E-blocks, Raspberry Pi & Android

by Ben Rowland, June 2012

Abstract

Discover how easy it is to use TCP network communications to link together high level systems. This article demonstrates techniques to pass data between embedded devices such as Microcontrollers, Computers, Raspberry Pi and Android.

Requirements

Software:

- Professional or Educational version of Flowcode V4 / V5
- Python 2.7
- Android JAVA SDK with Eclipse

Hardware:

- E-blocks setup with EB023 Ethernet board
- Raspberry Pi with EB080 E-blocks adapter board
- Android Phone or Tablet
- Computer

Introduction

This article explores methods of connecting hardware platforms together using a network such as a local area network (LAN) or via the internet. The hardware covered will be E-blocks, Raspberry Pi, Android and PCs though other devices with network access will also work just as well. Along with the article are examples which act as a guide for creating the required software connections across the network interface. The software connections are handled in the form of TCP servers and clients. Each of the different hardware platforms can work together using the same TCP protocol so this allows you to create any number of flexible systems each tailored to match your requirements.

The main focus of the article is to be able to control the GPIO pins on the Raspberry Pi from a number of different sources but in reality the code could be used for a very wide range of purposes from SCADA control to home automation.

The network

To connect the different hardware platforms together we have a central Wifi enabled router which is physically connected to the EB023 and Raspberry Pi hardware using Ethernet cables. The PC and Android are then connected to the router using Wireless LAN. Every device on the network is assigned an IP address and we can use this IP address to control which device we are communicating with. There is normally a list of connected devices and their associated IP addresses as part of the router management pages. Refer to your router's documentation regarding accessing this information.

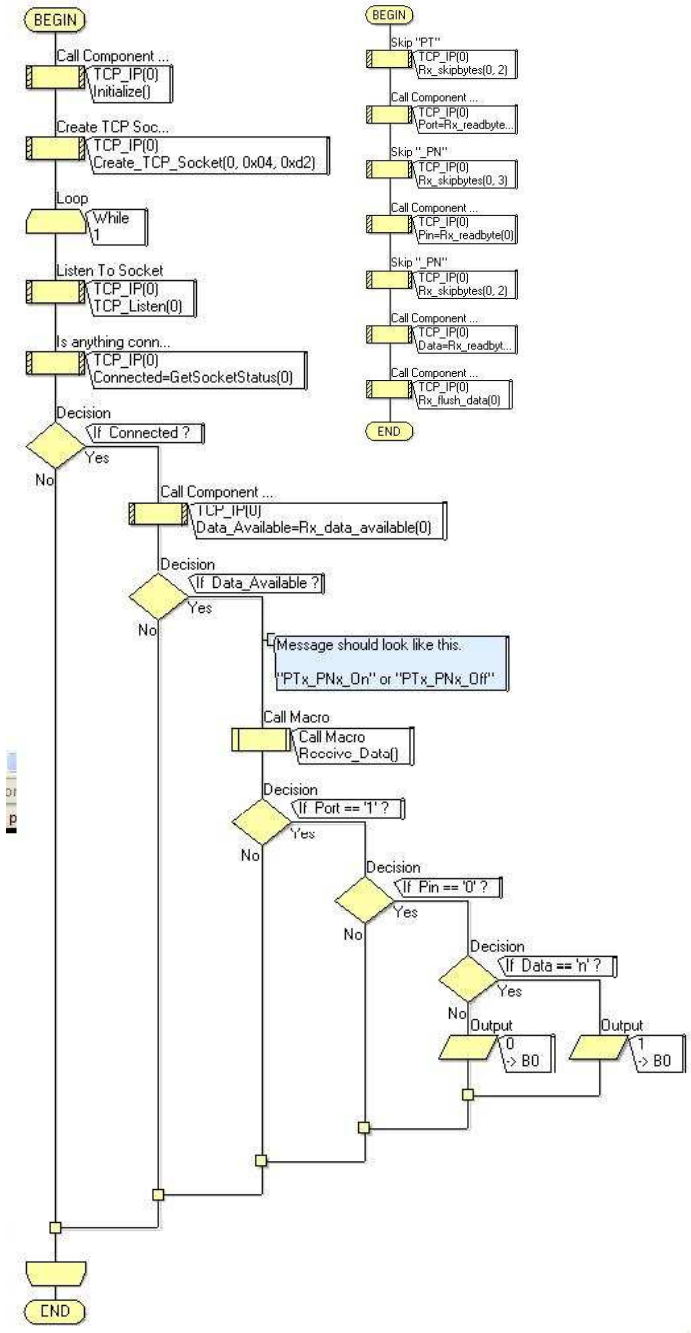
TCP communications using Microcontrollers and Flowcode

