TP SERVEUR WEB ESP8266

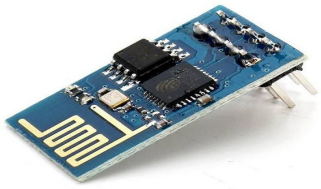
Objectifs : Etre capable de mettre en œuvre un mini serveur web contrôler par un PIC.
Être capable de créer un IOT (internet of think) « Objet connecté »

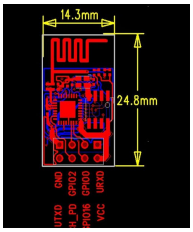
---

1. CONFIGURATION DU MODULE ESP8266

1.1. Matériel :

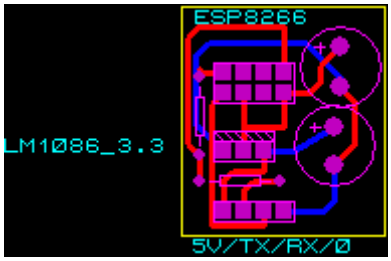
le module ESP8266 ESP01 :





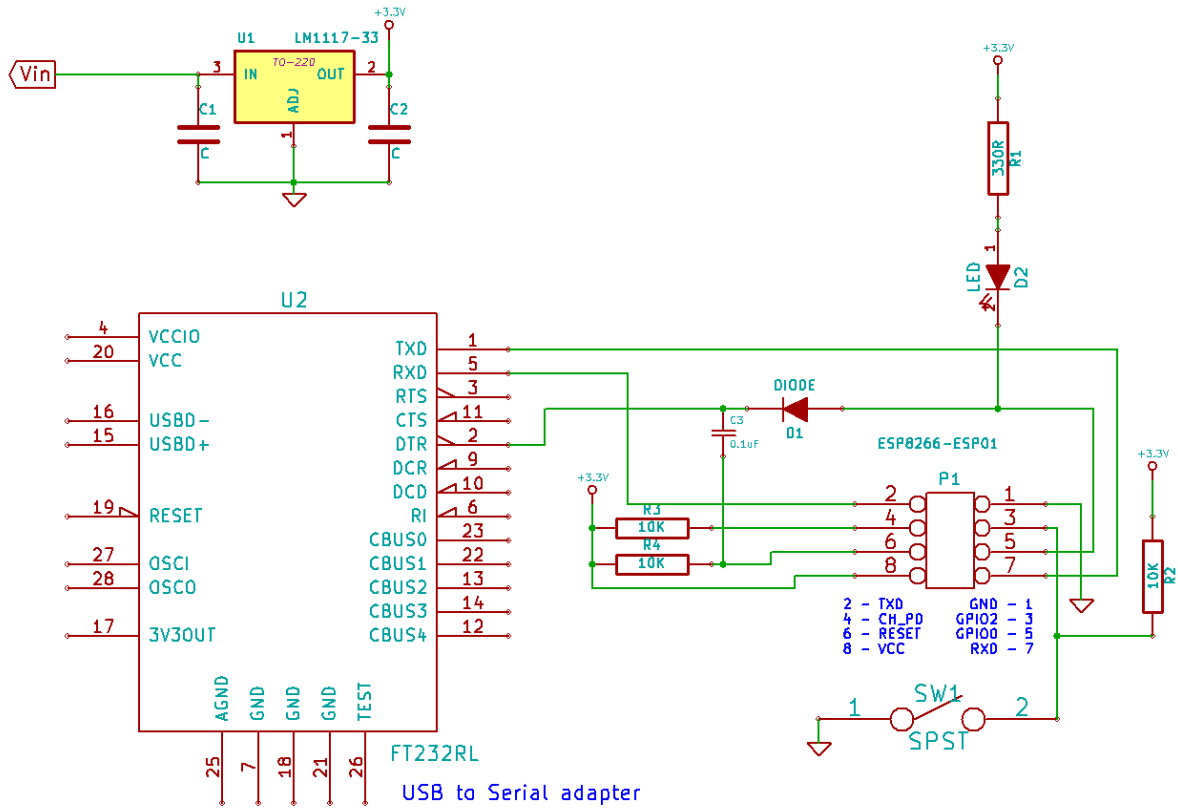
ATTENTION :LE MODULE SUPPORTE UNIQUEMENT 3.3V ET NON 5V.
un cable USB/TTL
une platine d'interfaçage : PICV2/module ESP8266 (fabrication maison ou platine d'essai utilisant un RIT5Vvers3,3V)
un PC avec le soft AppSTACK ESP8266 (disponible sur le net)
Ressources : ***http://www.esp8266.com***

