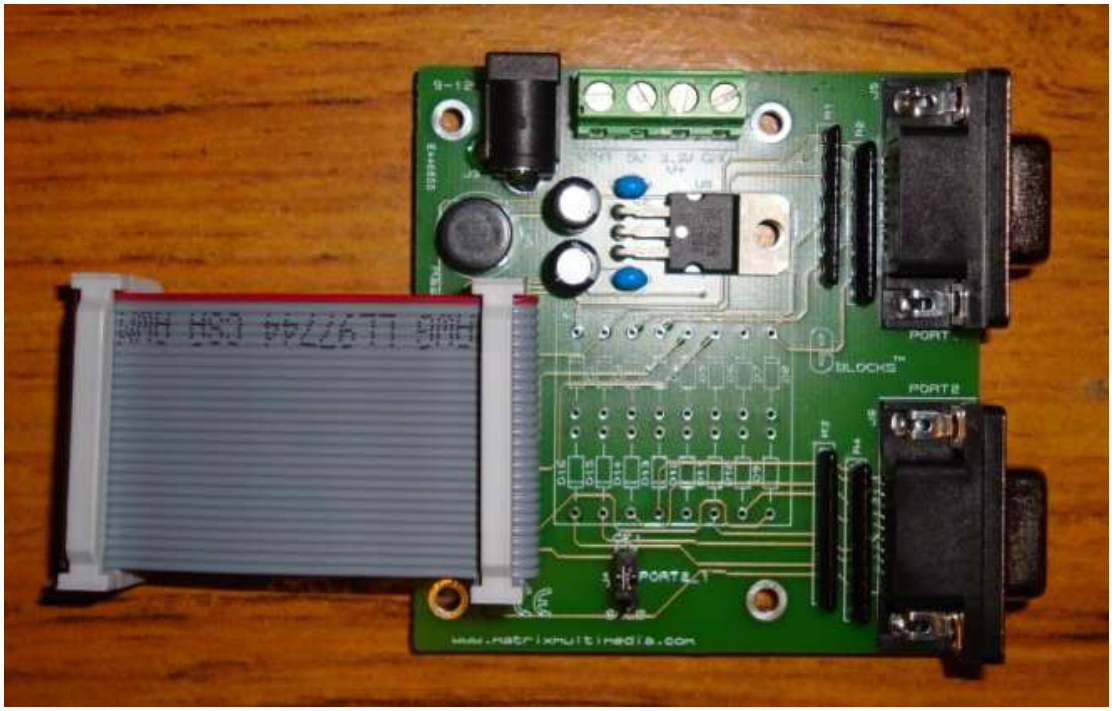
The board can take power from a standard Matrix PSU. Recommended minimum supply current 1A (Matrix suggest setting it to 7.5V so the 3pin regulator on the EB-080 does not have to drop too much voltage.)
The board uses a bridge rectifier as a protection against reverse polarity. This means the power supply can be wired either way round. This does mean a small drop in voltage to the 7805 regulator but not enough to cause any problems.

The E-block powers the RPi as well, so the micro-usb phone charger should not be connected to the RPi power socket.

The EB-080 has the normal screw terminals for 5V, 3.3V and Gnd.
The 3.3V line is taken from the RPi GPIO.

This article will describe some of the basic interface programs I have written to use the EB-080 with some of the more basic e-blocks.
The RPi will support more advanced protocol such as I^2C , SPI and UART, though I just cover the more basic ideas in this tutorial.

This tutorial is based on my own experience with the RPi & EB-080 boards.



The EB-080 Raspberry Pi Interface E-Block

Programs are all written in Python 3.1
I have included Python 2.7 versions of some of the programs.

There is plenty of information on- line information as to the differences in the versions, and as a general tutorial on python. One major difference between versions that will affect users of these projects is the way the print statement is formatted.