The Flowcode TCP/IP component allows for fast and simple access to the TCP protocol using the EB023 E-block.

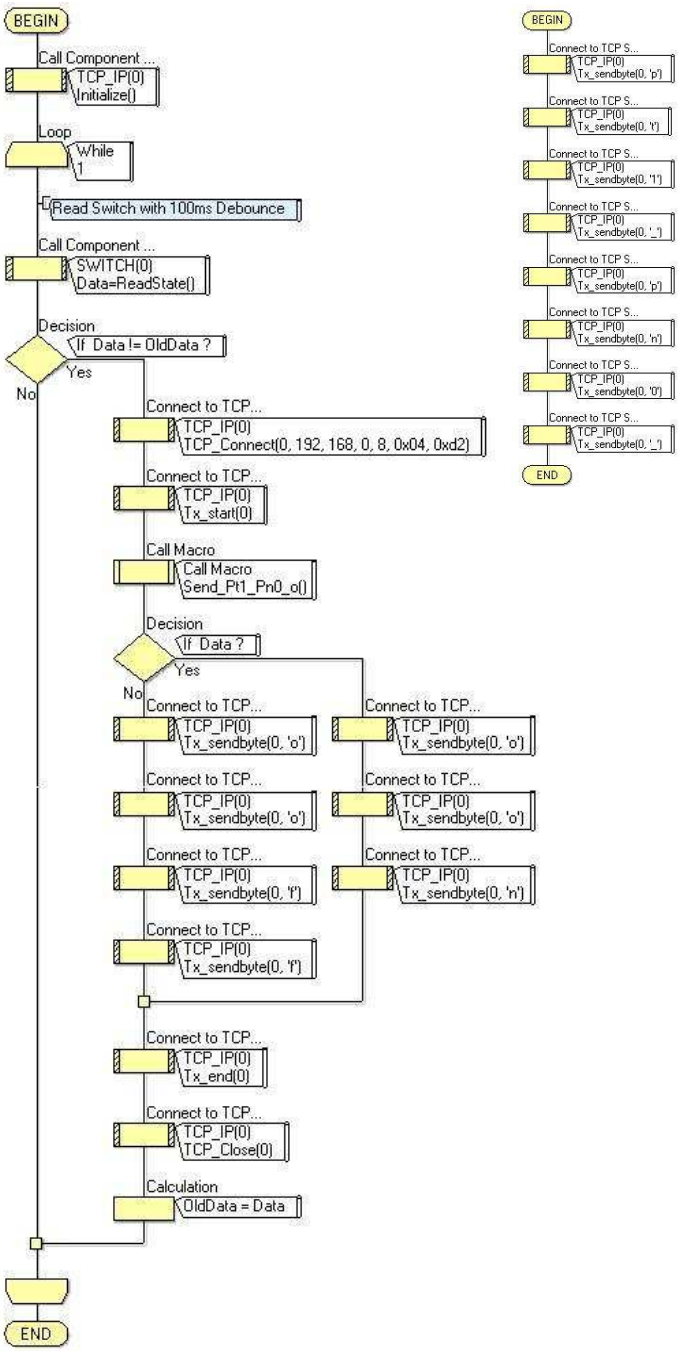
The Flowcode file “EB023 TCP Server” uses the EB023 Ethernet E-block to host a listening server that can switch on or off PORTB pin 0.

The Flowcode file “EB023 TCP Client” uses the EB023 Ethernet E-block to allow you to control a remote server by sending a command whenever a switch connected to PORTB pin 0 changes state.