1.2. Carte interface USB/ESP8266



Mettre un pont diviseur entre TXPIC et RXESP afin d'adapter le 5V en 3.3V (4,7k et 10k).
Le croisement de RXTX se fait sur la carte interface car la carte PICV2 ne croise pas avant le connecteur RXTX.

Le schéma complet sans module USB/TTL est :



1.3. Configuration logiciel.

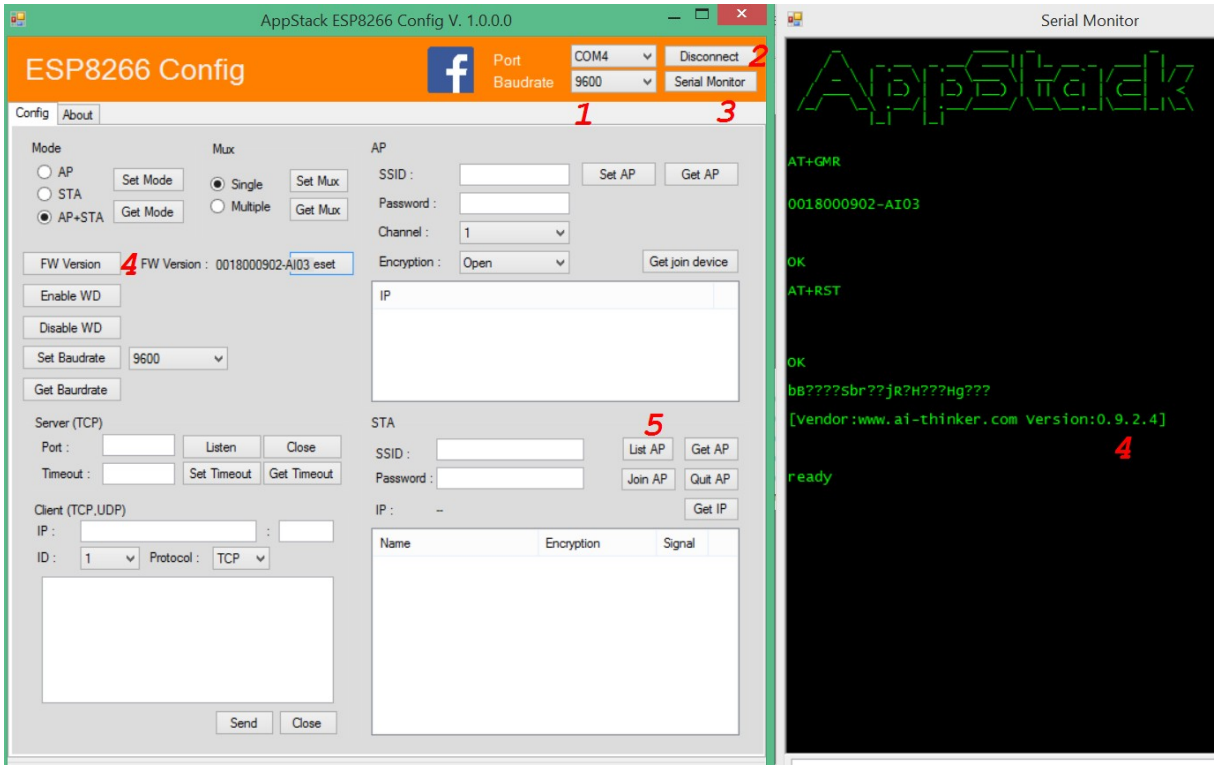
L'objectif est de fournir une adresse IP à notre module en mode STA (172.16.2.126 pour le lycée)

Détecter le COM du cable USB/TTL par Périphérique+PortCOM+ choisir le com USBtoSerial

COM4 pour moi.