Programming IDE's:

The RPi does come with basic shells for both Python2 and Python3.
Idle = Python 2
Idle3 = Python 3
These while suitable for simple program development, are very basic and not user friendly for development of large programs, although are perfectly adequate for the programs included with this tutorial.
My personal choice is for the following programs.

Geany (This is freeware).
http://www.geany.org

This is on both my desktop PC and is also the program I use on the RPi.
Note the original Debian release for the RPi included Geany, however from the Wheezy-Raspian release it has to be downloaded and installed from the internet.

Wingware Personal (This is commercial software and has to be purchased)
http://www.wingware.com

A much more advanced programming environment than Geany.
I use this for developing software on the desktop, before moving to the Rpi.
Both a freeware version and a advanced professional version of this program are also available.
Currently cannot be used on the RPi.

The use of these packages is outside the scope of this tutorial. However they both come with comprehensive documentation.

Installing Geany on the Raspberry Pi:

To install the program use the following command.
You need to be connected to the internet to do this.
The sudo statement means you are running with administrator privileges.

sudo apt-get install geany

This is not aimed at being to Python tutorial, though the example programs show below are well documented, so should be easy for a beginner to follow.

If you prefer to use C my personal choice is Code::Blocks, which is another freeware package.
Both Geany and Code::Blocks are released under the GNU licence.
Code::Blocks includes the GNU compiler.
http://www.codeblocks.org/

sudo apt-get install codeblocks
Setting up the GPIO class.

The GPIO class files need to be downloaded from the RPi website and installed.
Without this Python will not know how to control the GPIO lines.

Download the GPIO program from the RPi website
http://code.google.com/p/raspberry-gpio-python/downloads/list

The versions needed depends on if you are using Python 3 or Python 2
Download and extract the files on the PC and copy the folder to the RPi (Or download direct to the RPi)
Using the terminal window, open the folder created above.
Enter the following command to install the package

sudo python3 setup.py install or **sudo python setup.py install**

Close terminal window

Port Mapping:

The two 9-Pin E-block ports are mapped to the RPi as follows.

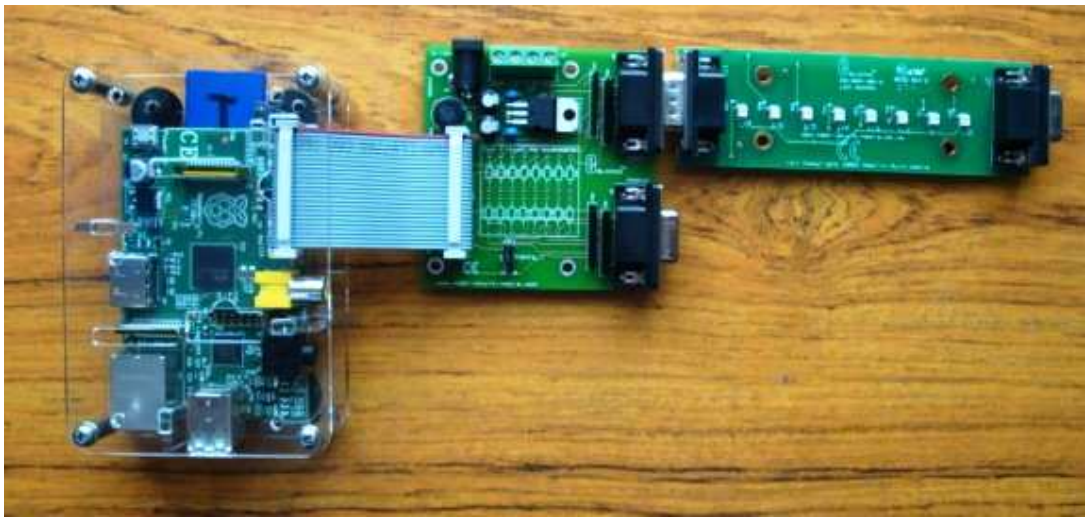
Port 1 pins 1 to 8 = RPi GPIO pins 11, 12, 13, 15, 16, 18, 22, 7

