Treball de Fi de Màster

Màster en Enginyeria Industrial

Connexió wifi de baix cost i amb finalitats didàctiques per a la placa Open18F4520

MEMÒRIA

Autor: Aziz Mahouti
Director: Juan Manuel Moreno Eguílaz

Convocatòria: Quadrimestre de tardor 2018

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

El projecte consisteix en la programació del microcontrolador inclòs en la placa OpenPic18F5420, fent servir codi C, el MPLAB X IDE v 4.20 i el compilador XC8, per tal que sigui capaç de llegir dades d'un sensor digital de temperatura DS18B20 i enviar-les sense fils (Wifi) tant a un ordinador que actua com a magatzem de dades com a un altre microcontrolador que actua com a receptor de les dades i que mostra la temperatura rebuda en una pantalla LCD.

Posteriorment, també s'ha escrit un petit programa en llenguatge PHP per tal que l'ordinador que actua com a magatzem de dades guardi les dades que va rebent en un arxiu de text.

Per acabar, s'ha escrit el programa necessari per al microcontrolador que actua com a receptor de dades i així aconseguir que sigui capaç de rebre les dades enviades per Wifi i mostrar-les en una pantalla LCD.

Índex

ETSEIB

1. Glossari

µC: Microcontrolador

ASCII: “American standard code for information interchange”

CPU: “Central processing unit”

EEPROM: “Electrically erasable programmable read only memory”

ESP: ESP8266

I2C: “Inter-integrated circuit”

IIoT: “Industrial internet of things”

IoT: “Internet of things”

IT: “Information tecnologies”

KPI: “Key procés indicator”

LCD: “Liquid cristal display”

PIC: PIC18F4520

RAM: “Random acess memory”

SPI: “Serial peripheral interface”

UART: “Universal asynchronous receiver-transmitter”

Wifi: “Wireless fidelity”

2. Introducció

2.1. Necessitat de recollida de dades

En general, l'objectiu principal de qualsevol empresa, negoci, organització... és el de cobrir una necessitat, i per a tal fi, aquestes ofereixen serveis o productes al mercat capaços de cobrir-les.

Per tal de cobrir aquestes necessitats (situació A) i passar a una situació B on aquestes han estat cobertes amb un bé o servei, cal que es produeixi un procés.

Això posa de manifest que en qualsevol empresa o entitat s'estan produint processos contínuament per tal de cobrir necessitats, i no només això, sinó que donat que qualsevol empresa o entitat, en general, es troba en un entorn on més empreses intenten cobrir la mateixa necessitat, apareix la necessitat de realitzar aquests processos de la forma més òptima possible per tal de mantenir la competitivitat de la mateixa dins del mercat.

Per mantenir la competitivitat d'una entitat, donat que tant la tecnologia com les necessitats dels clients canvien contínuament, és necessari mantenir un procés de millora continua, i així evitar que la competència tingui un avantatge competitiu mitjançant l'oferta d'un producte millor, ja sigui pel que fa a preu, qualitat o rendiment.

Per millorar el procés actual, s'ha de tenir el màxim coneixement possible tant del procés actual com de les possibles tecnologies que vagin apareixent. Per tal d'estar atent dels possibles avanços tecnològics que es vagin produint i que poguessin ajudar al procés actual, cal estar al dia de possibles publicacions, presentacions, fires, o a allò que la competència està implementant, però abans d'això, és vital entendre els KPI's ("Key Process Indicator") del procés de fabricació actual, com per exemple, els colls d'ampolla de l'empresa o els temps de cicle de cada procés, i així determinar que és allò que ens podria ajudar a millorar.

Per tant, per entendre el procés de fabricació actual, s'ha de recollir dades dels KPI's per tal de poder analitzar-les posteriorment i determinar quines accions podrien ser interessants, i donat que en general, a qualsevol procés real existeix variabilitat en els valors que poguéssim recollir com per exemple el temps que es tarda a realitzar una acció, cal recollir una mostra estadísticament significativa, i aquesta, en general, sol ser elevada.

Per tant, la recollida de dades, donat que ha de ser suficientment elevada, implica una quantitat de temps elevada, i si aquestes dades les ha de recollir una persona, pot implicar

un cost elevat.

Per tant, la conclusió a la que s'arriba és que la recollida manual d'aquestes no sol ser la forma més encertada, i la millor opció sembla ser la recollida d'aquestes de forma automàtica.

2.2. "Industrial Internet of Things"

Basant-se en la idea de que a la indústria existeix la necessitat de recollir dades per tal de ser capaços d'entendre la cadena de valor actual i així poder optimitzar-la, va aparèixer la idea de l'IoT, el qual es basa en tres pilars:

- Recollida de dades (monitorització del procés).
- Intercomunicació (de les diferents màquines i elements que intervenen en el procés).
- Presa de decisions (de les mateixes màquines en funció de les dades).

Tenint dades de totes les variables que intervenen en el procés, les pròpies màquines són capaces de determinar què s'ha de fer, com s'ha de fer i quan s'ha de fer, i així evitar al màxim possible tot allò que no aporta valor al producte o no forma part de la cadena de valor, a més de facilitar la millora del procés, ja que amb totes les dades recollides, es pot simular de forma força simple una possible millora en aquest.

Aquesta idea ha generat molt interès a la indústria durant els últims anys, tant que es parla de la implementació de l'IoT com la quarta revolució industrial (les tres anteriors són la implementació de les màquines de vapor a la indústria a finals del segle XVIII, la utilització de l'electricitat com a força motriu a finals del segle XIX, i la utilització de les IT ("information technologies") el qual es podria traduir com la utilització de sistemes electrònics a la indústria (sobretot ordinadors) implementada a mitjans del segle XX).

2.3. Objectius del projecte

Donats els interessants avantatges de l'IoT, es va decidir realitzar un projecte en què s'implementessin les idees d'aquest.

Una forma d'implementar això, és amb un sensor capaç de recollir les dades que es vol estudiar, un mòdul wifi capaç de connectar els diferents elements que intervenen en la transmissió de dades, una base de dades on guardar i tractar les dades recollides, i un microcontrolador per donar les ordres necessàries al mòdul wifi i al sensor, i intercomunicar el sensor amb el mòdul wifi.

A partir d'aquesta idea, l'objectiu del projecte és la unió dels elements comentats anteriorment, i programar el microcontrolador perquè sigui capaç de recollir dades d'un sensor, tractar-les i enviar-les mitjançant un mòdul wifi a una base de dades o a un altre microcontrolador. El conexeiment adquirit permetrà millorar algunes pràctiques de laboratori del Grau o del Màster en Tecnologíes Industrials de l'ETSEIB.

2.4. Abast del projecte

Donat l'entorn acadèmic en el qual es realitza el projecte, a més que seria car i força arriscat implementar aquestes idees sense una experiència suficient en un entorn industrial, s'ha decidit utilitzant la placa Open18F4520, la qual treballa amb un microcontrolador PIC18F4520 de la marca Microchip, crear les llibreries necessàries en codi C utilitzant el programa MPLAB X v4.20, que la mateixa empresa Microchip facilita, amb llicència gratuïta per a aquestes tasques, i el compilador XC8 també de llicencia gratuïta, per tal de poder llegir dades del sensor digital de temperatura DS18B20 (Maxim) i enviar-les a traves d'el mòdul Wifi ESP8266 a una base de dades, i un cop rebudes les dades a la base de dades es grafiquen mitjançant el programa KST 2.0, per tal de poder veure la temperatura del sensor en temps real.

