TP Programmation logiciel ARDUINO pour Objet connecté ESP8266

Matériels utilisés

- carte nodemcu + ESP8266 (ESP-15 avec 4Mo)
- multimètre USB
- Logiciel arduino
- 1 Raspberry Pi 3 pour faire serveur web ou un PC avec apache
- ou installer apache sur un PC avec page web et le code upload.php pour écrire dans le fichier texte (tout dans /*var*/www)
- fils dupont femelle-femelle

Objectifs

- Installer le programme de développement : IDE Arduino
- Exécuter un programme test (WiFiScan)
- Réaliser plusieurs programmes différents liés au réseau, à l'interfaçage du microcontrôleur et la mesure de consommation

Installation

Télécharger le logiciel Arduino sur Internet – Sélectionner la version 64 bits (salle 203)
Installer le logiciel Arduino sur le poste dans le répertoire de téléchargement
Décompresser le fichier (xz –d ard…tar.xz)
Installer l'archive (tar xvf ard…tar)
Lancer le programme arduino dans le répertoire arduino (./arduino)

Pour ajouter le plugin de compilation pour ESP8266, il faut ajouter l'URL dans Fichier/préférences/URL gestionnaire de cartes additionnelles
http://arduino.esp8266.com/stable/package_esp8266com_index.json
Puis il faut aller dans Outils/Cartes/Gestionnaire de cartes et installer le plugin ESP8266 situé en dernière partie de la liste.
Pour compiler sur un ESP-01S ou ESP-201, il faut sélectionner "generic ESP8266 module", puis flash 1Mo(256K SPIFFS).
La cadence de flash est 115200 bauds pour les "generic ESP8266".
Pour compiler sur un ESP-12E avec la carte de développement USB, il faut sélectionner "nodeMCU 1.0 (ESP-12E module)".
La cadence de flash peut aller jusqu'à 921600 bauds sur certaines cartes nodemcu avec port USB. Probablement que le flash s'effectue depuis le nodemcu et pas en passant par l'ESP8266. Sinon, il faut flasher à 115200 bauds.
Pour compiler un programme il faut effectuer Sketch/Compile ou Croquis/Compile.
Si la carte ESP8266 est directement connectée sur l'ordinateur il suffit de sélectionner "Croquis/Télécharger" pour installer automatiquement le programme, à la condition d'avoir sélectionné le port USB correctement (outils/port).
3 possibilités de cartes :

- Chip sur labdec, il faut connecter le GPIO-00 sur GND pour flasher
- Chip sur une carte avec bouton, le configuration de flash s'effectue automatiquement sans appuyer sur les boutons

Programmation
1) Beaucoup de programmes exemples sont fournis dans Fichiers/Exemples

Connecter l'ESP8266 sur un port USB, sélectionner dans le menu le port de connexion « Outils/port » et le type d'ESP8266 « Outils/Type de carte » Nodemcu 1.0 puis ouvrir la console « outils/moniteur série » en sélectionnant 115200 bauds.
Charger le programme CheckFlashConfig à partir du sous-menu ESP8266, puis compiler et télécharger l'ESP8266.
Le programme indique la taille de la mémoire flash du composant.

2) Charger le programme WiFiScan à partir du sous-menu ESP8266WiFi, puis compiler et Télécharger l'ESP8266.
A présent, vous devez observer la liste des points d'accès Wi-Fi présents dans le bâtiment dans la console.

3) Création du programme « hello world »
Créer un nouveau programme.
Dans la fonction setup() placer les 2 lignes suivantes :

```
Serial.begin(115200);
Serial.println("Hello World");
```

Et dans la fonction loop() :

```
Serial.printf("valeur %d\n",i++);
delay(1000);
```

La variable entière i doit être déclarée et initialisée en globale

4) Le programme serveur web
Créer un nouveau programme.
Dans la partie setup(), il faut paramétrer l'interface WiFi en mode Acces-Point, activer le service http sur le port 80 et définir 2 pages html en les liant chacune à une procédure.

```
WiFi.mode(WIFI_AP);
WiFi.softAP(essidAP, motDePasseAP);
server.on("/", pageAccueil);
server.on("/parametres.html", parametres);
server.begin();
```

Dans la partie, loop(), le serveur attend une requête d'un client.
server.handleClient();
L'objet server doit être déclaré en global et rattaché au port 80 :
ESP8266WebServer server(80);
La procédure pageAccueil() contient la fonction suivante :

```
server.send(200, "text/plain", "hello from esp8266!");
```

La procédure parametres() doit renvoyer l'adresse IP du serveur dans une page html.
Les 2 bibliothèques suivantes doivent être ajoutées pour compléter le programme :

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

Vous devez vérifier le fonctionnement de votre programme à l'aide de votre smartphone. Il faut se connecter sur le réseau Wi-Fi créé puis à l'aide du navigateur vous consultez les 2 pages web créées.

5) Le programme capture de données et sauvegarde sur un serveur web.
Ecrire un programme qui mesure la température (valeur aléatoire ou quelconque) et renvoie une fois par minute la température sur un serveur web à l'aide du protocole http en mode get. Le script sur le serveur peut être soit un shell-script soit un script php.

