Published on *Analog Digital Lab: <br><br>electronics articles and tutorials* (http://10.240.11.2)

posted by maintenance on Wed, 02/06/2013 - 16:59

PIC18: Internet Controlled Robot

The project:

Build a robot that can be controlled over the Internet with video feedback.

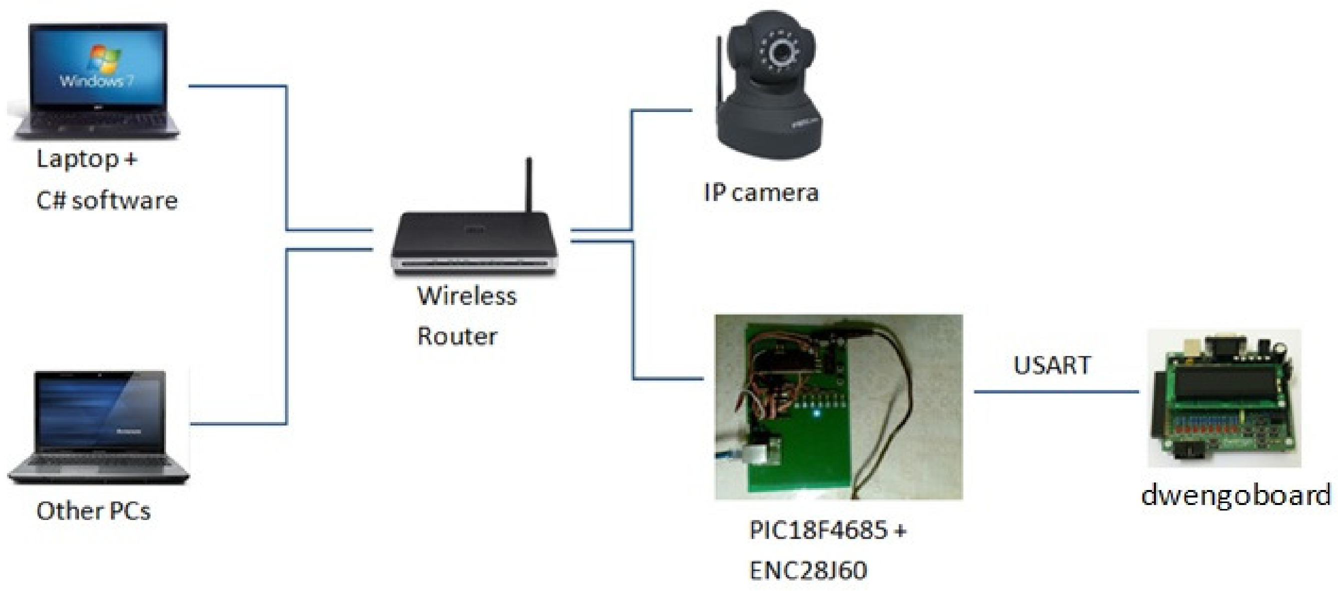
Components:

PIC18F4685
PIC18F4550+L293D H-bridge (or dwengoboard)
in28j60 (Ethernet iMod Module with ENC28J60)
Wireless router or bridge
3.3V voltage regulator: LD1117AV33 (or LM3940)

