K64 - TCP , UDP , Uart

* TCP to UART
* Uart to TCP
* UDP to Uart
* Uart to UDP

To start you need:

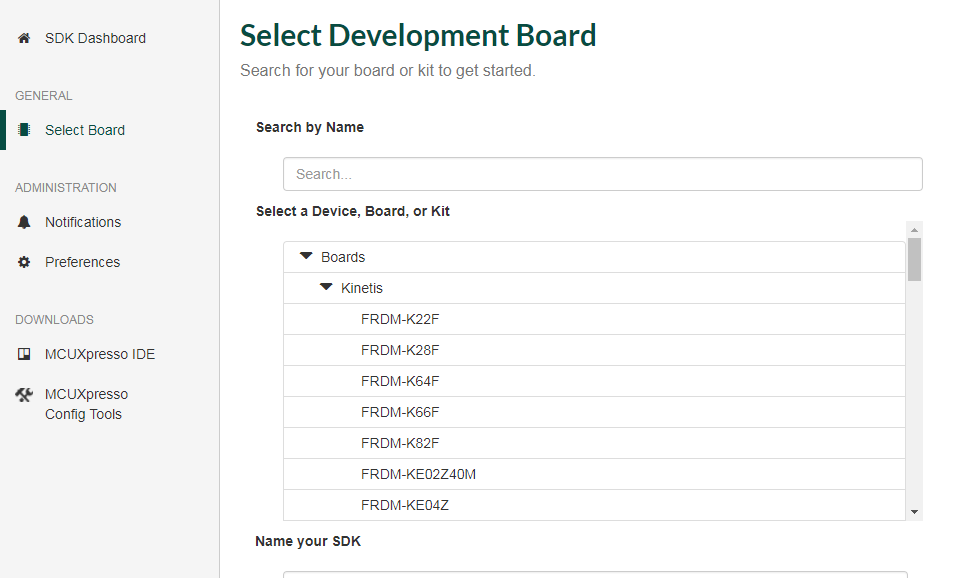
1. Kinetis Design studio 3.0

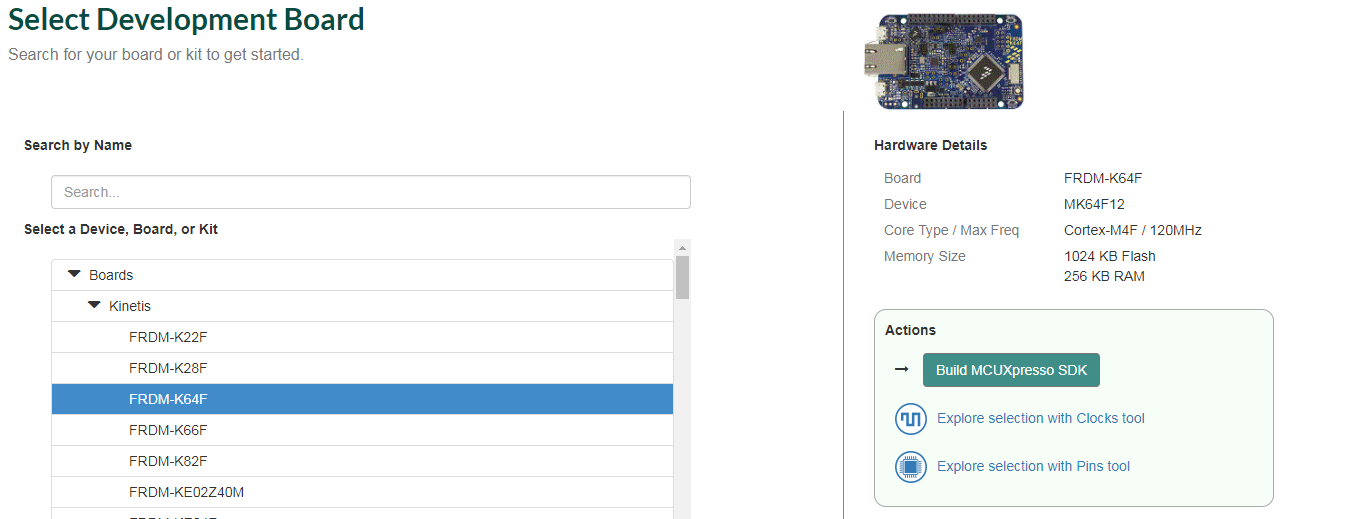
https://community.nxp.com/docs/DOC-330211