A més, també es realitzarà un segon programa per tal d'enviar les dades a una segona placa Open18F4520, la qual rebrà les dades i les mostrarà en una pantalla LCD 22 de la marca WaveShare.

2.5. Motivació

Donats els interessants aspectes de l'IoT, es va decidir realitzar un treball que implementes els principis d'aquests conceptes amb el menor cost possible, i donat que el departament d'Enginyeria Electrònica ja tenia gran part del material mencionat anteriorment, es va

decidir treballar amb aquest tipus de microcontrolador (PIC18F4520), encara que amb altres models també s'hagués pogut realitzar la mateixa tasca.

S'ha decidit treballar amb un sensor digital de temperatura, que també s'hagués pogut substituir per un altre sensor i llegir algun altre tipus de dada, com per exemple utilitzar un giroscopi i mesurar l'orientació respecte de la direcció de la gravetat de qualsevol element al qual aquest estigues enganxat, però donat que el Departament d'Electrònica ja tenia aquest sensor de temperatura, a més que qualsevol sensor segueix més o menys el mateix principi pel que fa a mesura de dades, amb diferencies en el protocol utilitzat per transmetre les dades al microcontrolador (SPI, I2C,UART...), les direccions de lectura i singularitats pròpies de cada sensor, s'ha decidit treballar amb aquest, ja que per l'objectiu del projecte, el sensor utilitzat no té gaire importància.

També s'ha decidit treballar amb el mòdul wifi ESP4866, que tot i que el Departament no el tenia, és un mòdul que es pot trobar per aproximadament 2 €, i per tant, donat que un dels objectius del projecte és que aquest tingui el mínim cost possible, s'ha decidit que aquest és el mòdul més adequat per a la tasca desitjada, ja que permet la possibilitat de treballar amb el microcontrolador mencionat anteriorment, a més de ser el més econòmic del mercat.

3. Introducció al material utilitzat

3.1. Microcontrolador

Tal com s'ha dit anteriorment, el microcontrolador utilitzat en aquest projecte és el PIC18F4520 de la marca Microchip.

Fig. 1. Fotografia del Pic18f4520. Font: [11].

Juntament amb una CPU de 8 bits, aquest incorpora internament una memòria de programa de 32 kb, una memòria RAM estàtica de 1532 bytes i 256 bytes de EEPROM.

Un altre element del microcontrolador que s'utilitza és l'oscil·lador intern que té aquest, el qual pot treballar a entre 31 kHz i 8 MHz, tot i que es podria incorporar un oscil·lador extern de fins a 48 MHz si es consideres que per la implementació del programa requerís una major velocitat. Tot i això, en aquest cas, s'ha decidit treballar amb el mateix oscil·lador que el microcontrolador porta internament, ja que 8 MHz són suficients per a l'aplicació desitjada, a més de que un oscil·lador extern encariria el projecte.

Un altra propietat del PIC és que aquest model està optimitzat per treballar a 5 V, el qual en certa forma podria ser un inconvenient, ja que el mòdul wifi amb el qual es treballa té un voltatge nominal de 3,3 V, i si treballéssim amb el voltatge nominal del PIC, podríem cremar el mòdul wifi (si no s'incorpora un sistema per retallar la tensió de les dades que surt del PIC i van al ESP8266), a més de necessitar amplificar a 5 V les dades que surten del ESP per tal que el microcontrolador les pugui entendre. Tot i això, donat que el microcontrolador pot treballar a voltatges d'entre 2,0 i 5,5 V, si s'alimenta aquest a 3,3 V, els senyals que

surten del mateix estan a 3,3 V, a més de que a aquest voltatge, es pot llegir dades on la tensió màxima és de 3,3 V, evitant la necessitat d'instal·lar cap tipus d'amplificador o retallador de senyal.

Per acabar, tot i que es podria seguir parlant de molts altres elements i propietats que té aquest microcontrolador, seria interessants parlar sobre els pins d'entrada i sortida de dades, concretament els pins RC6 i RC7, els quals estan optimitzats per treballar com a pin d'entrada (RC) i sortida (TX) de dades en mode de comunicació UART (tot i que qualsevol pin podria ser utilitzat per transmetre dades utilitzant aquest protocol, la utilització d'aquests dos pins permet utilitzar algunes funcions hardware que el pic incorpora, com per exemple la utilització de "flags" que indican quan s'ha rebut una dada). Les propietats d'aquests pins es tractaran més en profunditat en l'apartat 4.2.2.1 quan es parli del protocol de comunicació UART i la seva implementació.

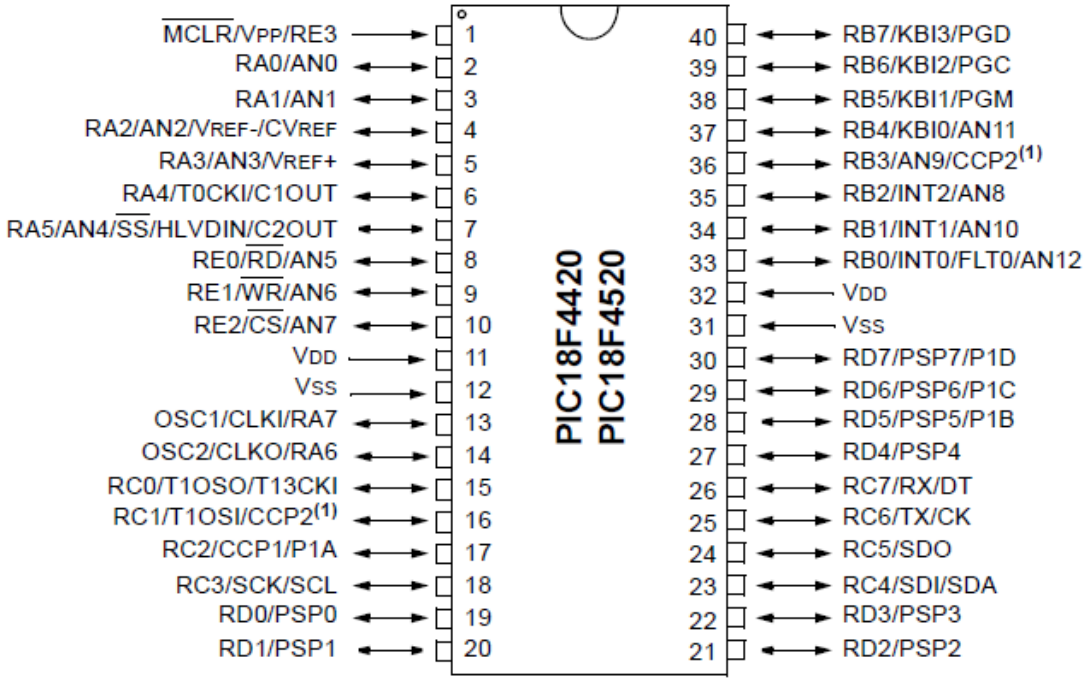


Fig. 2. "PinOut" del PIC18F4520. Font: [11].

3.2. Mòdul Wifi

El mòdul wifi utilitzat, com ja s'ha dit, és l'ESP8266, fabricat per l'empresa Espressif, del qual es poden diferenciar diferents models, tal com es pot veure a la Fig. 3.