Lancer le logiciel AppStackESP8266.

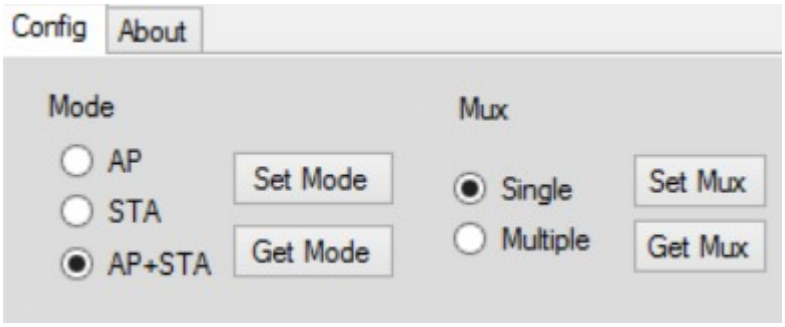
Après avoir configuré le port série (1) faire Connect (2)et ouvrir le Serial monitor (3).
La vitesse pour mon module est 9600Bauds (faire des tests car la vitesse des modules est différentes suivant la version du firmware)
Un clic sur FW Version (4) permet de connaitre la version du Firmware installé sur le module (version 0.9.2.4)



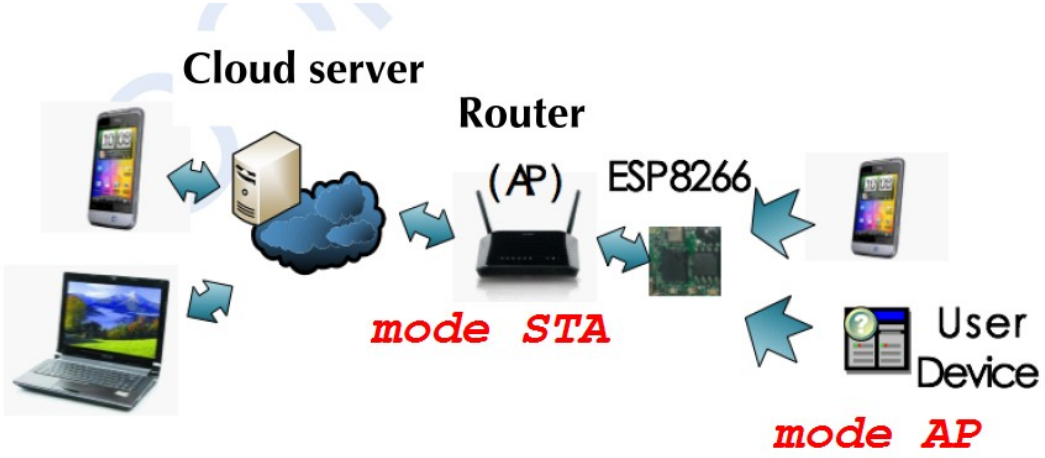
Mode de fonctionnement du module ESP8266

Le module peut fonctionner en 2 modes :
AP (access point) : le module est alors un point d'accès Wifi et peut communiquer avec des Hotes.
STA (station) : le module est un hote et il doit se connecter à un point d'accès.
Le module peut fonctionner

- soit en AP,
- soit en STA
- soit dans les 2 modes en même temps AP+STA (il possède alors 2 adresses IP : une comme AP et une comme hôte)