2. Kinetis SDK

<https://mcuxpresso.nxp.com/en/welcome>

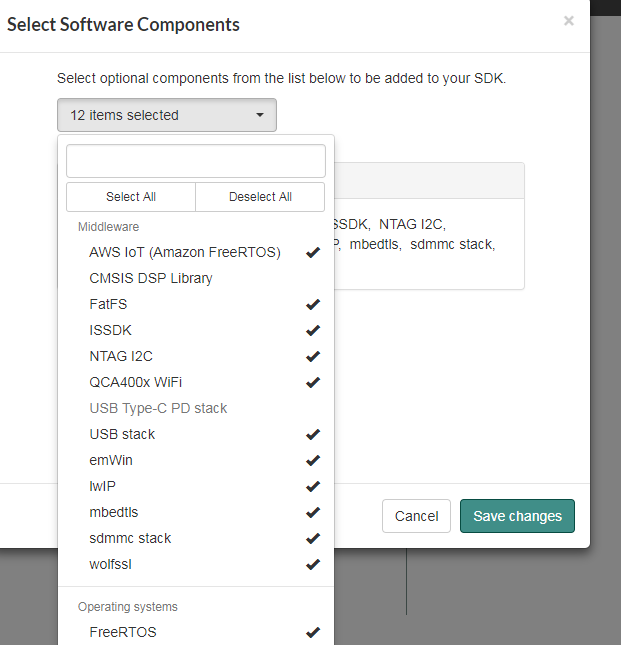
Select a board from the dashboard:



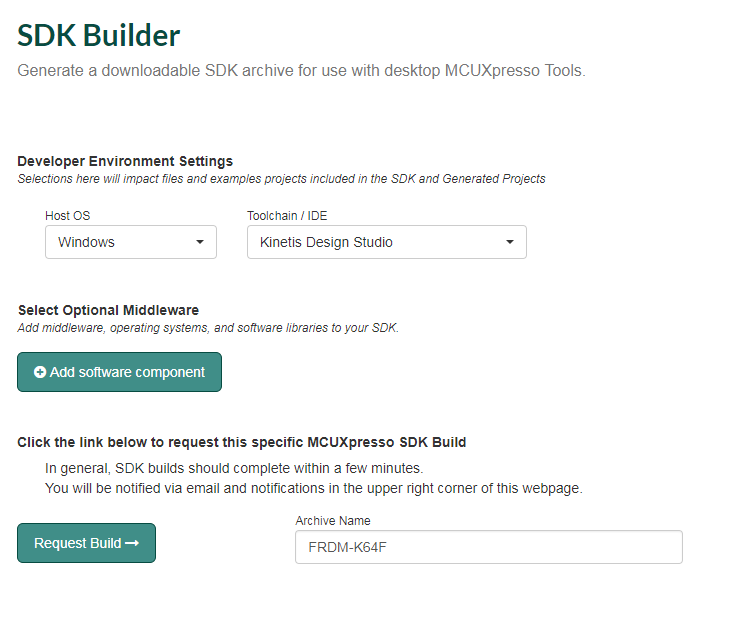


Make sure the selected components include lwip, and free rtos

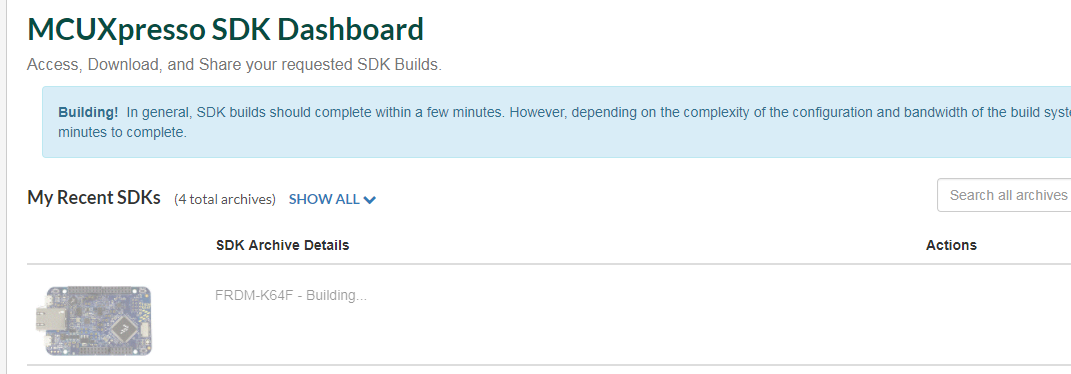
I selected all ☺



We are ready to download our SDk

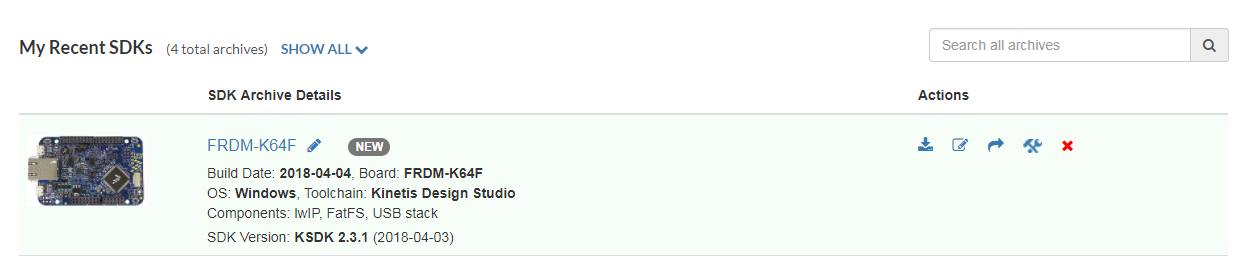


The builder is preparing our build:



It is ready for download

Version 2.3.2



We will build the following projects:

Echo Windows form app in c# TCP -> Uart -> windows form c# app

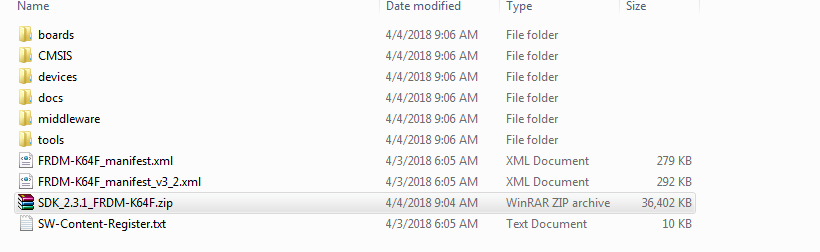
A Windows C# TCP application to K64 to accept some messaging and send back via uart

Echo Windows c# Uart - > k64 -> Windows c# TCP

A Windows C# TCP application to K64 to accept some messaging and send back via uart

Extract the zip file

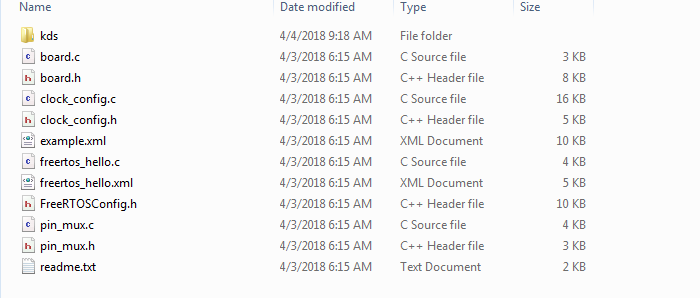
SDK\_2.3.1\_FRDM-K64F.zip



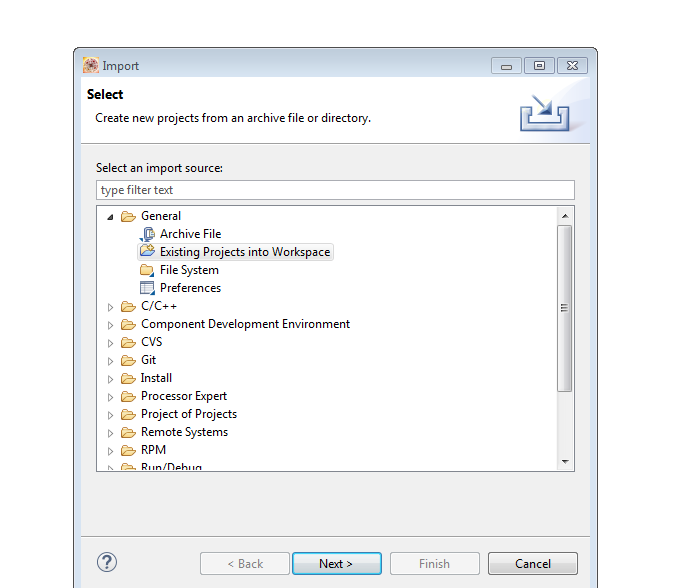
Starting from Hello world – free rtos example