En general, en quant a funcionament, tots funcionen de la mateixa manera internament, ja que tots tenen el mateix microprocessador (l'ESP8266). Tot i això, existeixen diferències significatives entre cada model tant de software com de hardware.

En quant a les diferències de hardware, està clar que tots visualment són diferenciables, i això és degut al fet que alguns incorporen elements que altres no porten.

Fig. 3. Diferents models de plaques amb l'ESP8266. Font: [12].

Una de les diferències més significatives és el nombre de pins que surten de la placa sobre la qual van instal·lats, és a dir, el nombre de cables que es poden connectar a aquesta. Aquesta diferència és força significativa, ja que cada pin o sortida té una funció determinada, i per tant la incorporació de pins extra implica la incorporació de funcions extra.

A les següents imatges (Figs. 4 i 5), es pot veure la configuració dels pins d'aquests, i la principal diferència que es pot veure és que a l'ESP 1 hi ha 8 pins, i a l'ESP12 n'hi ha 24.

Aquesta diferència, principalment, afecta en el protocol de comunicació que accepta l'ESP.

En el primer cas, l'ESP 1, només es permet la comunicació tipus UART, i en canvi, en l'ESP12, també es permeten els protocols SPI i I2C.

Tot i això, l'únic protocol que es pot utilitzar amb el microcontrolador desitjat és l'UART, ja que aquest protocol és l'únic a traves del qual ens podem comunicar amb l'ESP mitjançant comandaments AT, i per la resta de protocols, s'hauria d'utilitzar comandaments Arduino.

Per tant, donat que només necessitem els pins de la comunicació UART, el millor model que es pot utilitzar en aquest projecte és el model ESP 1, el qual incorpora tots els elements necessaris i a més és el més barat. Aquest es pot trobar per aproximadament 1,50 €, a diferéncia de l'ESP 12 que costa al voltant de 3 €.

Un altra diferència que podria ser significativa, o podria donar problemes durant la implementació del projecte, és la versió de l'ESP, ja que aquest té diferents versions, i si la versió en la que s'està treballant és massa antiga, podria donar-se el cas de que certs comandaments no estiguessin incorporats.

A més, cal tenir en compte que es podria passar que un ESP concret no fos capaç de suportar una certa versió, donat que, en funció del model o el moment en què aquest ha estat fabricat, el tipus de memòria que incorpora podria no ser suficient.

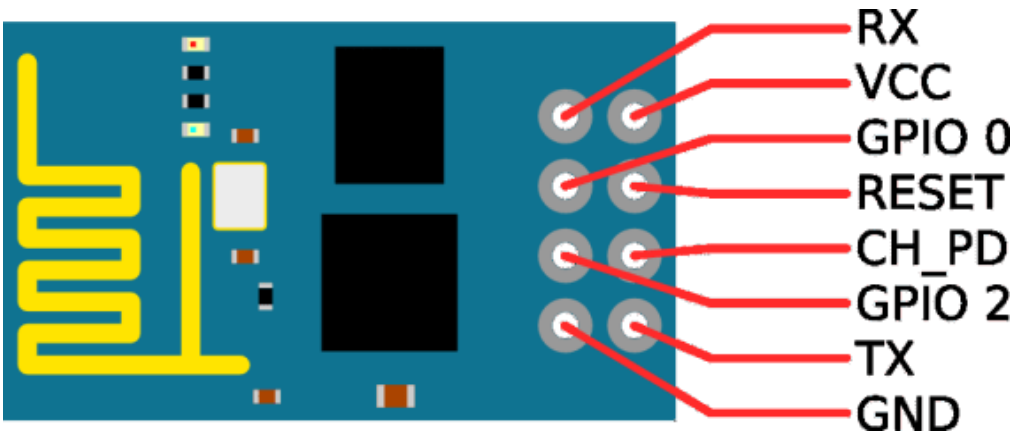


Fig. 4. ESP 1 "PinOut". Font: [13].

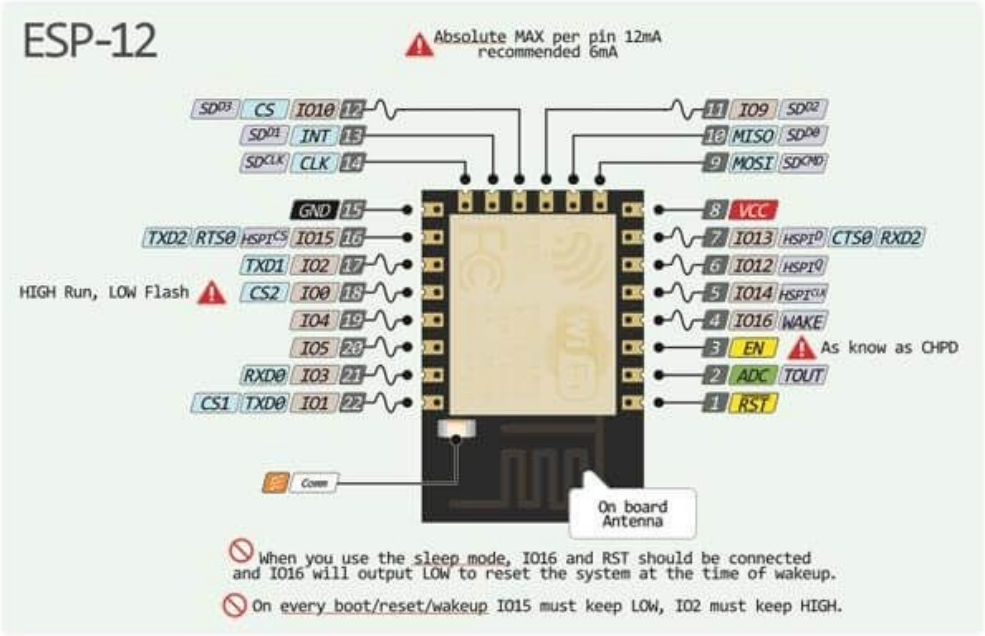


Fig. 5. ESP 12 “PinOut”. Font: [14].

En aquest cas, es treballa amb la versió 0.9.5.2 [1].

A continuació es presenta la forma d’instal·lar-la.

3.2.1. Actualització de l’ESP

Abans de començar amb l’actualització, cal tenir un seguit d’elements per poder realitzar aquesta tasca. El primer és un convertidor USB-TTL, el qual permet enviar comandaments i informació entre una connexió USB i un port UART.

Un convertidor d’aquest tipus es pot trobar, per exemple, a Amazon per aproximadament 2 € (Fig. 6), i no només es farà servir durant l’actualització de l’ESP, sinó que més endavant, durant la implementació del projecte també es farà servir per determinar si la comunicació entre l’ESP i el PIC s’està realitzant de forma correcta.

Fig. 6. Convertidor USB RTT. Font: [15].

Un cop tinguem l'USB-TTL es possible que l'ordinador, per poder treballar amb aquest element, requereixi algun driver, i per tant, si aquest fos el cas, caldria buscar a internet quin driver requereix, (en aquest cas no hi ha links, ja que els divers utiitzats dependran del convertidor comprat).

Posteriorment ens hem de descarregar e instal·lar el programa RealTerm [2].