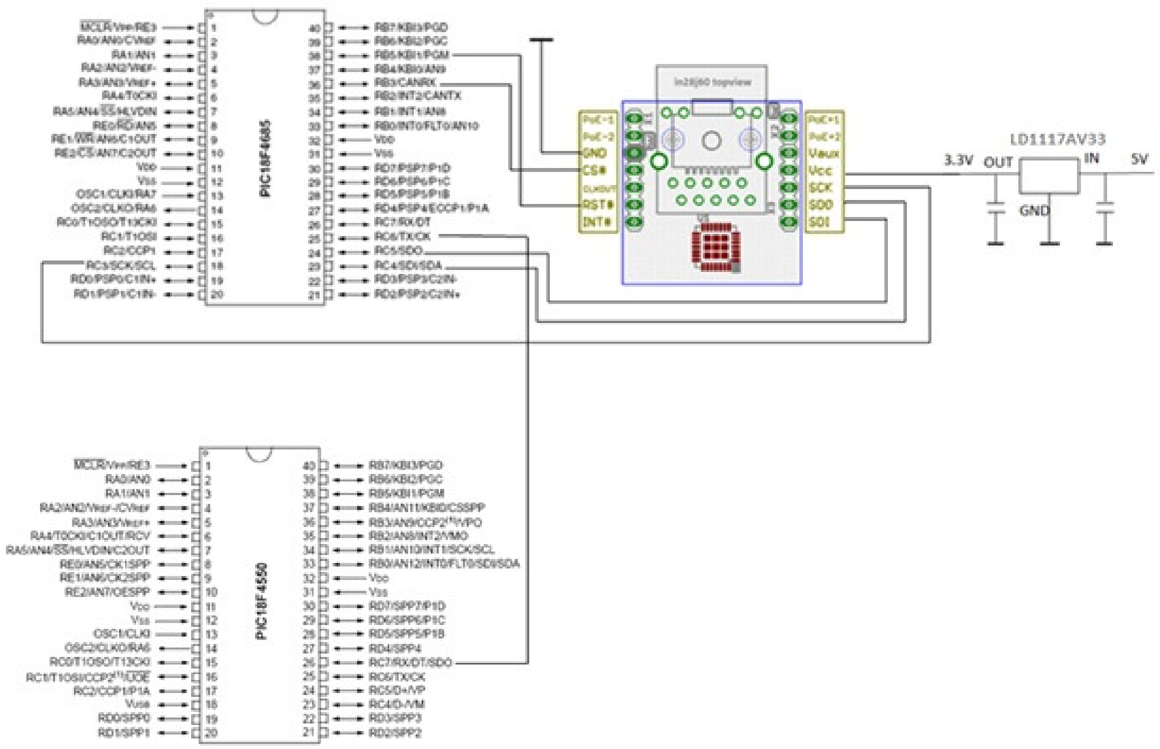
Documentation:

AN833 (The Microchip TCP IP Stack)
AN1120 (Ethernet Theory of Operation)
Datasheet of in28j60 ethernet module

Hardware Layout:



As showed above, your laptop serves as TCP Server which runs the C# software to connect the robot over the Internet. The other PCs or laptops can connect to the TCP Server with C# software(TCP Client) or LAN webpage. The IP camera can be connected to the router directly via Ethernet-cable as well as wireless WIFI, normally you will get a driver for camera when you buy it online. The PIC18F4685 connects to the router with the help of ENC28J60, and communicate with dwengoboard via USART, since the PIC18F4685 has only one pwm(which means only one motor can be controlled). And because of the small memory size of PIC18F4550, I made a decision to use the combination of both PICs.



Software:

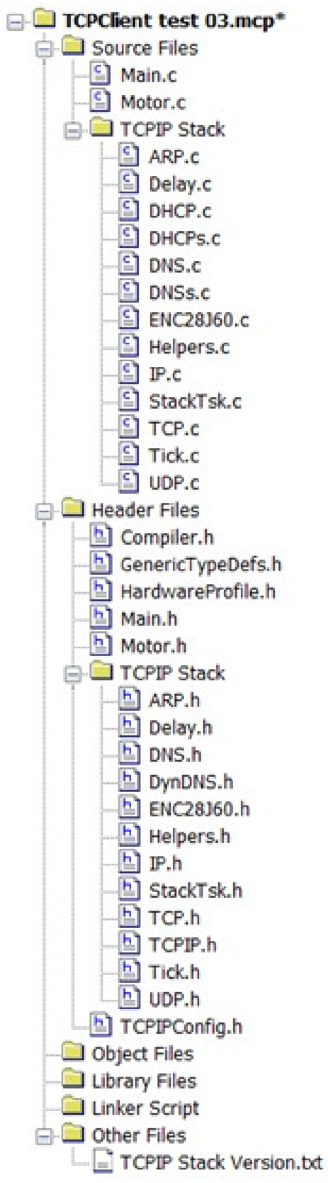
In this section there are two parts, TCP Client/Robot with MPLAB and TCP Server with C# .