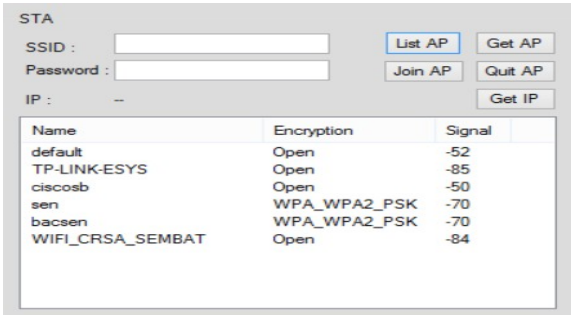


Le schéma suivant illustre le mode AT+STA :



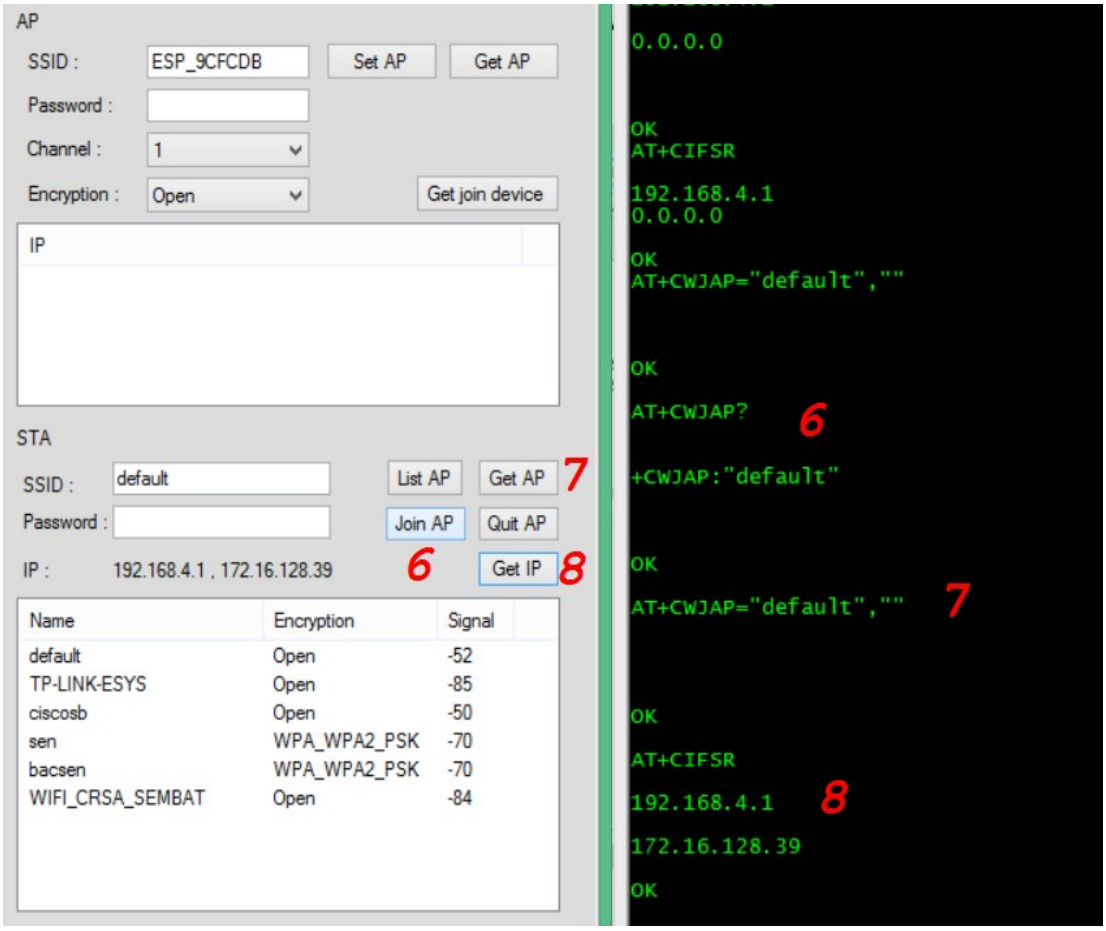
Mode STA : le module doit se connecter à un AP.

Cliquer sur « LIST AP »(5) afin de faire apparaître les Access Points présents dans votre environnement.



