ESP8266 WiFi Module with multi-session HTTP API (pure MMBasic)

ESP8266

This is a complete ESP8266 module that will connect to the WiFi of your choice and then accept connection inbound. No CFunctions or anything, purely handling a serial port and chopping up whatever comes through. It works really well and is fast. It handles multiple inbound connections - up to five at a time. It looks like a big deal but a lot of it is simply functions to make string handling easier (at the bottom).

This might be for a MicroMite based web server or to allow you to set the MicroMite doing things. Here I don't write back HTML (API doesn't require it) but you could just as easily, you could ask for status on bits, analogue inputs. You could send Instructions for quite complex activities, your imagination is the limit.

This is an extract (with changes) from my SMS Gateway. I connect in with a web page and issue commands via this collection to send text message from my website. This bit works perfect, all the time... (the M590 GSM module is more temperamental and I have some work to do before I can publish that bit).

This is enough to get you going, you have to pick the bones apart of the main loop and understand what is happening but it is mainly just detecting the inbound connection and then using string slicing to pick apart the HTTP request to see what is being asked for.

Here's a load of links I used when I was getting this working. Research loads!

https://alselectro.wordpress.com/2015/05/13/wifi_module_esp8266_2_tcp_client_server_mode/
https://www.sparkfun.com/products/13678
http://www.instructables.com/id/Using_the_ESP8266_module/
http://www.instructables.com/id/Using_the_ESP8266_module/step3/Configuring_the_8266_Module/
http://www.instructables.com/id/How_to_Build_a_ESP8266_Web_Server/step7/Results/
https://github.com/esp8266/esp8266_wiki/wiki
http://www.esp8266.com/
http://dominicm.com/esp8266_send_receive_data/
https://www.youtube.com/watch?v=aqjx_85_hwg
https://cdn.sparkfun.com/datasheets/Wireless/WiFi/Command%20Doc.pdf
http://41j.com/blog/2015/01/esp8266_client_mode_connect_remote_host_simple_example/