L'url sera de la forme : http://192.168.0.x/update.html?temperature=12

Vérifiez que votre donnée apparaît dans le fichier texte du serveur.

6) la veille profonde
Le microcontrôleur dispose d'une fonction de veille profonde consommant très peu de courant (environ 20 microampères) alors que le fonctionnement du microcontrolleur seul consomme environ 10 milliampères et 80 à 120 milliampères avec le WiFi activé. La fonction est :
ESP.deepSleep(microsecond)
Dans ce mode, seul l'horloge fonctionne (Real Time Clock) et le port GPIO16 passe à l'état DOWN à la fin du décomptage.
La reprise du programme à la fin de la veille provoque un reset et relance le programme depuis le début (exécution de la fonction setup()).
Pour cela, il faut relier la broche du port GPIO16 (D0) à la broche RESET.
Reprendre le programme Hello World et remplacer la fonction delay par deepSleep.

7) La gestion de fichier
Il est possible de créer quelques fichiers sur un système de gestion de fichiers SPIFFS avec les ESP8266 à la condition de disposer d'un peu de mémoire pour cela. Les programmes exécutables disposent de la première partie de la mémoire flash jusqu'à un maximum de 1Mo. La suite de la mémoire est mise à disposition pour le « file system ».
Les fonctions suivantes seront utilisées :

```
#include <FS.h> // pour SPIFFS
  SPIFFS.begin(); // Pour accéder au système de fichier
  SPIFFS.format(); // Pour formater l'espace mémoire. Cette fonction dure environ 30 secondes et efface tout.
  SPIFFS.end() ; // Pour fermer l'accès au système de fichier
  File f = SPIFFS.open("/test.txt", "w"); // r w a pour lire écrire et ajouter
  f.println("premiere ligne"); // pour ecrire une ligne dans le fichier
  f.available() // retourne le nombre de caractères à lire. Fin de fichier avec une valeur à 0
  f.readStringUntil('\n'); // pour lire une ligne de texte
  f.close();
```

Montrez votre programme à l'enseignant lorsque vous aurez écrit et affiché le contenu d'un fichier.

8) Client MQTT
A l'aide de la bibliothèque Arduino MQTT, écrire un programme client MQTT publisher qui va écrire dans une arborescence une valeur simulant une température. Il faudra vérifier que les données sont bien écrite à partir d'un client MQTT subscriber depuis un PC.
Installer PubSubClient by knoleary

9) Objet connecté home-assistant
utiliser la bibliothèque Arduino home assistant pour qu'un objet domotique apparaisse dans l'application Android Home.
Installer home-assistant-integration by davidchyrzynski

Ecouter le broker MQTT avec le subscriber : mosquitto_sub -h localhost -t "#" -d
Exécuter le code exemple : home-assistant-integration/nodemcu

Lorsque nodemcu s'exécute, taper la commande suivante pour éteindre et allumer la led :
mosquitto_pub -h 192.168.1.50 -t "aha/f4cfa26c164d/789/cmd_t" -m OFF et ON

installer home assistant sur le smartphone avec google play

Faites une démonstration de votre programme à l'enseignant lorsqu'il fonctionne.

A tester mais semble plus simple ?
https://github.com/plapointe6/HAMqttDevice

Semble plus riche
https://github.com/dawidchyrzynski/arduino-home-assistant

Référence
http://www.whatimade.today/flashing-the-nodemcu-firmware-on-the-esp8266-linux-guide/

10) Mesure de consommation sur l'ESP8266
Ecrire un programme qui effectue une boucle de calcul pendant 10 secondes et relevez la consommation à l'aide du multimètre USB : mA
Faire une ou plusieurs boucles avec les 2 instructions suivantes :
x = x * 2 ;
x = x / 2 ;

Ecrire un programme qui effectue une boucle d'émission de messages HTTP pendant 10 secondes et relevez la consommation à l'aide du multimètre USB : mA

Si vous n'observez pas de différences significatives, il faudra sans doute couper/réactiver l'alimentation du module RF (Wi-Fi) avec les commandes :
wifi_set_sleep_type(LIGHT_SLEEP_T);
ou, si l'on veut que le module Wi-Fi ne s'active au démarrage suivant
ESP.deepSleep(100000, WAKE_RF_DISABLED);
ESP.deepSleep(100000, WAKE_RF_DEFAULT);

Ecrire un programme qui effectue une pause pendant 10 secondes (fonction sleep) et relevez la consommation à l'aide du multimètre USB : mA

Ecrire un programme qui effectue une veille profonde pendant 10 secondes et relevez la consommation à l'aide du multimètre USB : mA

A l'aide de la mesure, indiquez le temps théorique de fonctionnement d'un programme effectuant un calcul continu et fonctionnant sur une batterie de 2000 mA.h : …..

Exemple : si la conso est 200mA, donc 2000 mA.h / 200 mA = 10 h de fonctionnement

A l'aide de la mesure, indiquez le temps théorique de fonctionnement d'un programme effectuant un calcul d'1 seconde et transmettant pendant 1 seconde et fonctionnant sur une batterie de 2000 mA.h : …..