Port2 pins 1 to 8 = RPi GPIO pins 24, 5, 3, 23, 21, 19, 8, 10 (Jumper 1)
Port2 pins 1 to 8 = RPi GPIO pins 24, 26, 3, 23, 21, 19, 8, 10 (Jumper 2)

As can be see pin2 on the 9-pin D-Connector can be connected to either pin 5 or 26 on the GPIO depending on how the jumper is set on the EB-080.
The pins are all protected with 220R resistors.
The lines also have the option of installing 3.3 V zener diodes for OVP protection.
However the safest option is to ensure the lines are not connected to anything over 3.3V.

Take care when making any connections to you RPi either directly or via the EB-080, as wrong connections could destroy your RPi.
Always make connections before applying power to either the RPi or the EB-080, and make sure to only use 3.3V.

Ensure the Red strip on the ribbon cable is connected to pin 1 on both the RPi and the EB-080. If the ports are facing each other as in the photo below then this should be the default connection. The ribbon cable supplied with the GERT-Board is not suitable for the E-Block as Pin 1 is crossed over.



The Raspberry Pi Connected To The EB-080 & EB-004 LED Board

Part 1 - Output Test Programs:

This is a sample of 4 basic programs that shows a simple way to control the output and drive the EB-004 LED Board.
The LED board is connected to Port 1 on the interface board.

Note as presented here the programs won't run on a windows system as they are calling the GPIO hardware on the RPI.

The following 6 programs are intended as a simple demonstration of the RPi and the E-Block adaptor.
The programs can be run either by opening them in Geany and clicking on the run button.
They can also be run by opening a terminal window, change directory to where the file is saved and typing:

sudo python3 filename.py

Where filename is the name of the program you want to run. Note you need to add the .py extension.

Program 1: Binary Counter.

A simple binary counter on the LEDs. Counts from 0 to 255. Then the cycle starts again.

Program 2: Sequencer.

This turns on and off each LED in sequence with a short gap.
Runs 0 to 7, then the cycle restarts.

Program 3: Knight Rider Light Effect.

Generates the "Knight Rider" effect on the LED's by running a single LED back and forth across the LED board. Runs 0 to 7 then 7 to 0. Then the cycle restarts.

Program 4: Simple LED Flasher.

Just turns an LED on and off with a short time delay.

Program 5: Keyboard Selected LED

This is to allow a entry of 0 -7 on the RPi keyboard and will turn the corresponding GPIO line on, hold for a set period of time and then turn it off and wait for the next number.
Error checks for both negative entry and values out of range.

Program 6: Random LED Flasher

This is will generate a random number between 0 and 7.
Using that number the corresponding LED will be turned on for a short period, then off for the same length of time. The program will then generate another random number and repeat.
The number of loops is set in the program.

An alternative method of generating a random number is commented out in the program, to use this just un-comment out the line and comment out the current line.

Part 2 - Input Test Programs:

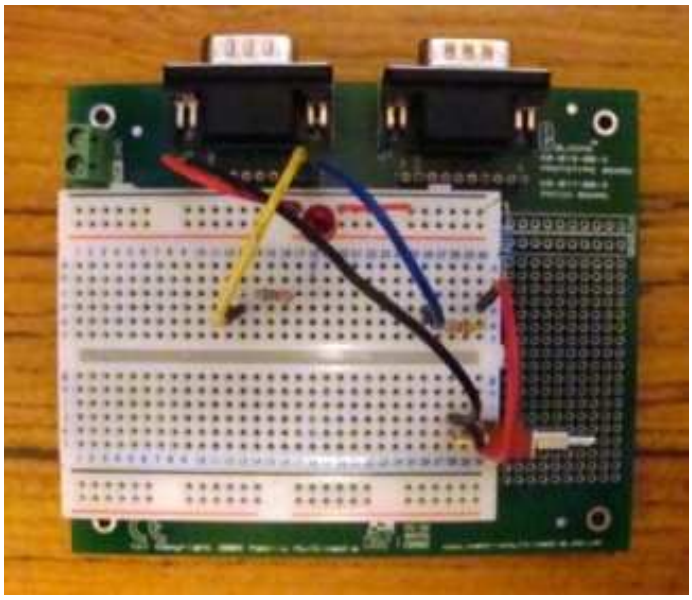
The following are simple programs to demonstrate the input capabilities of the RPi GPIO Ports.
It is important to ensure the Ports are not set up as OUTPUT when applying a signal to them.
The Matrix board has current limiting resistors, which will help prevent damage should this occur, however it is better to ensure this is not needed.
Circuit was built on the EB016 Protoboard.