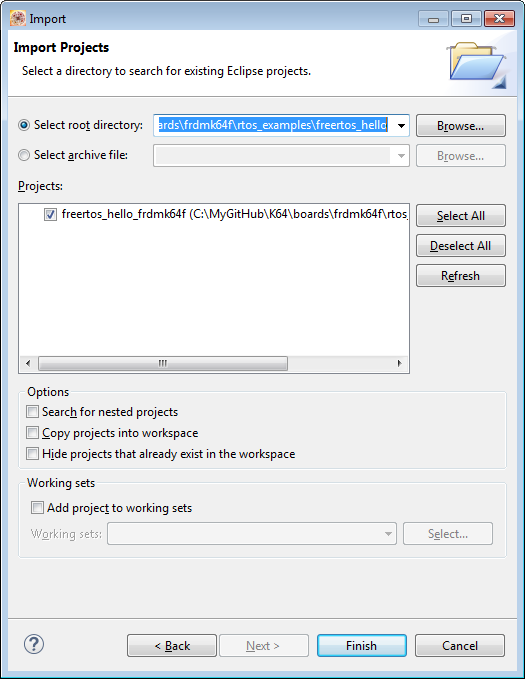
To make sure our board is setup correct, lets build the hello world free rtos example

K64\boards\frdmk64f\rtos\_examples\freertos\_hello



Select in kinetis design studio to import proeject





And click finish.

The K64 support MBED to download firmware to the board.

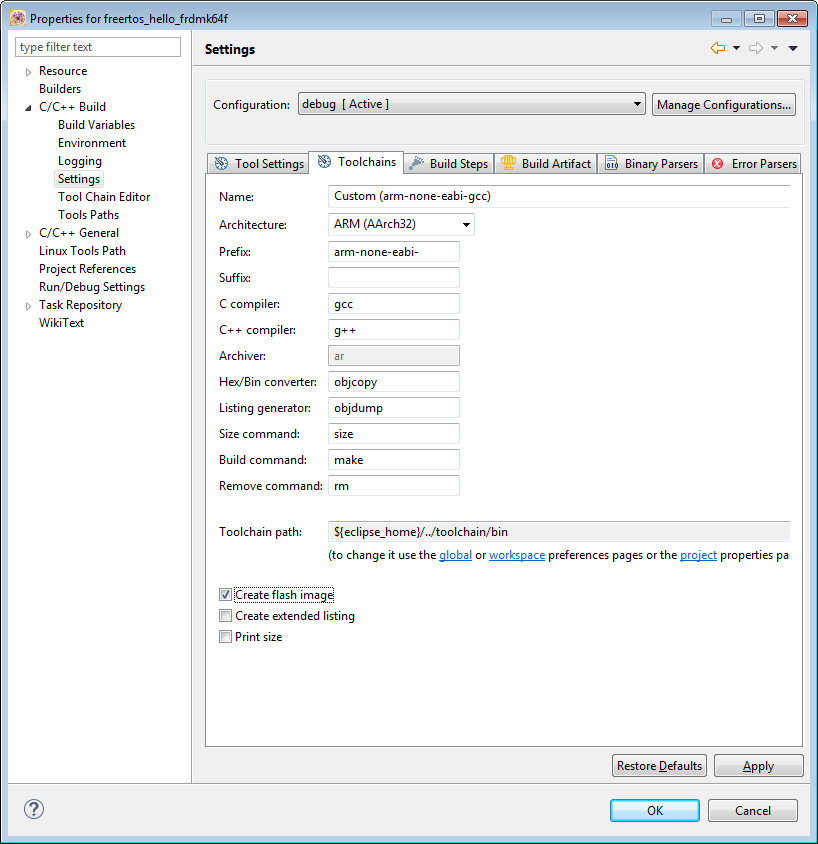
You need to prepare hex file and drag it to the disk drive.