Enjoy

```
'
'ESP8266 Module. Andrew Henderson. October 2016
'

'Preamble
    OPTION BASE 0

    CONST LAN=2


'S/W Init
```

```
    DIM STRING A$,B$,D$
    DIM STRING LNBuf$
    DIM STRING LANRxLN$,LANTxLN$,TChr$ LENGTH 1
    DIM INTEGER C,Z,TOFg,Chan,Conn(4),PingTMR,PingCtr,RL1TMR LANf
    DIM FLOAT x

    'Starting WLAN Interface

    Open "COM2:9600,1024" As #LAN

    ESP8266Reset


Main:



        ' don't do this every time through the loop, set a timer for no more
than 60second intervals {
        'Test Internet connectivity - you'll have to handle what to do if
this is broken, there's nothing here for it.
    A$=ESP8266Exec$("AT+PING="+PA,4000,0)   ' PA=Ping address. try to be
nice here and set this to a DNS server from your ISP e.g. 100,200,111,222 or
www.bbc.co.uk or something that won't mind you pinging them
    IF A$="OK" THEN
        'good
        '...
    ELSE
        'internet down
                '...
    END IF
        }






    'Web API parsing

    LANRxLN$=GetIO$(LAN,1)
    IF LANRxLN$="" OR LANRxLN$="T/O" THEN GOTO Main ' nothing waiting on the
input

    IF MID$(LANRxLN$,2,8)=",CONNECT" THEN ' ESP8266 provides the connect as
an indication of HTTP input
        Chan=VAL(LANRxLN$)
        IF Chan>=0 AND Chan<=4 THEN
            Conn(Chan)=1
                        PRINT "Inbound connect channel "+STR$(Chan)
            'Inbound connect channel on  channel 0 - 5
```

```
            GOTO IPEXIT
        END IF
    END IF

    IF MID$(LANRxLN$,2,7)=",CLOSED" THEN ' ESP8266 provides the closed as an
indication of HTTP channel closing
                                            ' either coz it was explicit or
it timed out
        Chan=VAL(LANRxLN$)
        IF Chan>=0 AND Chan<=4 THEN
            IF Conn(Chan)=0 THEN A$=" Abnormally " ELSE A$=""
            PRINT "Closed channel "+STR$(Chan) + A$
            Conn(Chan)=0
            GOTO IPEXIT
        END IF
    END IF

    IF LEFT$(LANRxLN$,5)="+IPD," THEN
        Chan=VAL(MID$(LANRxLN$,6,1))
        IF Chan>4 THEN GOTO IPDerrEXIT
        IF Conn(Chan)<>1 THEN GOTO IPDerrEXIT 'channel not open

        z=INSTR(LANRxLN$,":GET /<my page, e.g. index.htm>") ' +IPD,0,nnn:GET
/index.htm HTTP/1.1

        do:a$=GetIO$(LAN,2):loop until a$="Connection: Keep-Alive" ' empty
the rest of the envelope, this string is the last part of an HTTP connection
header

        IF z=>8 AND Z<=11 THEN ' checking for the position of the arguments
in the string - your mileage will vary... simple sanity check
            b$=MID$(LANRxLN$,z+13)
            z=INSTR(b$," HTTP/")
            IF z<21 THEN GOTO IPDerrEXIT '... and again
            b$=LEFT$(b$,z-1) ' remove the HTTP bit
            b$=Replace$(b$,"+"," ")' + to spaces
            args=Split(b$,"?") '  split the request arguments out
            IF args <3 THEN GOTO IPDerrEXIT ' your arguments might vary - I
have two, the header + two arguments = 3 strings from the spilt
            FOR n=2 TO args 'args=1 is a dummy
                SELECT CASE LEFT$(ucase$(SP$(n)),4) ' check for the name of
the argument here and assign the value to a variable
                    CASE "VAR1="
                        var1$=LTrim$(RTrim$(MID$(sp$(n),5)))
                    CASE "VAR2="
                        var2$=LTrim$(RTrim$(MID$(sp$(n),5)))
                END SELECT
            NEXT
            IF var1$="" OR var2$="" THEN GOTO IPDerrEXIT ' don't allow empty
values - your mileage may vary
            ERASE SP$
```

```
            Msg$=URLDecode$(Msg$)

                ' other api calls go here, I only have one in this
application


        END IF

        GOTO IPDerrEXIT 'default action IF we don't find an API call

    END IF


    GOTO IPEXIT
IPDerrEXIT:
    B$="ERROR: Malformed request {"+LANRxLN$+"}"
IPOKEXIT:

    PRINT #LAN,"AT+CIPSENDEX="+STR$(Chan)+",255" ' send our response back to
the requestor
    PAUSE 100

    PRINT#LAN,B$+"\0"
    PAUSE 100

    PRINT#LAN,"AT+CIPCLOSE="+STR$(Chan) ' then close the channel nicely
    Conn(Chan)=0

IPEXIT:

    GOTO Main

    END


'------------------ ESP8266 WiFi Modem SUBsys -------------

    SUB ESP8266Reset
        LOCAL STRING A$
        LOCAL INTEGER N

        PULSE LANRST,10                                 ' whatever pin you
have connected to the reset of the ESP8266
        PAUSE 400

        LANExec "ATE0",2,0
        LANExec "AT+CWAUTOCONN=0",2,0                   ' turn off
autoconnect while we initialise
        LANExec "AT+CWMODE=1",2,0