Part I: Software with MPLAB IDE or MPLAB X

Here I used MPLAB IDE v8.86, newest version of MPLAB IDE. You can use the demo project of Microchip, aftering installing the Microchip Solutions (including libraries/demos etc) you can find the demo in this path: Microchip Solutions v2012-07-18 \TCPIP\Demo App. For PIC18 use this project:

C18-PIC18EX_ENC28.mcp

The TCPIP Stack is in Microchip Solutions v2012-07-18\Microchip.
The following figure shows which c- files or h-files you should include in your project. Based on the demo project given by the Microchip I modified and reduced the program to make it more readable and simple. Note that the motor.h and motor.c are not needed for this project since we are going to use the dwengoboard for the motors.



```
TCPClient test 03.mcp*
  Source Files
    Main.c
    Motor.c
    TCPIP Stack
      ARP.c
      Delay.c
      DHCP.c
      DHCPs.c
      DNS.c
      DNSs.c
      ENC28J60.c
      Helpers.c
      IP.c
      StackTsk.c
      TCP.c
      Tick.c
      UDP.c
  Header Files
    Compiler.h
    GenericTypeDefs.h
    HardwareProfile.h
    Main.h
    Motor.h
    TCPIP Stack
      ARP.h
      Delay.h
      DNS.h
      DynDNS.h
      ENC28J60.h
      Helpers.h
      IP.h
      StackTsk.h
      TCP.h
      TCPIP.h
      Tick.h
      UDP.h
    TCPIPConfig.h
  Object Files
  Library Files
  Linker Script
  Other Files
    TCPIP Stack Version.txt
```

In these included files above, I modified the main.h, main.c, Delay.h, Delay.c,Compiler.h, HardwareProfile.h, TCPIPConfig.h.
In HardwareProfile.h, I only adjusted the I/O-pins for buttons, motor,LEDS. Nothing special to be mentioned here.
In Compiler.h, I kept the configuration for compiler C18, and removed the rest.

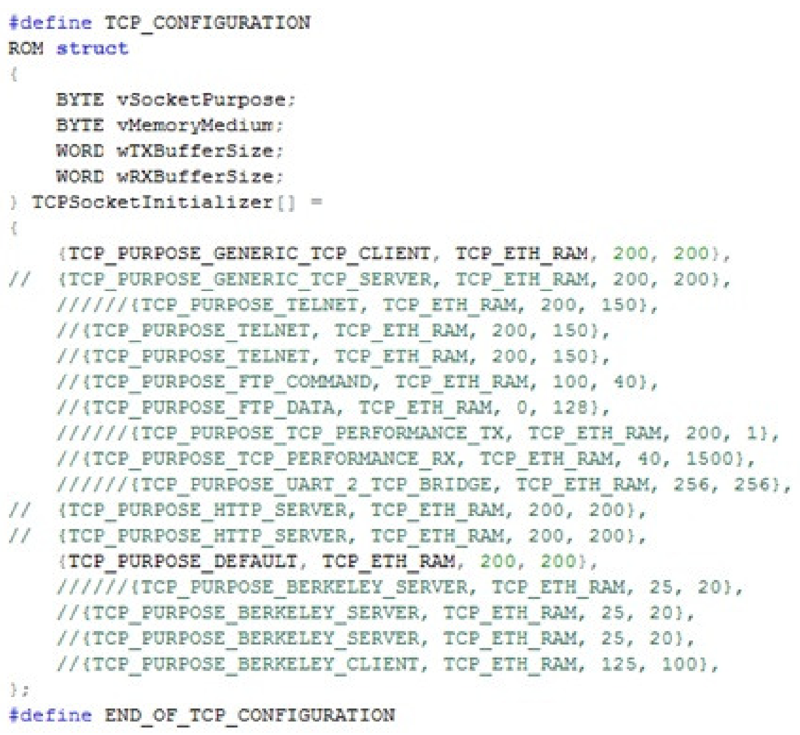
Details:

TCPIPConfig.h

Define only two Application Options, the rest is not needed for Generic TCP Client demo.