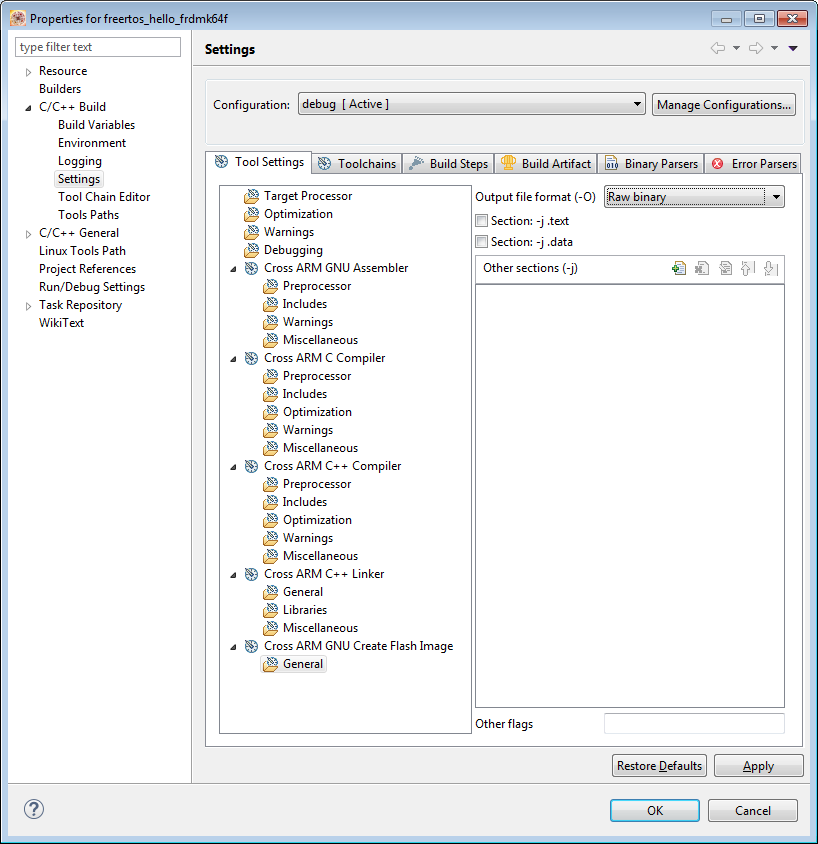
I have another document:

<https://github.com/EliArad/MyDocuments/blob/master/Kinetis_K64/First%20steps%20with%20Kinetis%20SDk.docx>

explaining the steps to compile , download using MBED.

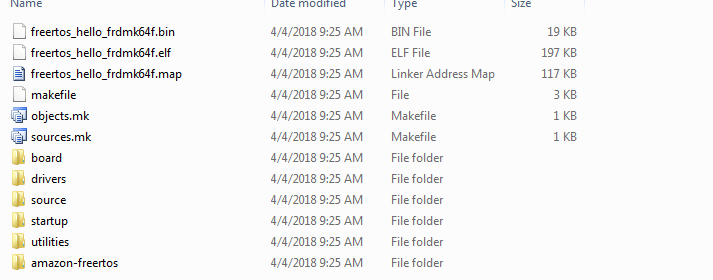
Quick screens:



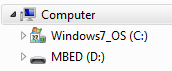


Build the project

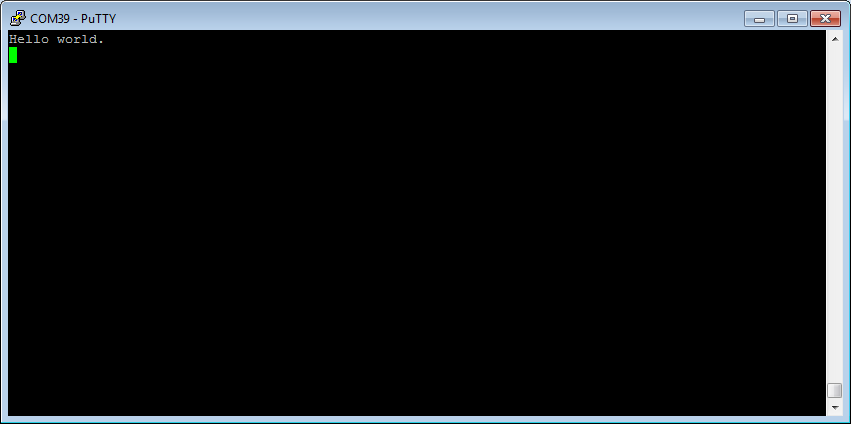
We can see that in the debug directory we have the bin file