        LANExec "AT+CWQAP",4,0                          ' disconnect any
```

```
existing WiFi connections
        LANExec "AT+RFVDD",2,0
        LANExec "AT+CWJAP="+AP+","+PW,10,1              ' AP= name of the
network you want to join e.g. your WIFI SSID - and PW= it's password
        LANExec "AT+CIPSTA="+IP+","+GW+","+MS,2,0       ' the IP and Gateway
address for static IP addresses, rem this line for a static IP
        LANExec "AT+CIFSR",2,0
        LANExec "AT+CWAUTOCONN=1",2,0                   ' turn autoconnect
back on so if the WiFi drops, the ESP8266 will self negotiate to get it back
        LANExec "AT+CIPMUX=1",2,0
        LANExec "AT+CIPSERVER=1,19090",2,0              ' turn on the server
with the port you want your API to listen on, normally HTTP is 80 or
sometimes 8080, here I listen on 19090... you can use anything from 1 -
65536 but as a guideline; avoid anything below 1000 if you are not using 80

        LANTxLN$="":LANRxLN$=""                         ' buffers for receive
and transmit

        FOR N=0 TO 4:Conn(N)=0:NEXT                     ' clear the channel
connection flags

    END SUB




    SUB LANExec(CM$,T,Suppress) 'temp - will go when we have better handling
of the startup
        LOCAL STRING A$
        A$=ESP8266Exec$(CM$,T,Suppress)
        IF Suppress<2 THEN CLog A$,0
    END SUB

    FUNCTION ESP8266Exec$(Com$,TimeOut,Obf)
        LOCAL STRING A$ LENGTH 1,LN$
        LOCAL INTEGER C

        PRINT #LAN,Com$

        DO
            LN$=GetIO$(LAN,TimeOut)
            IF LN$="T/O" THEN ESP8266Exec$="T/O":EXIT DO
            IF LN$<>"" THEN
                IF Obf<2 THEN CLog "["+LN$+"]",0
                IF LN$="OK" OR LN$=">" OR LN$="SEND OK" THEN
ESP8266Exec$=LN$:EXIT DO
                IF INSTR(LN$,"ERROR") THEN ESP8266Exec$="ERROR
["+LNBuf$+"]":EXIT DO
            END IF
        LOOP
```

```
    END FUNCTION


'-----------------------------------------------------


    FUNCTION GetIO$(io,TimeOut)
        LOCAL STRING a$,b$
        LOCAL INTEGER c

        StartTOTimer TimeOut
        DO
            IF TOFg=1 THEN GetIO$="T/O":EXIT DO
            IF LOC(#io)<>0 THEN
                A$=INPUT$(1,#io):C=ASC(A$)
                SELECT CASE C
                    CASE 13
                        GetIO$=b$:EXIT DO
                    CASE 0 TO 31
                    CASE ELSE
                        IF LEN(b$)+LEN(a$)<255 THEN b$=b$+a$ ELSE b$=a$
                END SELECT
            END IF
        LOOP
        ClearTOTimer
    END FUNCTION


    SUB FlushIO(stream)
        LOCAL STRING A$
        DO WHILE LOC(#stream)<>0
            PAUSE 50:A$=INPUT$(LOC(#stream),#stream):PAUSE 50
        LOOP
    END SUB



    FUNCTION Replace$(a$,b$,c$)
        LOCAL INTEGER z
        LOCAL STRING x$,y$
        z=1
        DO
            z=INSTR(z,a$,b$)
            IF z=0 THEN EXIT DO
x$=LEFT$(a$,z-1):y$=MID$(a$,z+LEN(b$)):a$=x$+c$+y$:z=LEN(x$)+LEN(c$)
        LOOP
        Replace$=a$
    END FUNCTION
```

```
    FUNCTION URLDecode$(a$)
        LOCAL INTEGER z
        LOCAL STRING b$,c$
        z=1
        DO
            z=INSTR(z,a$,"%")
            IF z=0 OR z=LEN(a$) THEN EXIT DO
b$=MID$(a$,z,3):c$=CHR$(VAL("&h"+MID$(b$,2))):a$=Replace$(a$,b$,c$)
        LOOP
        URLDecode$=a$
    END FUNCTION


    FUNCTION LTrim$(a$)
        LOCAL INTEGER m
        FOR m=1 TO LEN(a$)
            IF not(MID$(a$,m,1)=" " OR MID$(a$,m,1)=CHR$(9)) THEN
LTrim$=MID$(a$,m):EXIT FUNCTION
        NEXT
        LTrim$=""
    END FUNCTION


    FUNCTION RTrim$(a$)
        LOCAL INTEGER n
        FOR n=LEN(a$) TO 1 STEP -1
            IF not(MID$(a$,n,1)=" " OR MID$(a$,n,1)=CHR$(9)) THEN
RTrim$=LEFT$(a$,n):EXIT FUNCTION
        NEXT
        RTrim$=""
    END FUNCTION


    FUNCTION Split(a$,b$)
        LOCAL INTEGER z,n,m
        ON ERROR SKIP
        ERASE SP$
        z=1:n=0
        DO
            z=INSTR(z,a$,b$)
            IF z=0 THEN
                IF n=0 THEN
                    DIM SP$(1):SP$(1)=a$:Split=1:EXIT FUNCTION
                ELSE
                    EXIT DO
                END IF
            ELSE
                n=n+1:z=z+LEN(b$)
            END IF
        LOOP
```

```
        m=n+1:n=1
        DIM SP$(m)
        DO
            z=INSTR(1,a$,b$)
            IF z=0 THEN
                SP$(m)=a$:EXIT DO
            ELSE
                SP$(n)=LEFT$(a$,z-1):a$=MID$(a$,z+LEN(b$)):n=n+1
            END IF
        LOOP
        Split=m
    END FUNCTION
```