Je souhaite m'intégrer à « default » (JoinAP (6)) avec l'IP 172.16.2.126 (adresse autorisée par mon gestionnaire réseau)
Ceci n'est pas possible avec ce soft donc je choisis de me faire attribuer un adresse IP en DHCP (GetIP (8)).
Je peux vérifier ma connexion par (GetAP (7)).

Mon adresse IP sur le réseau est donc : 172.16.128.39.



Une fois cette configuration effectuée le module se souviendra des informations.

Mode AP (le module est un point d'accès)

Des informations sur le mode AP (access point) sont aussi disponibles :
l'adresse IP : 192.168.4.1
le SSID du module : ESP_9CFFCDB (ESP_xxxx xxx = fin de l'adresse MAC)
si vous utilisez un PC ou un téléphone en wifi vous verrez cet AP dans le réseau environnant.

1.4. Travail à faire .

Connecter votre module au réseau du lycée en suivant la procédure précédente.

2. COMMUNICATION AVEC LA CARTE PICV2.

Le module PICV2/ESP8266 permet de rendre la carte PICV2 compatible avec le Wifi (cf paragraphe 1.2)
IMPORTANT : Retirer le CI max233 afin d'éviter un conflit sur le port COM.
Brancher le module sur la carte PICV2.

Charger le programme wifiesp8266.c
Lire les commentaires
Téléverser le programme dans le PIC en utilisant PICKIT2
Tester le bon fonctionnement de la carte connectée :

A l'aide d'un navigateur internet aller à l'adresse du module en mode STA (172.16.128.39)

Modifier la valeur du CAN en agissant sur le potentiomètre AIN0 et vérifier que la valeur s'affiche sur la page Web du module.
Lors de l'affichage : "Attente POST.. "
Cliquer sur l'allumage de la LED ROUGE et vérifier l'action sur la carte PICV2.
Cliquer sur l'extinction de la LED ROUGE et vérifier l'action sur la carte PICV2.
Il peut être nécessaire de recommencer plusieurs fois la manoeuvre.

3. ETUDE DE LA COMMUNICATION

La communication utilise un protocole HTTP. (cours christian caleca...)
Le PIC réalise par programme un serveurWeb simplifié et répond aux requêtes du module (Les requêtes sont effectuées par l'utilisateur par l'intermédiaire de la page HTML.)

3.1. Configuration du module par le PIC.

Pour configurer le module il suffit d'envoyer les commandes AT appropriées. (cf doc. Technique du module ESP8266)

3.2. Affichage d'une page Web d'accueil

Demande d'envoi d'une page Web du module+PIC (serveur) vers le client (navigateur internet : mozilla , IE, chrome, opera...)

La requête d'une page Web se caractérise par un requête GET :

```
 +IPD,0,307:GET / HTTP/1.1\0D
Host: 192.168.0.13\0D
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:36.0) Gecko/20100101 Firefox/3
\0A\0D
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\0D
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3\0D
Accept-Encoding: gzip, deflate\0D
Connection: keep-alive\0D
\0A\0D
\0A\0D
OK\0D
```

Solution simple : détecter le 'G' du GET puis attendre le '>' qui indique que l'on peut envoyer la page HTML.

3.3. Lecture des données du CAN

Le renvoi de données du système vers le web se fait en créant une page HTML contenant les données mesurées par le système.

Exemple :

```
void CreerPageHtml(){

    printf("<HTML><HEAD></HEAD>");//19 char
    CreerStyleHtml();//256char
    printf("<BODY bgcolor = '#FFFF7F'>");//26char
    printf("<H1>SERVEUR WEB</H1>");//20 char
    printf("<H2>CAN1 = %03u </H2>",valCAN0);//20char
    CreerFormHtml();//194char
    printf("</BODY></HTML>");//14char
    printf("\r\n\r\n");

}//fin CreerPageHtml
```