Un cop instal·lat el programa Realterm, es procedirà a connectar l'ESP amb l'USB-TTL. Les connexions que s'han de realitzar són les següents (Fig. 7):

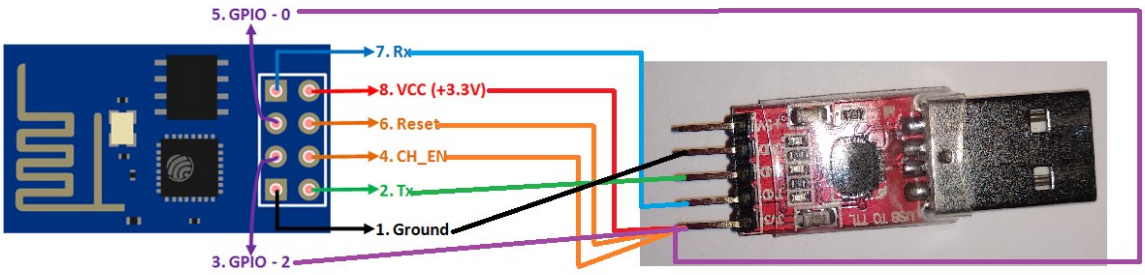


Fig. 7. Esquema de les connexions entre l'USB-TTL i l'ESP.Font: [Pròpia].

ESP Ground	⟷	USB-TTL Ground
ESP Vcc	⟷	USB-TTL 3,3 V
ESP Tx	⟷	USB-TTL Rx
ESP Rx	⟷	USB-TTL Tx
ESP Chanel Enabler	⟷	USB-TTL 3, 3V
ESP Reset	⟷	USB-TTL 3,3 V
ESP GPIO-0	⟷	USB-TTL 3,3 V
ESP GPIO-2	⟷	USB-TTL 3,3 V

Fig. 8. Taula amb les connexions a realitzar. Font: pròpia.

Donat que moltes connexions van a parar al mateix pin (Vcc), a més de que es podrà donar el cas de que el convertidor USB-TTL no fos capaç d'alimentar l'ESP amb la suficient petencia, és recomanable, utilitzar els pins marcats a la (Fig. 10) com a referència (GND) i font de tensió (Vcc).

Per tal de realitzar aquesta connexió, cal recordar connectar tant els pins de terra de l'ESP com els del convertidor al mateix GND, per assegurar que tots treballen amb la mateixa referéncia.

Posteriorment, s'ha de connectar els pins que hagin d'anar a 3,3 V amb la connexió de 3,3 V de la placa (Fig. 10).

Si no, també es pot utilitzar una "breadboard" (Fig. 9),(les quals es poden trobar per aproximadament 2 €), per realitzar les connexions.

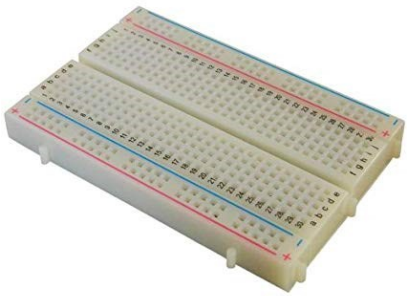


Fig. 9. Breadboard. Font: pròpia.

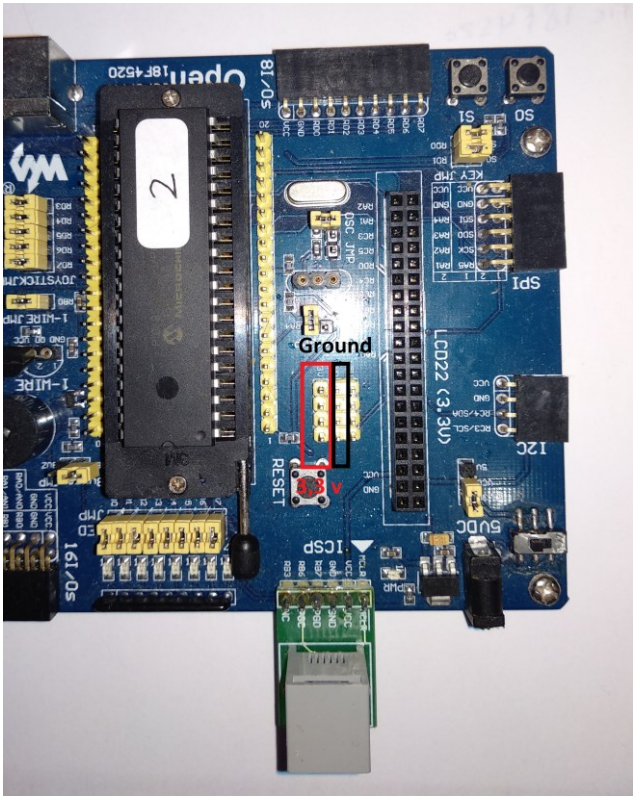


Fig. 10. Pins a 3,3 V i GND de la placa OpenPic18F4520. Font: pròpia.

Un cop realitzades les connexions, es connecta l'USB a l'ordinador i s'obra el programa RealTerm. Dins el programa, a la pantalla *display*, configurem la comunicació perquè ensenyi caràcters ASCII (Fig. 11).

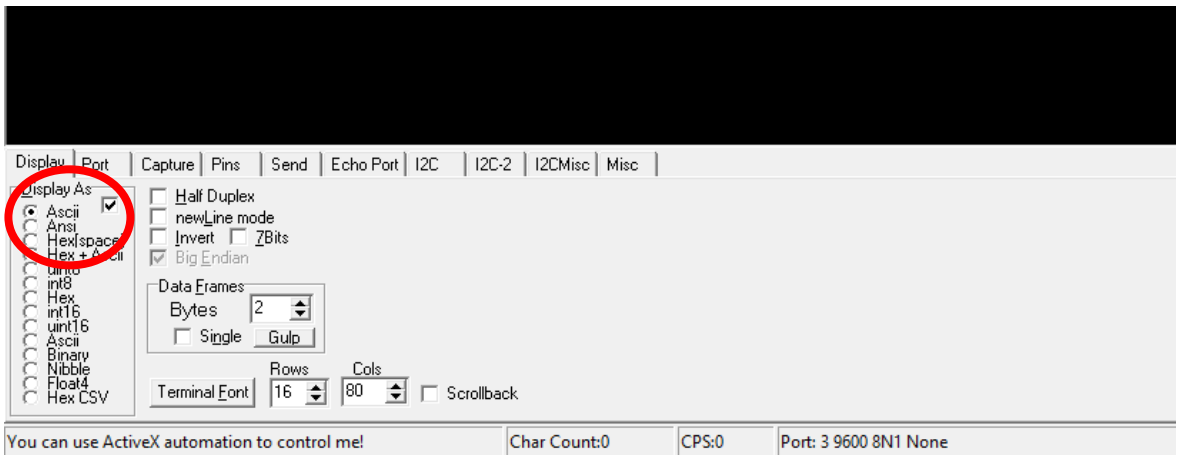


Fig. 11. Configuració de l'apartat *Display*. Font: pròpia.

ETSEIB

Posteriorment, anem a l'apartat *port* i trobem el port al que està connectat el convertidor USB-TTL. Per determinar-ho, podem anar a l'inici del Windows, i obrir l'administrador d'equips. Un cop allà, obrim la pestanya de l'administrador de dispositius. Allà, hauríem de veure l'apartat "ports", i el port en el que està connectat el convertidor, tal com es pot veure a la Fig. 12.