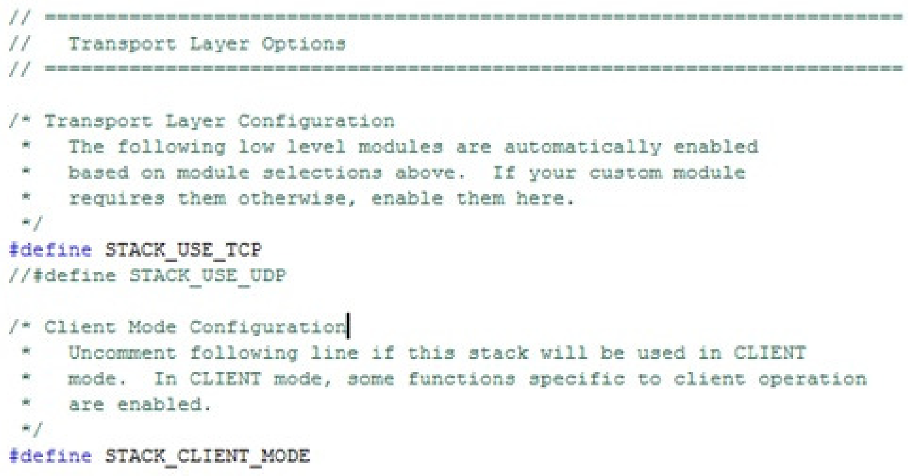
#define STACK_USE_DHCP_CLIENT
#define STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE

Make sure that you have assigned enough memory for TCP Client.



```
#define TCP_CONFIGURATION
ROM struct
{
    BYTE vSocketPurpose;
    BYTE vMemoryMedium;
    WORD wTXBufferSize;
    WORD wRXBufferSize;
} TCPSocketInitializer[] =
{
    {TCP_PURPOSE_GENERIC_TCP_CLIENT, TCP_ETH_RAM, 200, 200},
//  {TCP_PURPOSE_GENERIC_TCP_SERVER, TCP_ETH_RAM, 200, 200},
    //////{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
    //{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
    //{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
    //{TCP_PURPOSE_FTP_COMMAND, TCP_ETH_RAM, 100, 40},
    //{TCP_PURPOSE_FTP_DATA, TCP_ETH_RAM, 0, 128},
    //////{TCP_PURPOSE_TCP_PERFORMANCE_TX, TCP_ETH_RAM, 200, 1},
    //{TCP_PURPOSE_TCP_PERFORMANCE_RX, TCP_ETH_RAM, 40, 1500},
    //////{TCP_PURPOSE_UART_2_TCP_BRIDGE, TCP_ETH_RAM, 256, 256},
//  {TCP_PURPOSE_HTTP_SERVER, TCP_ETH_RAM, 200, 200},
//  {TCP_PURPOSE_HTTP_SERVER, TCP_ETH_RAM, 200, 200},
    {TCP_PURPOSE_DEFAULT, TCP_ETH_RAM, 200, 200},
    //////{TCP_PURPOSE_BERKELEY_SERVER, TCP_ETH_RAM, 25, 20},
    //{TCP_PURPOSE_BERKELEY_SERVER, TCP_ETH_RAM, 25, 20},
    //{TCP_PURPOSE_BERKELEY_SERVER, TCP_ETH_RAM, 25, 20},
    //{TCP_PURPOSE_BERKELEY_CLIENT, TCP_ETH_RAM, 125, 100},
};
#define END_OF_TCP_CONFIGURATION
```

Use TCP protocol considering that the TCP is more stable and realiable than UDP.



```
// =======================================================================
//   Transport Layer Options
// =======================================================================

/* Transport Layer Configuration
 *   The following low level modules are automatically enabled
 *   based on module selections above.  If your custom module
 *   requires them otherwise, enable them here.
 */
#define STACK_USE_TCP
//#define STACK_USE_UDP

/* Client Mode Configuration
 *   Uncomment following line if this stack will be used in CLIENT
 *   mode.  In CLIENT mode, some functions specific to client operation
 *   are enabled.
 */
#define STACK_CLIENT_MODE
```

Main.c

// Defines the server to be accessed for this application

static BYTE ServerName[] ="192.168.0.194";
static WORD ServerPort = 2000;

"192.168.0.194" is the IP address of your TCP Server which the microcontroller should be connected with. The port number can vary almost between 2000 and 5000, make sure that this port number is the same that you set in C# software.