Program 7a : Simple Switch - LED

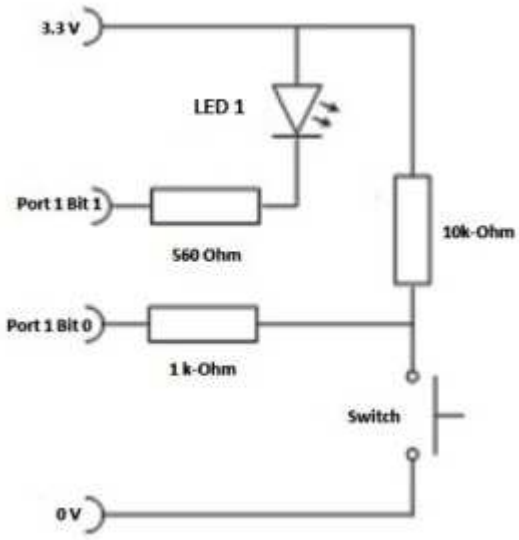
When the switch is closed GPIO 11 goes low (It is normally pulled high by a 10k-ohm resistor to 3.3V)
When this is low (False), GPIO 12 goes low and allows the LED to turn on. I have used a fairly high value resistor on the LED to prevent too much power being drawn from the Rpi. Therefore the LED is fairly dim. Do not use high brightness LEDs for this as they a fairly high current.

Program 7b : Simple Switch - Screen

When the switch is closed GPIO 11 goes low (It is normally pulled high by a 10k-ohm resistor to 3.3V)
Depending on the state of GPIO 11 a message is printed on the screen each time the switch changes state, indicating if it is open or closed.



Simple Switch Circuit On Protoboard.



Simple Switch Circuit Diagram

Program 8a: Matrix Switch Board – LED

This has the Matrix E-Blocks connected to the EB080 as follows.

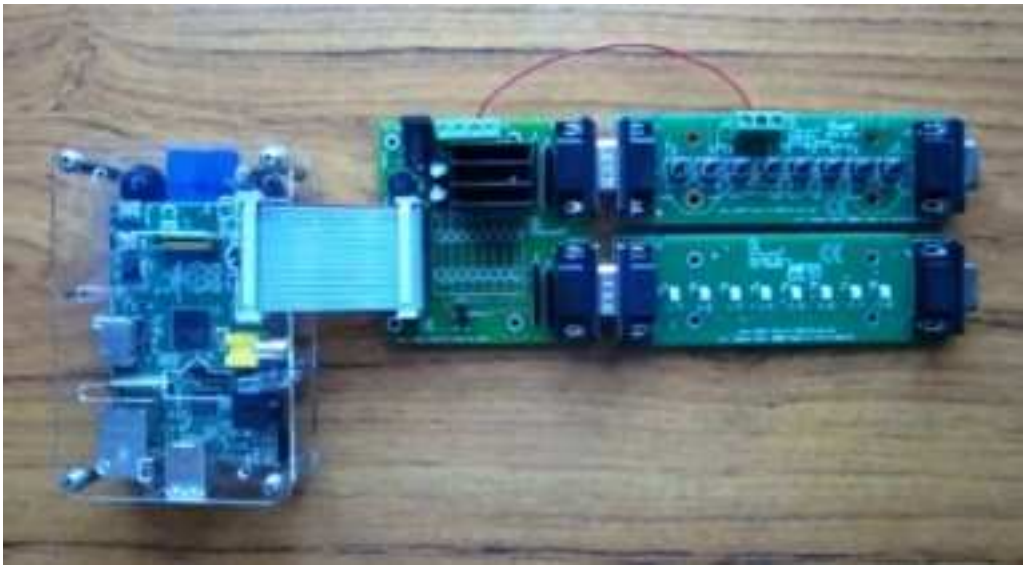
Port 1 = Switch Board – Connect V+ to 3.3V on EB080 (Uses Bit 0)
Port 2 = LED Board