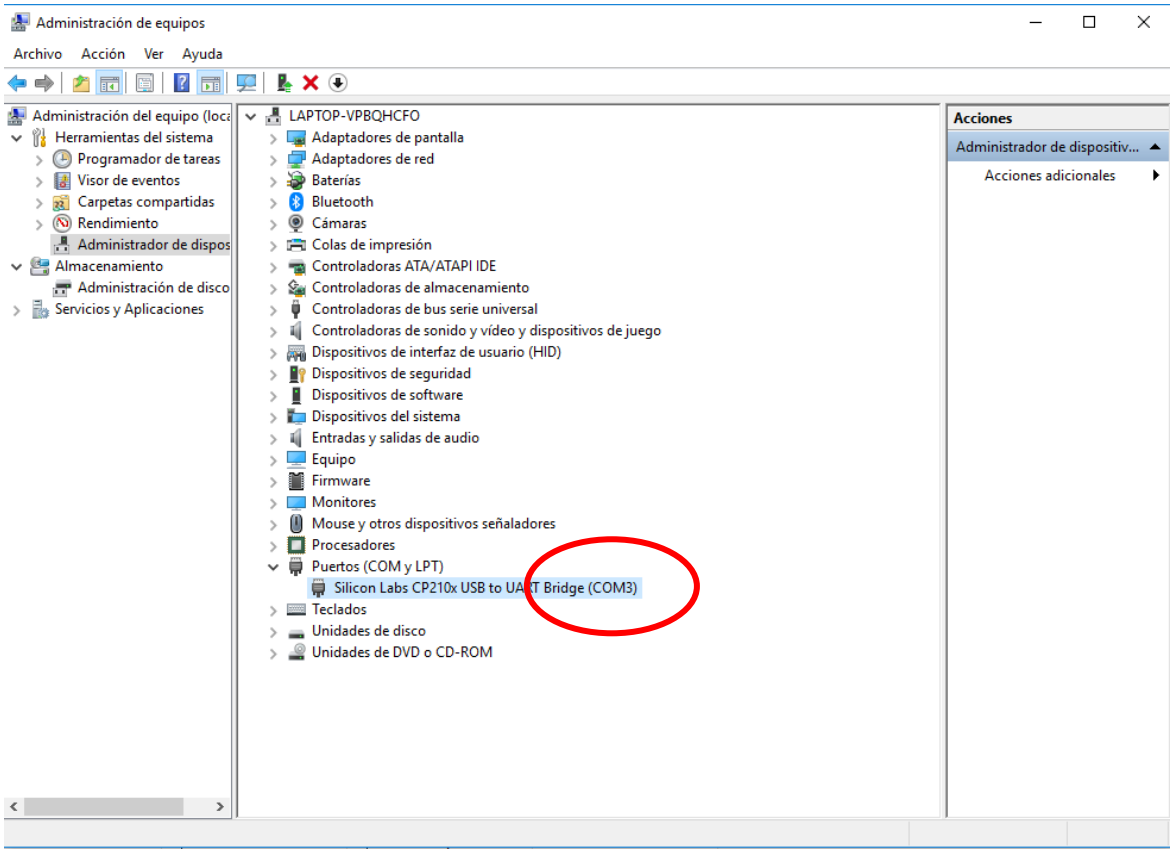


Fig. 12. Determinació del port al que està connectat l'USB-TTL. Font: pròpia.

Un cop determinat a quin port està connectat el convertidor, anem al RealTerm, i seleccionem el port determinat (Fig. 13). També s'ha de configurar la resta d'opcions que apareixen en aquesta pestanya, per tal que la comunicació es produeixi de forma adequada. Les opcions s'han de configurar tal com apareixen a la Fig. 14 (sense bit de paritat, 8 bits per dada, 1 bit d'aturada, i sense hardware flow control).

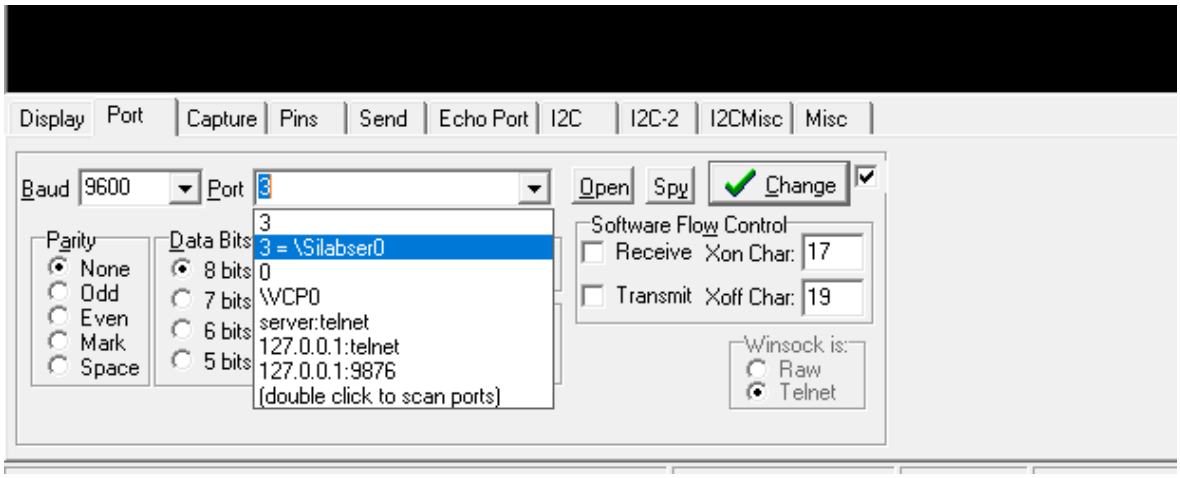


Fig. 13. Selecció del port al que està connectat l'USB-TTL. Font: pròpia.

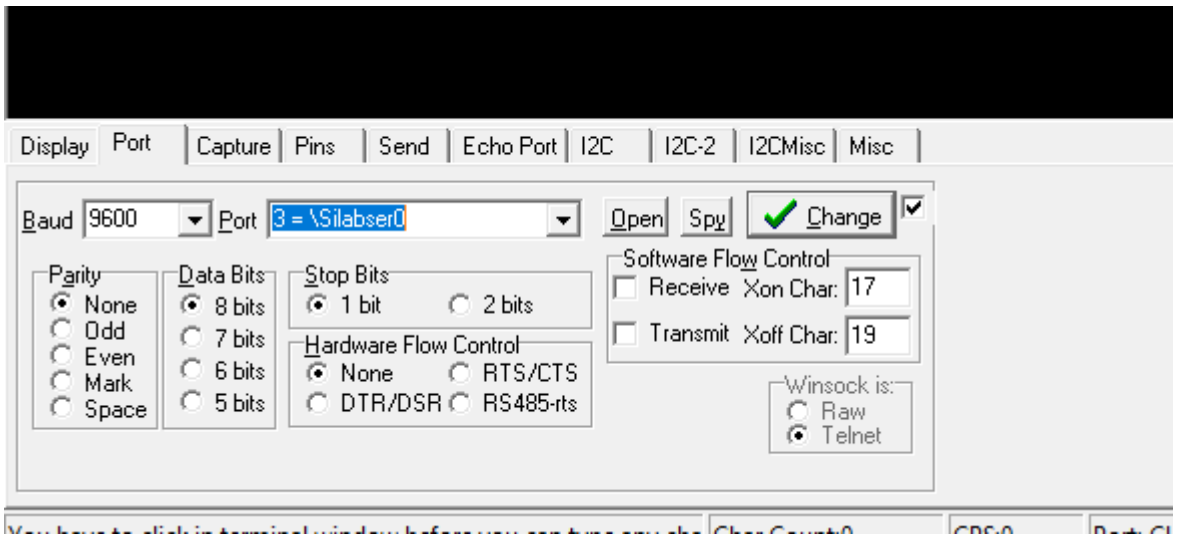


Fig. 14. Configuració de la comunicació entre l'USB-TTL i l'ESP. Font: pròpia.

