IOT : CONNECTÉ UN IOT AVEC UN HOTSPOT WIFI CRÉÉ PAR CONNECTIFY

Objectifs : Etre capable d'installer un objet connecté sur un point d'accès (AP) wifi crée avec Connectify.

1. Introduction	1
2. Configuration du PC	2
2.1. Installer les logiciels sur le PC :	2
3. Configuration du module ESP8266	3
4. Adaptation du programme PIC aux paramètres du module ESP8266	4
5. Conclusion	5
6. Annexe	5
6.1. Code source du fichier PIC :	5

1. INTRODUCTION

Cette activité permet de créer un hotspot wifi à partir de votre PC équipé d'une carte Wifi. Il permet de créer sans autre matériel un sous réseau privé Wifi pour vos objets connectés. Cette solution est idéale dans une salle de classe.

Le serveur Web,Php, MySql sera installé sur ce même PC ce qui réduit encore le cout et confine votre mise en oeuvre évitant les problèmes avec le réseau extérieur.

Logiciels nécessaires : Connectify (tranforme le PC en AP Wifi) WAMP : tranfsorme le PC en serveur Web, Php, MySql PICKIT2 : téléversement vers maquette à PIC connecté Programme en C pour l'objet connecté (IoT) ESP8266Config.exe : configuration et test de l'ESP8266

Matériel : Module ESP8266 sur l'IoT + Adaptateur vers cartedev PicV2 Cable USB/TTL 5V.

2. CONFIGURATION DU PC

2.1. Installer les logiciels sur le PC :

Connectify :

Dans l'onglet « setting » : Mettre un nom au réseau : Connectify-sb Mettre un mot de passe : 0123456789 Ne pas utiliser pour connecter à internt : No Internet Sharing Configurer en WiFi Access Point, WPA2 Démarrer le HotSpot : start Hotspot

L'adresse du HotSpot (AP wifi) donnée par Connectify à votre PC apparaît en passant la souris sur le logo en haut à gauche :



Wamp :

définir l'adresse local dans httpd.conf à l'adresse fournit par Connectify "192.168.73.1" en utilisant un éditeur de texte (Notepad...)

Fichier C:\wamp\bin\apache\apache2.4.9\conf\httpd.conf #dans httpd.cong

ServerName 192.168.73.1:80

Sauvegarder le fichier et relancer Wamp.

Vous pouvez tester si le serveur Web est accessible en Wifi par un téléphone connecté sur le réseau Connectify-sb en allant à la page 192.168.73.1:80 à l'aide d'un navigateur.

Settings Clients Services	د ء	
Connected Clients (2)	~	
18-fe-34-9c-fc-db 192.168.73.102	 3 4 5 5<	
NokiaPhone-4b8816f82 192.168.73.101	 C2 Ta O 	
Disconnected Clients (0)	*	

Connexion du Nokia phone

Si la page d'accueil de Wamp apparaît tout est ok pour continuer. Sinon revoir vos configurations et la cohérence des adresses IP.

3. CONFIGURATION DU MODULE ESP8266

Brancher le module ESP8266 sur le câble USB/TTL

Lancer le logiciel ESP8266Config.exe :

Connectez vous au port série USB/TTL : 9600 COM?? (1)

Chercher les réseaux présents "List AP" (2),

choisir votre réseau Connectify-sb (3),

entrer le mot de passe (4) puis faire "JoinAP" (5).

Sur le logiciel Connectify un nouveau client apparaît avec l'adresse MAC (6) de votre module ESP8266. (Adresse visible en partie sur le SSID de l'ESP8266 (7))