When the switch is open the RPi GPIO 11 is pulled via the 4k7 resistor to 0V . This is read as False.
When the switch is closed, the RPi GPIO 11 is connected to Vdd (3.3V) this is read as True and the LED on GPIO 24 is turned on.
Other LEDS will be in an indeterminate state due to the fact other channels have not been set.

Program 8b: Matrix Switch Board – Screen

Port 1 = Switch Board – Connect V+ to 3.3V on EB080 (Uses Bit 0)

As in version 8a except instead of toggling the LED, a simple message is shown on the screen.
Switch Open or Switch Closed.
The LED E-Block is not needed for this version.



Matrix Switch & LED Boards

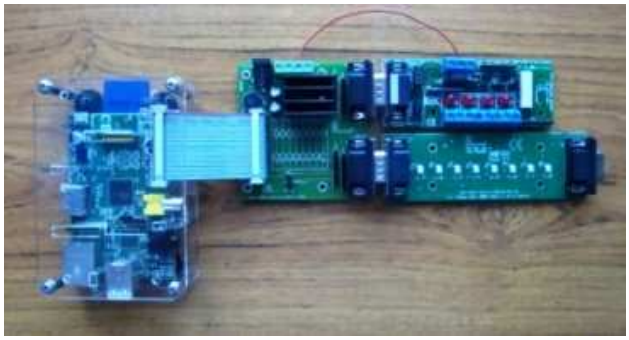
Program 8c: Matrix Opto-Isolator

This has the Matrix E-Blocks connected to the EB080 as follows.

Port 1 = Opto-Isolator Board – Connect V+ to 3.3V on EB080 (Uses Bit 0)
Set jumper to LOW BYTE
Port 2 = LED Board

When a voltage is placed on Ch1 input terminals on the opto isolator board, this causes a signal to be placed on the GPIO 11 input .
The voltage applied to the opto isolated input does not have to be 3.3V. Though I recommend using about this voltage (2x AAs) . That way if an error is made you are much less likely to cause damage to the RPi.
This causes the LED on GPIO 24 to be turned on in the same way as Program 8a.

Other LEDS will be in an indeterminate state due to the fact other channels have not been set.



Matrix Opto-Isolator Board & LED

Conclusions

I have put this article together using my experiences with the hardware and learning to program in Python.
The system is capable of much more complex interfacing such as I^2C, SPI, UART, however the programming for these is much more complex, and this article is aimed as an introduction.

The included programs have all been tested on my hardware.

Possibly, a follow up article on these protocols will follow once my Python knowledge has expanded a bit.

The Rpi is a wonderful little computer, and with the addition of the Matrix EB080 it has an inexpensive way of connecting to the outside world.

Further reading

Below are some links to other resources and articles on related subjects, and technical documentation relating to the hardware used for this project...

E-blocks:	http://www.matrixmultimedia.com/eblocks.php
Raspberry Pi	http://www.raspberrypi.org/
Learning Centre:	http://www.matrixmultimedia.com/lc_index.php
User Forums:	http://www.matrixmultimedia.com/mmforums
Product Support:	http://www.matrixmultimedia.com/sup_menu.php