Posteriorment, s'ha de determinar el "Baudrate" (velocitat de transmissió de dades) al que treballa l'ESP. Normalment, de fàbrica vénen configurats a 115200 bauds/s, tot i que pot ser que en algun cas la velocitat sigui diferent, i per tant, per determinar-la, s'haurà d'anar seleccionant les diferents velocitats i enviant el comandament AT, tal com es veu a la Fig. 15, (cal tenir en compte que cada cop que es canvia la velocitat de transmissió a l'apartat *Port*, cal desactivar i activar la pestanya *Open*, per tal que s'activi la comunicació a la nova velocitat). També s'ha de seleccionar les pestanyes +CR i +CF de la pestanya *Send* perquè la comunicació es realitzi de forma adequada.

Si tot va bé, un cop determinada la velocitat adequada, hauríem de rebre el missatge OK després d'enviar el comandament *AT* i clicar el botó *Send ASCII*, tal com es pot veure a la Fig. 15.

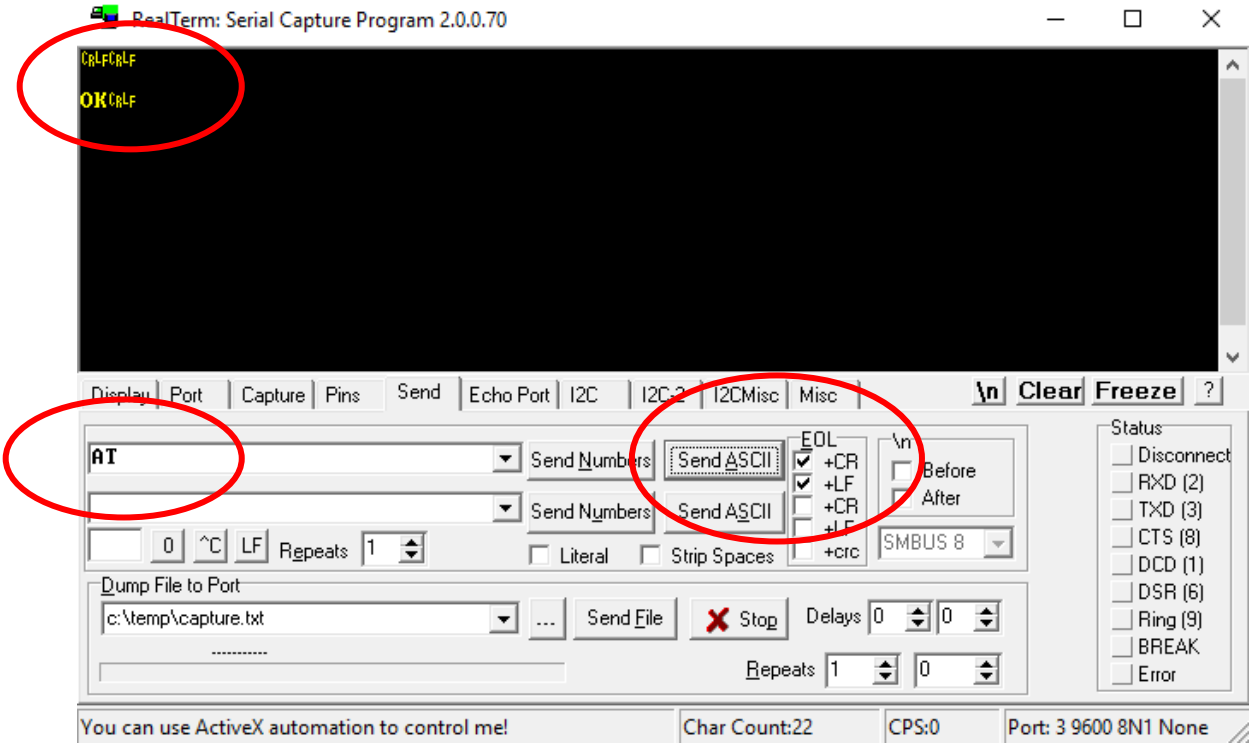


Fig. 15. Configuració de la comunicació entre l'USB-TTL i l'ESP. Font: pròpia.

Si la velocitat a la qual es realitza la comunicació és 115200 bauds/s, no cal realitzar el següent apartat, però si no, s'ha de configurar la velocitat de transmissió a 115200 bauds/s per tal de poder actualitzar l'ESP.

Per fer això, cal escriure el comandament "AT+UART=115200,8,1,0,0", tal com es pot veure a la Fig.16, i per assegurar-nos de que la comunicació es realitza de forma correcte, canviem la velocitat de transmissió a la pestanya *port*, i desactivem i activem el botó *open*, per tal de que s'efectuïn els canvis. Un cop efectuats els canvis, si enviem el comandament AT, ens hauria de tornar la resposta OK, a l'igual que abans.