3.4. Contrôle de la LED

Contrôle à travers une page Web des éléments de la carte PICV2.
Afin de réaliser ce contrôle il faut créer un formulaire dans la page HTML et récupérer les données de la page HTML grâce à un requête POST qui permet de renvoyer des données du HTML (client) vers le serveur Web.

```
void CreerFormHtml(){
//creation du formulaire HTML
//si LEDON alors
//alors cocher la case de la LEDON et mettre la valeur à 1
//sinon ne pas cocher la case de la LEDOFF et mettre la valeur à 0
   printf("<form method=\"post\" name=\"form1\"><br>");//37char le \" permet d'écrire " en html sans fermer le printf !
```

```
  if (LEDROUGEON)
  {
    printf("<input name=\"RDIOLR\" value=\"1\" ");//31char
    printf("type=\"radio\" checked>LED ROUGE ON<br>");//37char
    printf("<input name=\"RDIOLR\" value=\"0\" ");//31char
    printf("type=\"radio\">LED ROUGE OFF<br>");//30char = 129total
  }//fin if LEDROUGEON
  else
  {
    printf("<input name=\"RDIOLR\" value=\"1\" ");//31char
    printf("type=\"radio\">LED ROUGE ON<br>");//29char
    printf("<input name=\"RDIOLR\" value=\"0\" ");//31char
    printf("type=\"radio\" checked>LED ROUGE OFF<br>");//38char = 129total
  }//fin else
  //ajout bouton submit
  printf("<input type=\"submit\">");//21char
  printf("</form>");//7char


}//fin CreerFormHtml //37 + 129+21+7 = 194
```

l'entrée de formulaire <input name="RDIOLR" value="1 "crée une donnée qui sera utilisé dans le programme C :

```
   while(getc()!='R'){};//detection de la donnée RDIOLR
  while(getc()!='D'){};
  while(getc()!='I'){};
  while(getc()!='O'){};
  while(getc()!='L'){};
  while(getc()!='R'){};
  while(getc()!='='){};
  if(getc()=='1') //ok testé
  {
    output_high(LEDROUGE);
    LEDROUGEON=true;
  }//if
  else //sinon RDIOLR=0 alors eteindre ledrouge et memoriser l'info
  {
    output_low(LEDROUGE);
    LEDROUGEON=false;

  }//else
```

4. SOURCE COMPLET DU PROGRAMME EN C :

```
/*
Affichage val can ok sur page web
controle LED rouge ON et OFF ok
Pb de bouclage...

*/

/*
```

La procedure a suivre pour créer un serveur web avec envoi d'une donnée :

Configuration de l'esp8266 :
il faut que l'esp soit en mode AT+STA et connaitre l'adresse IP (IPscan32) ou utilisation du soft ESPconfig.exe pour le placer dans le bon mode.

envoyer les commandes AT pour configurer le module ESP

A. Construire un Web server
1. Accepter les connections multiples :
  envoyer "AT+CIPMUX=1\r\n"
2. Configurer en mode server à l'adresse 80 (80 car http sur port80)
  envoyer "AT+CIPSERVER=1,80\r\n"
3. Avec un navigateur web aller à l'adresse IP du module Wifi port80
  192.168.0.13:80 (chez moi)

4. Réception dune GET du module avec +IPD,0 ... avec 0 comme canal

5. Envoyer une réponse avec le canal et la taille des données en octet
  envoyer "AT+CIPSEND=channel,nboctet

Attention : finir l'envoi du html avec printf("\r\n\r\n"); le nb d'octet ne tient pas compte de ce final.