GenericTCPClient.c

I removed the GenericTCPClient() to main.c and modified the function as well. Use these two functions to retrieve data from the socket, then you can command the robot to do whatever you want according to the received data.

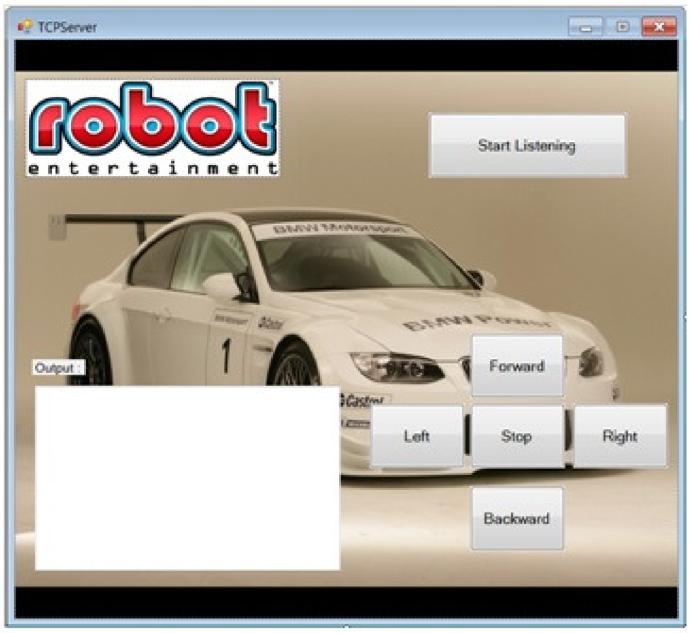
```
wMaxGet=TCPIsGetReady(MySocket);
w -= TCPGetArray(MySocket,(BYTE *)buffer, wMaxGet);
```

Communicate with PIC18F4550 using USART, e.g:

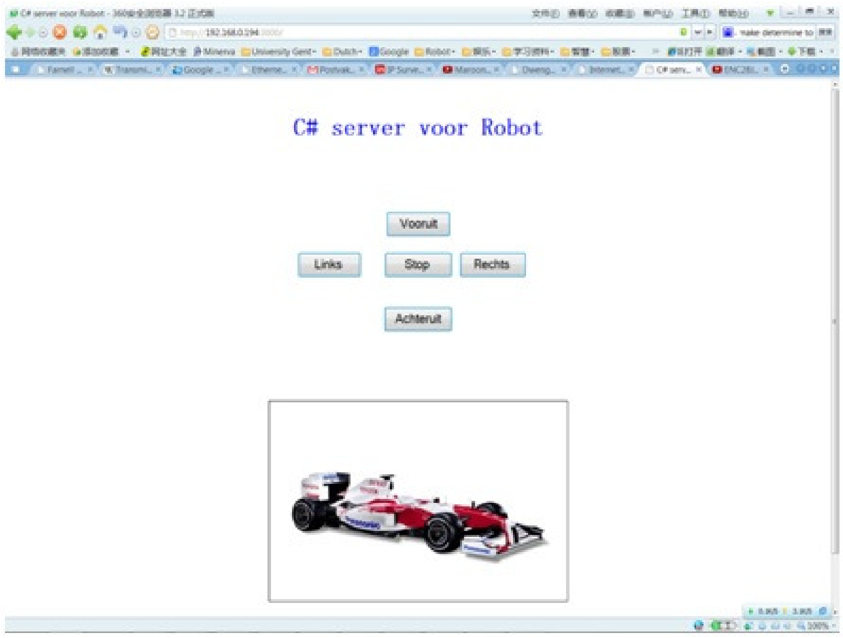
```
void initUsart() {
         OpenUSART(  USART_TX_INT_OFF &
                     USART_RX_INT_OFF &
                     USART_ASYNCH_MODE &
                     USART_EIGHT_BIT &
                     USART_SYNC_SLAVE &
                     USART_CONT_RX &
                     USART_BRGH_LOW,129);
                 }
```