A l'aide de la mesure, indiquez le temps théorique de fonctionnement d'un programme effectuant un calcul d'1 seconde, transmettant pendant 1 seconde et effectuant 8 secondes de pause et fonctionnant sur une batterie de 2000 mA.h : …..

A l'aide de la mesure, indiquez le temps théorique de fonctionnement d'un programme effectuant un calcul d'1 seconde, transmettant pendant 1 seconde et effectuant 8 secondes de veille profonde (deep-sleep) et fonctionnant sur une batterie de 2000 mA.h : …..

11) Mesure de consommation avec le wattmètre connecté : Meross MSS310
Pour cela, il faut installer l'application Meross sur votre smartphone. Ensuite, il faut brancher le wattmètre Wi-Fi : Meross MSS310 sur une prise électrique. Avec un appui long sur le bouton (clignote alternativement vert-orange), cela provoque un *factory reset*. Avant d'effectuer l'association, vérifiez que la prise est bien en mode Access-Point en consultant la liste des réseaux Wi-Fi. L'association Wi-Fi s'effectue à l'aide de l'application. Lorsque le wattmètre est connecté au WiFi, il faut brancher un écran dessus et mesurer sa consommation à partir de l'application. Faites de même avec une unité centrale, un Raspberry Pi et un ESP8266, Relevez vos mesures ci-dessous :

Ecran	:	W donc	mA
PC	:	W donc	mA
Raspberry PI	:	W donc	mA
ESP8266	:	W donc	mA

12) ESP8266 : mise à jour Over the Air (OTA)
Ecrire un programme pour l'ESP8266 qui affiche son numéro de version (en commençant par 1.0), puis qui se connecte sur le réseau Wi-Fi et affiche le numéro de version du code de votre serveur web/apache (apt-get install apache2). Si la version du serveur est différente de la version dans votre code, il faut lancer la mise à jour.

Pour cela il faut la bibliothèque : #include <ESP8266httpUpdate.h>

Charger le fichier texte contenant le numéro de version :

```
HTTPClient http;
http.begin("http://192.168.0.x/version.txt");
http.GET();
Serial.println( http.getString() );
```

Effectuer la mise à jour si la version est différente sur le serveur

```
t_httpUpdate_return ret = ESPhttpUpdate.update("http://192.168.0.x/monprog.nodemcu.bin");
```

Montrez votre résultat à l'enseignant

13) UART entre PC et ESP8266

L’objectif est de transmettre une valeur de température en degrés depuis le PC vers l’ESP8266 et qu’ensuite la valeur soit transmise sur le serveur web de l’exercice 5). A l’issue, un message sera renvoyé au PC pour indiquer que la valeur a été envoyée sur le serveur web.

```
Serial.available() // Si valeur de retour supérieure à zéro alors il y a un message à lire
String message = Serial.readString() ; // Pour lire le message reçu
```

Pour envoyer un message à l’ESP utilisez la console d’Arduino.
Montrez votre réalisation à l’enseignant.

14) UART entre 2 ESP8266

L’objectif est d’effectuer un échange de message entre 2 ESP8266 en utilisant un port série UART RS-232. Pour cela, il faut 2 ESP8266 avec un câblage entre broches pour transmettre vos données. Identifiez le branchement à réaliser sans alimenter les ESP8266. Lorsque votre branchement est prêt, notez le sur votre feuille et appelez l’enseignant pour qu’il puisse vérifier.

Le premier ESP8266 envoie le message "allumer la LED" ou "eteindre la LED" et le deuxième ESP8266 effectue l’action et renvoie le message "allumer ok" ou "eteindre ok".

Pour cela, il faut gérer 2 broches avec la norme UART RS232 avec la bibliothèque SoftwareSerial.

```
#include <SoftwareSerial.h>
SoftwareSerial serie(5, 4, false, 256); // RX, TX, bit inverted, buffer size. D1=GPIO5 D2=GPIO4
serie.begin(115200);  // pour définir le débit de communication
```

Lorsque votre code fonctionne, montrez le résultat à votre enseignant.

15) microcontrôleur et superviseur

L’objectif est d’améliorer la disponibilité d’un microcontrôleur en implantant une fonction de supervision dans un autre microcontrôleur et ainsi le redémarrer en cas d’indisponibilité.

Vous conservez le branchement de l’exercice précédent avec 2 ESP8266. L’objectif est d’avoir un programme qui s’exécute sur le microcontrôleur principal et d’avoir un superviseur sur l’autre ESP8266. Le rôle du superviseur est d’échanger régulièrement un message avec le microcontrôleur principal pour vérifier sa disponibilité. Cette fonction se comme heartbeat. Lorsque le microcontrôleur ne fonctionne plus, il ne répondra plus au message. Dans ce cas, il faudra le rebooter depuis le superviseur pour qu’il redémarre et continue de fonctionner. Le message échangé sera "hello" et la réponse sera "hello ok". Pour faire planter le micro-contrôleur, il faut par exemple effectuer une division par zéro.

Que faut-il ajouter pour implanter cette fonction ?
Lorsque votre code fonctionne, appelez l’enseignant pour faire votre démonstration.