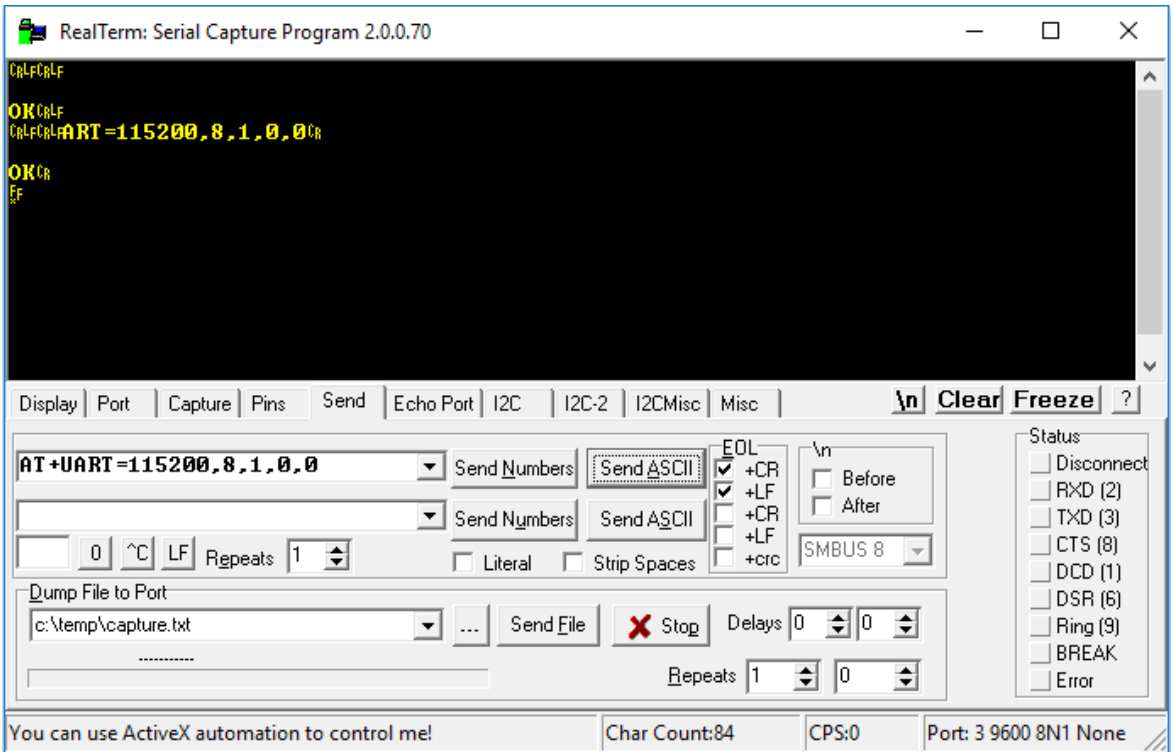


Fig. 16. Configuració de la comunicació entre l'USB-TTL i l'ESP. Font: pròpia.

Un cop efectuats els canvis, ens baixem la versió 0.9.5.2 del sistema operatiu de l'ESP, la qual es pot trobar a [1] i el programa per instal·lar aquesta versió anomenat *Flash tools*, que es pot trobar a [3].

Un cop descarregat aquest programa, obrim els *Flash tools* per al l'ESP8266, i es selecciona la versió 0.9.5.2 del sistema operatiu de l'ESP.

Posteriorment es configura la direcció de la memòria on s'ha de guardar (0x00000).

Seguidament, es configuren la resta de pestanyes de la pantalla, tal i com es pot veure a la Fig. 17, seleccionem el port COM on s'ha connectat el convertidor, USB-TTL i la velocitat de transmissió, la qual és 115200.

Tot i això, abans d'intentar actualitzar l'ESP, cal canviar la configuració dels pins, i **col·locar el pin GPIO 0 a GND (0 V)**. Un cop fet això, es pot apretar START, i esperar a que acabi l'actualització. (Es recomanable utilitzar una altra font d'alimentació a més de la que porta l'USB-TTL, ja que aquesta pot ser que no sigui suficient i pot donar problemes).

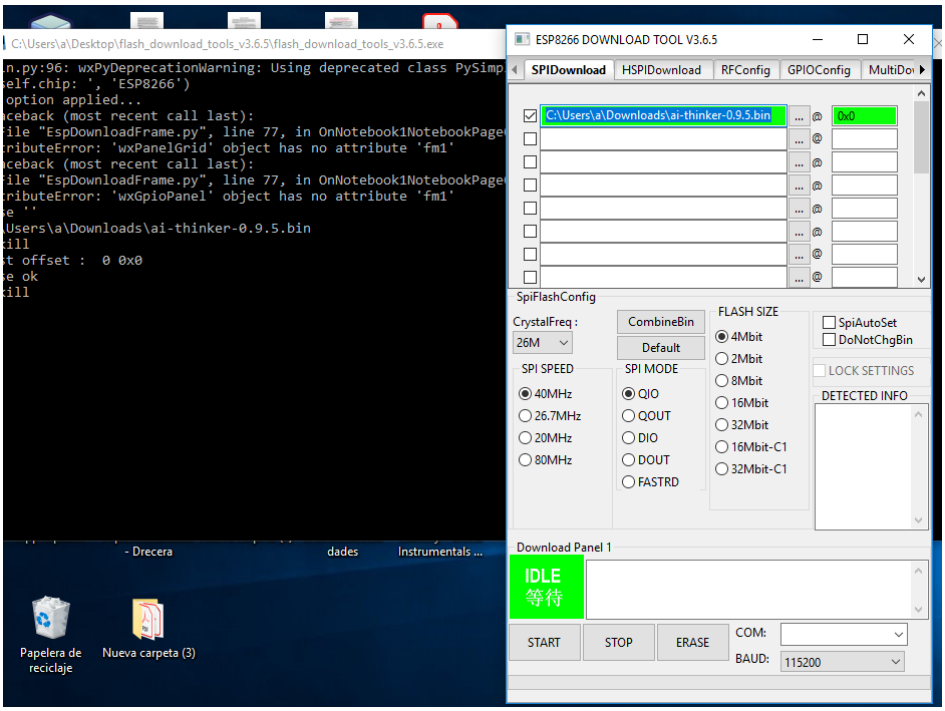


Fig. 17. Configuració de la pantalla del flash tools. Font: pròpia.

Si tot va bé, al cap d'una estona, l'*ESP* ja estarà actualitzat (Fig. 18).

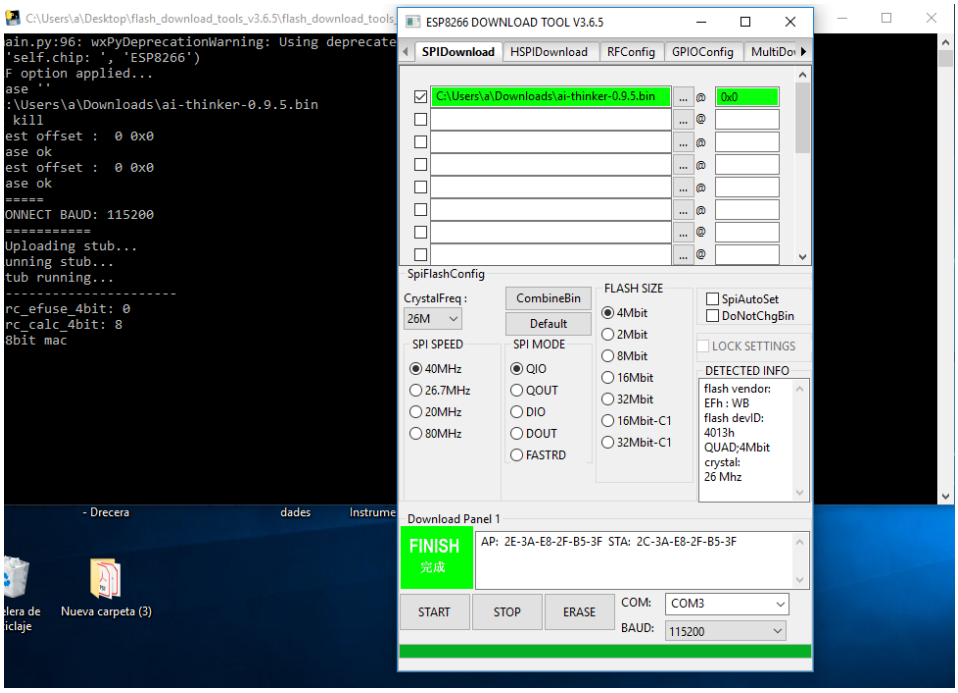


Fig. 18. Configuració de la pantalla del flash tools 2. Font: pròpia.

Un cop instal·lada aquesta versió, **tornem a connectar el pin GPIO 0 a 3,3 V**, i posteriorment, anem al RealTerm, per determinar si la instal·lació s'ha de realitzar correctament. Un cop allà, s'ha de configurar un altre cop les diferents opcions que dóna el RealTerm, i un altre cop s'ha de determinar si la velocitat de transmissió és de 115200, si és el cas, caldrà configurar la velocitat de transmissió a 9600 amb el comandament "AT+UART=9600,8,1,0,0".