🖳 AppStack ESP	8266 Config V. 1.0.0.0 — 🗆 📩	
ESP8266 Config	Port COM3 V Disconnect Baudrate 9600 V Serial Monitor	Connectify-sb (1 Client) -
Mode Mux AP Set Mode Single Set Mux STA Image: Set Mode Multiple Get Mux FW Version FW Version : 0018000902-AI03 eset Set Mux	AP SSID : 7 ESP_9CFCDB Set AP Get AP Password : Channel : 6 ~ Encryption : Open ~ Get join device	Connected Clients (1) ▼ Image: 18-fe-34-9c-fc-db Image: 192.168.73.102 Image: 192.168.73.102
Enable WD Disable WD Set Baudrate 9600 V Get Baudrate Server (TCP) Port : Listen Close Timeout : Set Timeout Get Timeout Client (TCP,UDP)	IP STA SSID : Connectfy-sb 2 List AP Get AP Password : 0123456789 4 Join AP Quit AP IP : - 5 Get IP	NokiaPhone-4b8816/82 +CWLAP: (0, "FreeWifi", -89, "f4:c +CWLAP: (3, "Connectify-sb", -38, +CWLAP: (2, "free_gm", -77, "2e:93 +CWLAP: (0, "FreeWifi", -76, "2e:93 +CWLAP: (2, "jyp1970", -90, "f4:ca
IP: : : ID: 1 V Protocol: TCP V	Name Encryption Signal FreeWifi_secure Open -90 FreeWifi Open -89 Connectify-sb 3 WPA2_PSK free_gm 3 WPA2_PSK FreeWifi Open -76 FreeWifi Open -77 print server 2302A2 Open -75 iyp1970 WPA2_PSK -90 FreeWifi Open -61	+CWLAP:(0,"FreeWifi",-61,"7a:0 +CWLAP:(0,"FreeWifi",-69,"46 +CWLAP:(0,"FreeWifi",-69,"44:C +CWLAP:(0,"FreeWifi",-69,"44:C +CWLAP:(2,"FREEBOX_SYLVIE_YY", +CWLAP:(2,"FREEBOX_SYLVIE_YY", +CWLAP:(2,"free7G",-62,"7a:04: OK AT+CWJAP="Connectify-sb","0123 OK
http://www.facspok.ccg/appstack.in/		

L'adresse IP de votre IoT apparaît dans connectify : 192.168.73.102.

4. ADAPTATION DU PROGRAMME PIC AUX PARAMÈTRES DU MODULE ESP8266

Rappel des paramètres : HotSpot : Wifi WPA2 : name : Connectify-sb, pass : 0123456789 Serveur Web : 192.168.73.1/projet_db_mesure/ IP ESP8266 : 192.168.73.102

Il faut donc faire des modifications dans le programme ESP8266client.c

Attention au nombre de caracteres ajouté ou retiré pour chaque ligne de code.

//#define ADRWEBSERVEUR "192.168.0.10" //adresse du serveur web (PC sur lequel WAMP est en marche) #define ADRWEBSERVEUR "192.168.73.1" //adresse du serveur web (PC sur lequel WAMP est en marche)

// printf("Host: 192.168.0.10\r\n");//22-2\=20 printf("Host: 192.168.73.1\r\n");//22-2\=20

-150

Reprogrammer le PIC avec le nouveau .hex.

Tester si l'IoT envoie bien les données vers le serveur Web en visualisant les datas de la base de donnée.

😡 http://localh...es/dbvisu.php 🚿 +Iocalhost/projet_db_mesures/dbvisu.php Essai utilisation db mesures Les data sont : N= 1 Date = 2015-06-18 10:12:46 Mesures1 = 100 Mesures2 = -100 N= 2 Date = 2015-06-18 10:13:01 Mesures1 = 100 Mesures2 = 50 N= 3 Date = 2015-06-18 10:18:10 Mesures1 = 131 Mesures2 = 12 N= 4 Date = 2015-06-18 10:19:03 Mesures1 = 100 Mesures2 = 50 N= 4 Date = 2015-06-18 10:19:05 Mesures1 = 100 Mesures2 = 50 N= 5 Date = 2015-06-18 10:19:16 Mesures1 = 131 Mesures2 = 12 N= 6 Date = 2015-06-18 10:19:23 Mesures1 = 100 Mesures2 = 50 N= 7 Date = 2015-06-18 10:19:33 Mesures1 = 255 Mesures2 = 255 N= 8 Date = 2015-06-18 10:19:46 Mesures1 = 100 Mesures2 = 50 N= 9 Date = 2015-06-18 10:20:06 Mesures1 = 253 Mesures2 = 244 N= 10 Date = 2015-06-18 10:20:13 Mesures1 = 100 Mesures2 = 50 Graphique des data de la base db mesures (table : table mesures) : Mesure2(•) Mesure1(•) 250 · • • 200 150 100 -50 -100

5. CONCLUSION

Il est facile du coup de développer en local en dédiant un pc qui fonctionne en mode serveur et en hotspot Wifi.

Extension de l'activité :

Une fois votre développement validé il est possible de dédier un raspberryPi en serveur après y avoir installé LAMP. Un Hotspot Wifi étant préalablement mis en place.