Part II: Software with C#

The C# should impliment the following steps:
1) Create a TCP Listener and listen to a certain port number, e.g. 2000
2) Accept the incomming TCP Client/Robot and assign a socket for it
3) Keep connecting and transfer data
To verify if the connection is successfully established, you can make use of Wireshark sniffing the internet traffic.
To make it more playful, you can accept multiple TCP Client or write a LAN webpage, sothat the others can also play with the robot, for example using the ipad or iphone control the robot via LAN webpage.



Webpage in LAN:

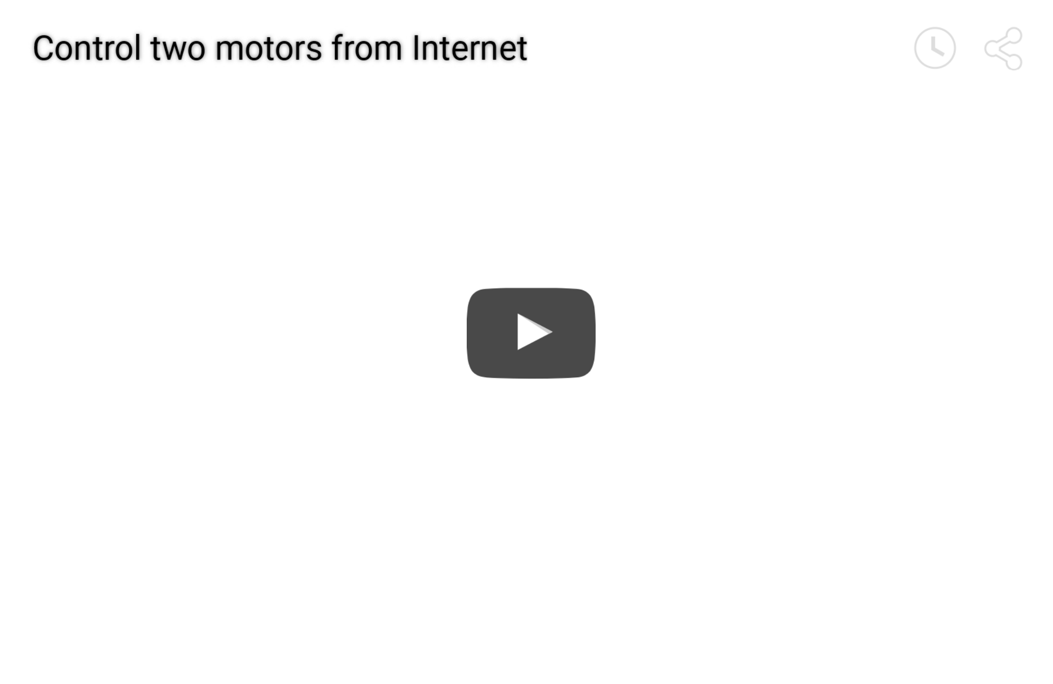


Demo videos:

Internet Controlled Robot Clip 1

Internet Controlled Robot Clip 2

ENC28J60 TCPClient Internet Robot



Files for Download
Build your own Internet-controlled robot

Generic TCP Client source code for PIC18 + ENC28J60

TCP Server using C#

dwengoboard
Get a dwengoboard

Tags:
internet
robot
remote
controlled
PIC18
PIC
dwengo
dwengoboard
enc28j60
C#

Comments

PIC firmware for Wifly RN-XV

Permalink On **Mon, 07/21/2014 - 03:07**

PIC firmware for Wifly RN-XV modem configuration without Wifly library and with front-end: https://dl.dropboxusercontent.com/u/101922388/WiflySanUSB.zip

By **SanUSB (not verified)**