This file need to be copied like we copy a file to a disk on key in the computer



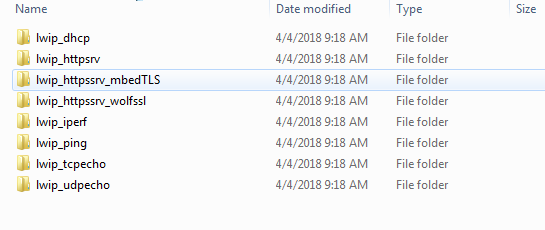
Open putty terminal ( see the first steps doc) and you can see the hello world



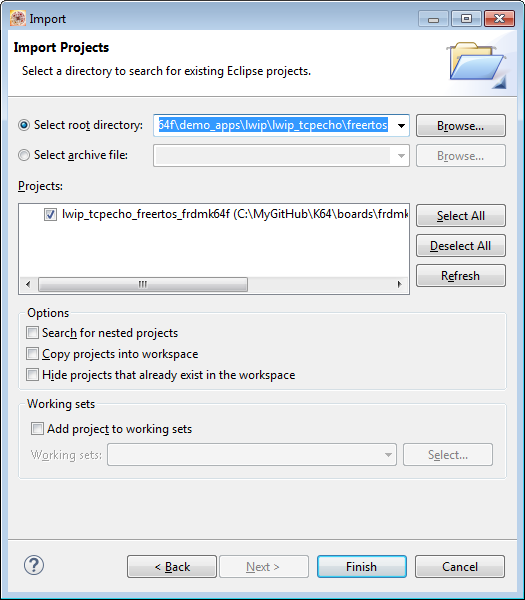
We are ready to continue with out TCP.

Using free rtos or bm LWIP examples are planty:

K64\boards\frdmk64f\demo\_apps\lwip



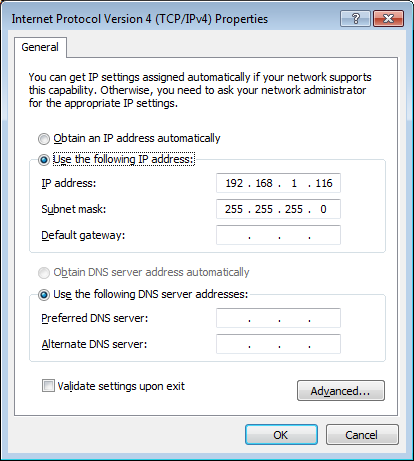
Insert tco echo:



In the main file ,

Change the IP address to your subnet mask

My subnet with a USB to Ethernet convertor is 1.