6. ANNEXE

6.1. Code source du fichier PIC :

/*

Programme envoyant des données mesurées par le pic vers une base de données web en utilisant le module ESP8266

/ /

Lorsque le message "appuyer pr envoi " apparaît il faut appuyer sur le bouton central du joystick. Le CIPSTART est alors envoyé et on attend la réponse de ESP8266 : "linked "

Le texte : "Linked : réappuyer" apparaît.

En appuyant de nouveau la requête POST est envoyé par la commande CIPSEND.

Les données envoyant sont les résultats des convertisseurs CAN0 = mesure1 et CAN1) mesure2. Le programme tourne en boucle.

*/

/*

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

Configuration de l'esp8266 :

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

```
activer le lien TCP vers l'adresse du serveurWeb et le port par l'envoi de l'AT :
AT+CIPSTART=\"TCP\",\"192.168.0.10\",80\r\n") le\ permet d'envoyer le caractere "
vérifier la réponse OK et Linked
envoyer le texte suivant qui correspond à la commande AT d'envoi suivi de la requete POST :
"AT+CIPSEND=159\r\n" puis
"
POST /projet_db_mesures/dbvisu.php HTTP/1.1
Host: 192.168.0.10
```

Content-Type: application/x-www-form-urlencoded Content-Length: 21

```
mesure1=11&mesure2=11
"
*/
#include <16F876.h>
```

#device adc=8

```
#FUSES NOWDT
                       //No Watch Dog Timer
                   //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES HS
                      //No Power Up Timer
#FUSES NOPUT
                         //Code not protected from reading
#FUSES NOPROTECT
#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 ADRWEBSERVEUR "192.168.0.10" //adresse du serveur web (PC sur lequel WAMP est en marche)
#define ADRWEBSERVEUR "192.168.73.1" //adresse du serveur web (PC sur lequel WAMP est en marche)
#define allume_LEDVERTE output_high(PIN_C2);
#define LEDROUGE PIN C0
#define LEDJAUNE PIN C1
#define LEDVERTE PIN C2
#define BP MILIEU PIN B4 //BP du milieu du joystick
#int RDA
void RDA isr(void)
{
}
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;

void main()
{
 int8 i=0;
 int8 channel=0;
 char strbuffer[20];
 int8 lchar;
 int8 mesure1=0,mesure2=0;

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(0b1111111); set_tris_c(0b10010000);

initCAN();//initialise le CAN

Init_Ecran();//initialise l'écran LCD Efface_Ecran();printf(Affiche_caractere,"ESP en client "); Lcd_Place_Curseur(2,1);printf(Affiche_caractere,"Connectify"); delay_ms(1000); allume_LEDVERTE;

 $do{$

//acquisition des valeurs de CAN0 et CAN1
mesure1=acquerirCAN(0);
mesure2=acquerirCAN(1);

if(!input(BP_MILIEU)) //si appui alors envoi des données
{
 LCD_Retour_Maison();printf(Affiche_caractere,"Envoi en cours");delay_ms(500);
 //etablir une connexion TCP : avec choix du IP destination et du port

// printf("AT+CIPSTART=\"TCP\",\"192.168.0.10\",80\r\n"); printf("AT+CIPSTART=\"TCP\",\"192.168.73.1\",80\r\n"); testOK(); Efface_Ecran(); printf(Affiche_caractere,"ok"); //test LINKED while(getc()!='L'){}; while(getc()!='i'){}; while(getc()!='n'){}; while(getc()!='k'){}; while(getc()!='e'){}; while(getc()!='d'){}; Efface_Ecran(); printf(Affiche_caractere,"Linked:\nReappuyer...");

while(input(BP_MILIEU)) {}; //attente
//envoyer les donnees en indiquant la taille

printf("AT+CIPSEND=159\r\n"); printf("POST /projet_db_mesures/dbvisu.php HTTP/1.1\r\n");//47octets-2\=45 // printf("Host: 192.168.0.10\r\n");//22-2\=20 printf("Host: 192.168.73.1\r\n");//22-2\=20 printf("Content-Type: application/x-www-form-urlencoded\r\n");//51-2\ = 49 printf("Content-Length: 23\r\n\r\n");//26-4\ = 22 //21=taille de la ligne suivante printf("mesure1=%3u&mesure2=%3u",mesure1,mesure2);//23 donc total = 159

}//if

Efface_Ecran();printf(Affiche_caractere,"Appuyer pr envoi");delay_ms(500);

}while(1);
}//main