6. Attente d'une requete POST et détection d'une variable de la page WEB (RDI0LDR qui controle les LED)

6. Il faut fermer la connection
*/

```
#include <16F876.h>

#device adc=8



#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES NODEBUG                  //No Debug mode for ICD

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,ERRORS)
#use rs232(baud=9600,parity=N,xmit=PIN_C5,rcv=PIN_C0,bits=8,stream=DEBUG,ERRORS)
#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)

#include <PCF2119_Driver_LCDI2C.c>
#include <stdlib.h>

#define allume_LEDVERTE output_high(PIN_C2);

#define  KEYHIT_DELAY   500     // in milliseconds
#define LEDROUGE PIN_C0
#define LEDJAUNE PIN_C1
#define LEDVERTE PIN_C2
```

```
const int8 TailleBufferIn=30;

char DataBuffer[TailleBufferIn] ; //buffer d'entree
char HTML[30];
int8 valcan0;//valeur acquise sur canal0
int16 nboctet = 549;//attention nbctet doit etre exacte et inferieur à 1024
boolean LEDROUGEON=true;//indicateur de LEDROUGE à ON ou off

#int_RDA
void  RDA_isr(void)
{
int8 li;
int8 lchar;

//   disable_interrupts(GLOBAL); // all interrupts OFF
//   disable_interrupts(INT_RDA); // RS232 OF


//   enable_interrupts(GLOBAL); // all interrupts OFF
//   enable_interrupts(INT_RDA); // RS232 OFF

}

/*****************************************************************************/
boolean testOK(){

int8 i;
//attente OK
  while(getc()!='O') {};
  while(getc()!='K'){return(true);};

//fin attente OK

}//fin testOK
/*****************************************************************************/
/*************************************/
// initialisation du CAN
void initCAN(){
// initialise le CAN//
   setup_port_a( RA0_RA1_RA3_ANALOG );
   setup_adc( ADC_CLOCK_DIV_32 );
}

/*************************************/
// fonction d'acquisition : la valeur retournée est la valeur convertie du canal0
int8 acquerirCAN(int lcanal){

   int lvaquire;

   set_adc_channel( lcanal );
   delay_us(100);
   lvaquire=read_adc();
   return lvaquire;
}

void CreerFormHtml(){
//creation du formulaire HTML
//si LEDON alors
```

```
//alors cocher la case de la LEDON et mettre la valeur à 1
//sinon ne pas cocher la case de la LEDOFF et mettre la valeur à 0
   printf("<form method=\"post\" name=\"form1\"><br>");//37char le \" permet d'écrire " en html sans fermer le
printf !

  if (LEDROUGEON)
  {
    printf("<input name=\"RDIOLR\" value=\"1\" ");//31char
    printf("type=\"radio\" checked>LED ROUGE ON<br>");//37char
    printf("<input name=\"RDIOLR\" value=\"0\" ");//31char
    printf("type=\"radio\">LED ROUGE OFF<br>");//30char = 129total
  }//fin if LEDROUGEON
  else
  {
    printf("<input name=\"RDIOLR\" value=\"1\" ");//31char
    printf("type=\"radio\">LED ROUGE ON<br>");//29char
    printf("<input name=\"RDIOLR\" value=\"0\" ");//31char
    printf("type=\"radio\" checked>LED ROUGE OFF<br>");//38char = 129total
  }//fin else
  //ajout bouton submit
  printf("<input type=\"submit\">");//21char
  printf("</form>");//7char


}//fin CreerFormHtml //37 + 129+21+7 = 194


void CreerStyleHtml(){
/*

creation du style de la page
*/
    printf("<STYLE>");//7char
    printf("H1 {color:#804040;background:#FF8040;font-weight:bold;");//54char
    printf("font-family:Comic Sans MS;font-size:28;font-style:normal;");//57char
    printf("text-align:center;}");//19char
    printf("H2 {color:#000000;background:#FFFFFF;");//37char
    printf("font-family:Comic Sans MS;font-size:16;");//39char
    printf("font-style:normal;text-align:left;}");//35char
    printf("</STYLE>");//8char
//256 char
}//fin CreerStyleHtml