/\* IP address configuration. \*/

**#define** configIP\_ADDR0 192

**#define** configIP\_ADDR1 168

**#define** configIP\_ADDR2 1

**#define** configIP\_ADDR3 102

/\* Netmask configuration. \*/

**#define** configNET\_MASK0 255

**#define** configNET\_MASK1 255

**#define** configNET\_MASK2 255

**#define** configNET\_MASK3 0

/\* Gateway address configuration. \*/

**#define** configGW\_ADDR0 192

**#define** configGW\_ADDR1 168

**#define** configGW\_ADDR2 1

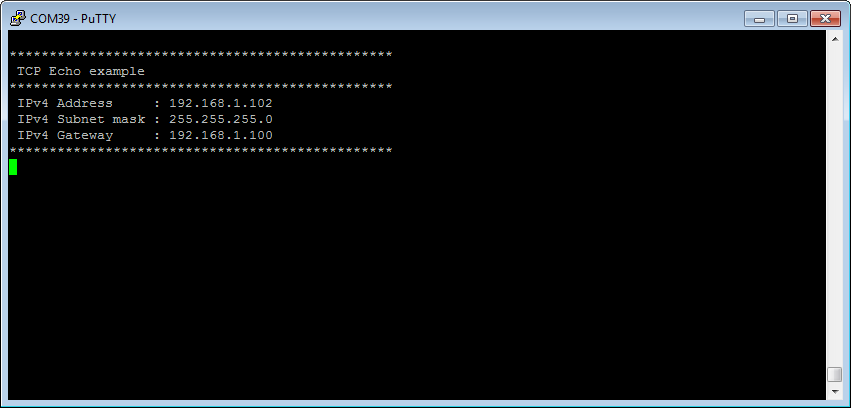
**#define** configGW\_ADDR3 100

So I changed it to 1.116

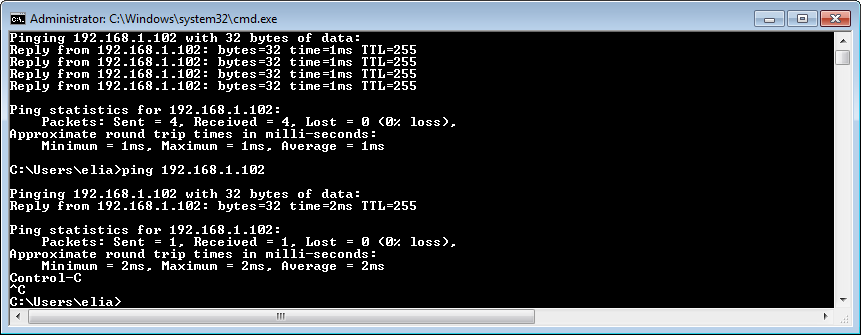
I also changed to output to row binary

lwip\_tcpecho\_freertos\_frdmk64f.bin and copy it to the MBED disk

we can see on the terminal the message:



Make sure you have ping to the board ( connect the Ethernet cable and make same subnet)



The Ethernet example is ready to be use , modify now,

In tcpecho.c the port is 7

Lets change it to 80 ( more resnoable)

**tcpecho\_thread**(**void** \*arg)

{

**struct** netconn \*conn, \*newconn;

err\_t err;

LWIP\_UNUSED\_ARG(arg);

/\* Create a new connection identifier. \*/

/\* Bind connection to well known port number 7. \*/

**#if** LWIP\_IPV6

conn = netconn\_new(NETCONN\_TCP\_IPV6);

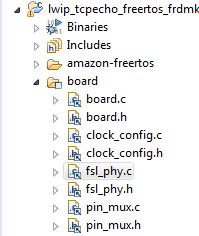
netconn\_bind(conn, IP6\_ADDR\_ANY, 7);

**#else** /\* LWIP\_IPV6 \*/

conn = netconn\_new(*NETCONN\_TCP*);

netconn\_bind(conn, IP\_ADDR\_ANY, 80);

reduce the timeout in fsl\_phy.c ( board directory)



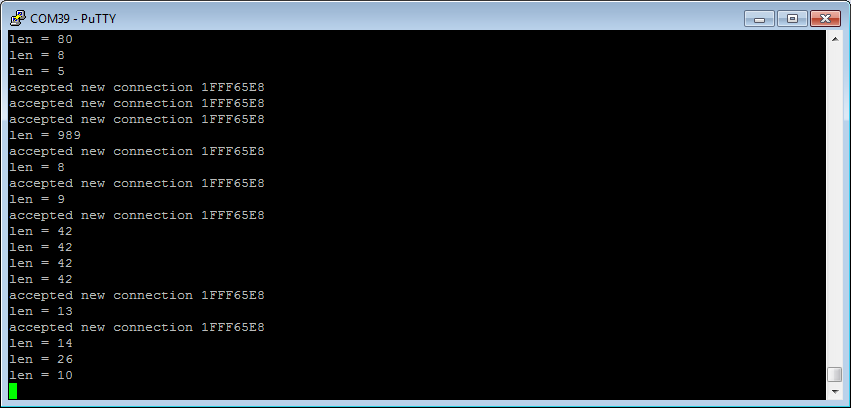
/\*! @brief Defines the timeout macro. \*/

**From #define** PHY\_TIMEOUT\_COUNT 0x3FFFFFFU

To:

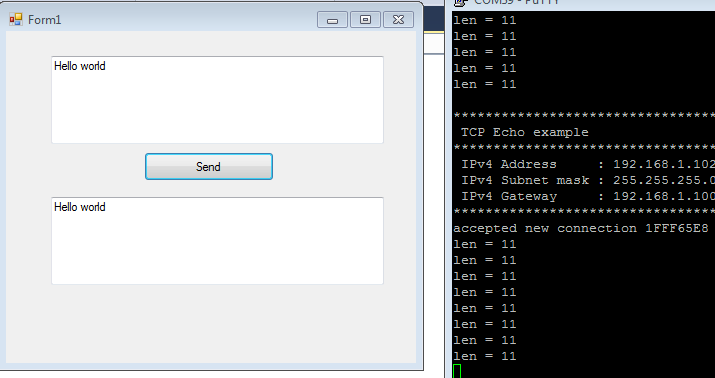
**#define** PHY\_TIMEOUT\_COUNT 0xFFFFU

I added some print in the code to see that we are connected and how many bytes was sent:



C# Application:

There are several ways to work as tcp client in c#



I will show several of them.