TCP Server



TCP Client



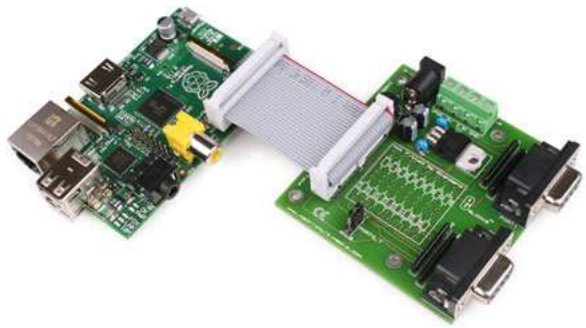
TCP communications using Python

Python is a commonly used scripting language that can be ran on a wide variety of computer systems. This means that the same Python scripts will work on a Windows PC, A Linux PC and also Embedded Linux devices like the Raspberry Pi.

Below is a Python script to create a listening TCP server designed to run on the Raspberry Pi and allow control of the GPIO. The same server code could be changed to run on a computer or other embedded Linux platform. There are even some Android based Python parsers being worked on.

```
import RPi.GPIO as GPIO
import SocketServer

portarray1 = [11,12,13,15,16,18,22,7]
portarray2 = [24,5,3,23,21,19,8,10]

#alt portarray2 setup depending on jumper
#portarray2 = [24,26,3,23,21,19,8,10]

def portout (port, value, mask):
   idx = 0
   while idx < 8:
      tempmask = 0x01 << idx
      if (mask & tempmask == tempmask):
         if (port == 1):
            pin = portarray1[idx]
         if (port == 2):
            pin = portarray2[idx]
         GPIO.setup(pin, GPIO.OUT)
         if (value & tempmask == tempmask):
            GPIO.output(pin, True)
         else:
            GPIO.output(pin, False)
      idx = idx + 1
   return

class EchoRequestHandler(SocketServer.BaseRequestHandler ):
   def setup(self):
      print self.client_address, 'connected!'

   def handle(self):
      data = 'dummy'
      while data:
         data = self.request.recv(1024)
         #self.request.send(data)

         strippeddata = data.strip()

         if strippeddata == 'pt1_pn0_on':
            portout(1,0x01,0x01)
         if strippeddata == 'pt1_pn0_off':
            portout(1,0x00,0x01)

   def finish(self):
      print self.client_address, 'disconnected!'

   #server host is a tuple ('host', port)
server = SocketServer.ThreadingTCPServer(('', 1234), EchoRequestHandler)
server.serve_forever()
```

The Matrix Raspberry Pi E-blocks Adapter EB080

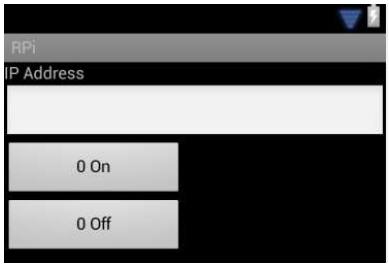


Below is a Python script to create a TCP client which can send control data to the TCP server.

```
import socket
import sys

HOST, PORT = "192.168.1.8", 1234

tx1 = "pt1_pn0_on"
txend = "bye"

# Create a socket (SOCK_STREAM means a TCP socket)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
   # Connect to server and send data
   sock.connect((HOST, PORT))

   sock.sendall(tx1)

   # Receive data from the server and shut down
   #received = sock.recv(1024)

finally:
   sock.close()

print "Sent:    {}".format(tx1)
```

TCP communications using Android

Smart phones and tablets such as Android and iPhone are becoming very popular and as such can be a fantastic way of controlling or interfacing to machinery or processes such as home automation. As the Android SDK is free to use and fully supports TCP communications we have also provided a simple demo for the 4.0.3 release of the Android SDK.



RPi app running in the Eclipse SDK Environment

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...

Flowcode:	http://www.matrixmultimedia.com/flowcode.php
Raspberry Pi:	http://www.raspberrypi.org/
Android:	http://developer.android.com/sdk/installing.html
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