/*****************************************************************************/
void CreerPageHtml(){

    printf("<HTML><HEAD></HEAD>");//19 char
    CreerStyleHtml();//256char
    printf("<BODY bgcolor = '#FFFF7F'>");//26char
    printf("<H1>SERVEUR WEB</H1>");//20 char
    printf("<H2>CAN1 = %03u </H2>",valCAN0);//20char
    CreerFormHtml();//194char
    printf("</BODY></HTML>");//14char
    printf("\r\n\r\n");

}//fin CreerPageHtml
/*****************************************************************************/

/*****************************************************************************/
void main()
```

```
{
int8 i=0;
int8 channel=0;
char strbuffer[20];
int8 lchar;
int8 valCAN;

  setup_adc_ports(AN0_AN1_AN3);
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);

  // TODO: USER CODE!!

  set_tris_a(0b00111111);
  set_tris_b(0b11111111);
  set_tris_c(0b10010000);

  initCAN();//initialise le CAN


  Init_Ecran();//initialise l'écran LCD
  Efface_Ecran();
  allume_LEDVERTE;

   printf(Affiche_caractere,"Debut Wifi\n\r");
   delay_ms(2000);
     //envoi command AT vers Wifi
  fputs("AT+RST"); //testOK() ; //reset le module ESP8266
  fprintf(DEBUG,"DEBUG ATRST\r\n");   delay_ms(1000);
//lecture de l'adresse IP du module ESP8266
  fputs("AT+CIFSR");
  for(i=0;i<20;i++)
  {
    DataBuffer[i]=getc(); //detecte l'arrivee d'un K de OK pour finir l'enregistrement des données
    if (DataBuffer[i] == 'K') break;
  }//for
  Efface_Ecran();printf(Affiche_caractere,"%s ",DataBuffer);




do{


//   fputs("ATE0");//annule l'echo des commandes AT
//   fprintf(DEBUG,"ATE0 \r\n");
//   delay_ms(1000);

  fputs("AT");//test la communication avec le module ESP8266
  testOK();//attent la confirmation OK du module

  delay_ms(1000);
  fputs("AT+CIPMUX=1");  testOK(); delay_ms(1000);  //configure en multi connexion
  Efface_Ecran();printf(Affiche_caractere,"CIPMUX=OK"); delay_ms(1000);

  valCAN0 = acquerirCAN(0);//acquiere la valeur du CAN
```

```
   fputs("AT+CIPSERVER=1,80");  testOK(); delay_ms(1000);//active (1) le module en mode server sur port 80
   Efface_Ecran();printf(Affiche_caractere,"Attente reqHTTP");
   Lcd_Place_Curseur(2, 1);  printf(Affiche_caractere,"CAN0 = %u",valCAN0);
   fprintf(DEBUG,"REQ HTTP? \r\n"); delay_ms(1000);

    while(getc()!='G') {};//attente du G de GET provenant d'une requete HTTP (il faut appeler le serveur par un navigateur)

   Efface_Ecran();printf(Affiche_caractere,"Cfg canaldata"); delay_ms(1000);
   channel = 0; //canal de transmission
   printf("AT+CIPSEND=%u,%lu\r\n",channel,nboctet);//indique le canal et la taille des données envoyées

//Efface_Ecran();printf(Affiche_caractere,"Attente >");//risque de gener la synchro
   while (getc()!='>'){};//attente >
   CreerPageHtml();//envoi page web

   Efface_Ecran();printf(Affiche_caractere,"Envoi PageWeb"); delay_ms(10000);
   puts("AT+CIPCLOSE=0"); // delay_ms(1000); //fermeture de la connection


     //printf(Affiche_caractere,"%s",buffer);
//attente d'une réponse de la page web par un POST
     //detecter la variable retour attendu "RDIOLR=1" (detecter RDI puis enregistrer dans un tableau afin de récuperer le =0 ou =1
   Efface_Ecran();printf(Affiche_caractere,"Attente POST..");
   while(getc()!='R'){};
   while(getc()!='D'){};
   while(getc()!='I'){};
   while(getc()!='O'){};
   while(getc()!='L'){};
   while(getc()!='R'){};
   while(getc()!='='){};
   if(getc()=='1') //ok testé
   {
     output_high(LEDROUGE);
     LEDROUGEON=true;
   }//if
   else //sinon RDIOLR=0 alors eteindre ledrouge et memoriser l'info
   {
     output_low(LEDROUGE);
     LEDROUGEON=false;

   }//else


}while(1);
}//main
```

-