Un cop actualitzat i configurat a 9600 bauds/s, l'ESP ja està preparat per realitzar les següents tasques.

3.3. Open18F4520

Per tal de facilitar la tasca de realitzar les connexions entre el microcontrolador i els altres elements que intervenen en el projecte, com per exemple el mòdul wifi, o la pantalla, s'utilitza la placa Open18F4520.

Tot i que el seu preu és d'aproximadament 50 €, s'ha decidit treballar amb ella, ja que el Departament d'Electrònica ja la tenia, i en el cas que no es volgués fer servir aquesta placa, es pot utilitzar una placa de prototipat, o "breadboard". En aquest cas, s'ha de tenir en compte que la realització de les connexions pot ser una mica més complexa, a més de que s'haurà de treballar en l'alimentació per així evitar possibles problemas.

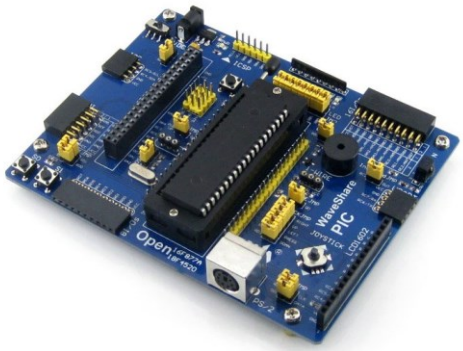


Fig. 19. Placa Open18f4520. Font: [16].

3.4. 2.2 Inch Touch LCD

Per la realització del projecte, també s'utilitza una pantalla LCD per mostrar les dades que es transmeten entre les dues plaques. Aquesta té un cost d'aproximadament 10 €, i la mateixa placa utilitzada ja està preparada per treballar amb aquest model de pantalla. El codi necessari per fer-la funcionar es pot trobar a la pàgina web del fabricant [4].

Fig. 20. 2.2 Inch LCD. Font: [4].

3.5. Sensor de temperatura

Per acabar, l'últim element amb el qual es treballa és un sensor digital de temperatura model DS18B20 fabricat per Maxim Integrated.

Tota la informació d'aquest es pot trobar a [5], on hi ha el "datasheet" d'aquest. Tot i això, en general, el que cal saber d'aquest és que té un rang de treball de entre -55ºC i 125ºC i una precisió de 0,0625ºC.

El voltatge que cal subministrar per fer funcionar el sensor ha d'estar entre 3,0 V i 5,5 V.

El consum d'aquest és d'aproximadament 1 mA durant el procés de mesura i 0,001 mA en "standby". A més, el protocol que utilitza per transmetre dades és el conegut com 1-Wire.

Fig. 21. DS18B20. Font: [17].

3.5.1. Protocol “1-Wire”

El protocol “1-Wire”, és un protocol inventat per l’empresa Dallas, que com el seu nom indica, només treballa amb un cable de dades (a més dels dos cables d’alimentació V+ i V0), i per transmetre els dos valors lògics dels quals es compon el codi binari, utilitza el següent sistema.

Per començar una transmissió de dades, el “Master”, o l’encarregat de donar ordres, en aquest cas el microcontrolador, baixa el bus a valor 0 volts durant entre 1 i 15 µs, i després en funció de si el valor a transmetre és zero o un, posa el bus a valor alt o el manté a zero fins a arribar a 60 µs (Fig.22). La lectura de dades es produeix aproximadament als 30 µs, les dades es transmeten en grups de 8 bits, i quant el dispositiu esclau vol contestar l’ordre que li ha donat el dispositiu “Master”, realitza el mateix procés, baixar la tensió uns 15 µs, i pujar o mantenir la tensió a zero en funció del valor lògic a transmetre.

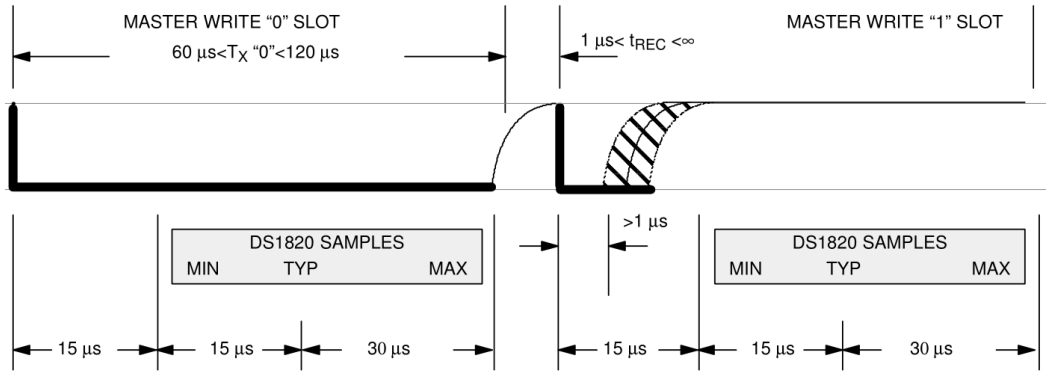


Fig. 22. Protocol 1-Wire. Font: [5].

4. MPLAB

A més del material comentat anteriorment, també s'utilitza el programa MPLAB, el qual es pot descarregar des de la mateixa pàgina de Microchip, i tal com s'ha dit a la introducció, aquest programa és gratuit [6].

La versió utilitzada de MPLAB és la X IDE v4.20, la qual incorpora el compilador XC8.

És important tenir en compte el compilador utilitzat, ja que la utilització del mateix codi amb diferents compiladors pot donar problemes, bàsicament perquè es pot donar el cas que algunes de les funcions o llibreries que es criden dins el programa no estiguin presents en un compilador diferent, i per tant, la implementació d'aquest programa no podria funcionar. Tot i això, si es volgués utilitzar un altre compilador, en general, fent una sèrie de petits retocs al programa fet amb el compilador XC8, es podria arribar a fer funcionar.

L'altra eina imprescindible per implementar el programa en un microcontrolador és un programador, el qual s'encarrega de gravar el programa al microcontrolador. En aquest cas, s'ha fet servir el programador/depurador MPLAB ICD 3 de la marca Microchip (Fig. 23).

Aquest programador es pot trobar a Amazon per aproximadament 200 €, el qual és força elevat, però donat que el Departament d'Electrònica ja el tenia, s'ha decidit treballar amb aquest programador.

Si es donés el cas de que es vol implementar aquest programa i no es té un programador, es pot optar per comprar el Pickit 3, el qual es pot trobar per aproximadament 20 €.

Fig. 23. ICD 3. Font: [18].

4.1. Inicialització de l'MPLAB

Per tal d'entendre millor les finestres de l'MPLAB, ja que més endavant es parlarà de funcions que incorpora aquest programa, i per tal d'evitar que un lector no habituat amb l'eina es perdi, a continuació es presenten les principals funcions o les principals eines que s'han fet servir durant el desenvolupament del programa de transmissió de dades entre el microcontrolador i una base de dades.

El primer que s'ha de fer és començar un nou projecte (Fig. 24) entrant a la pestanya "File", i posteriorment entrant a "New Project".

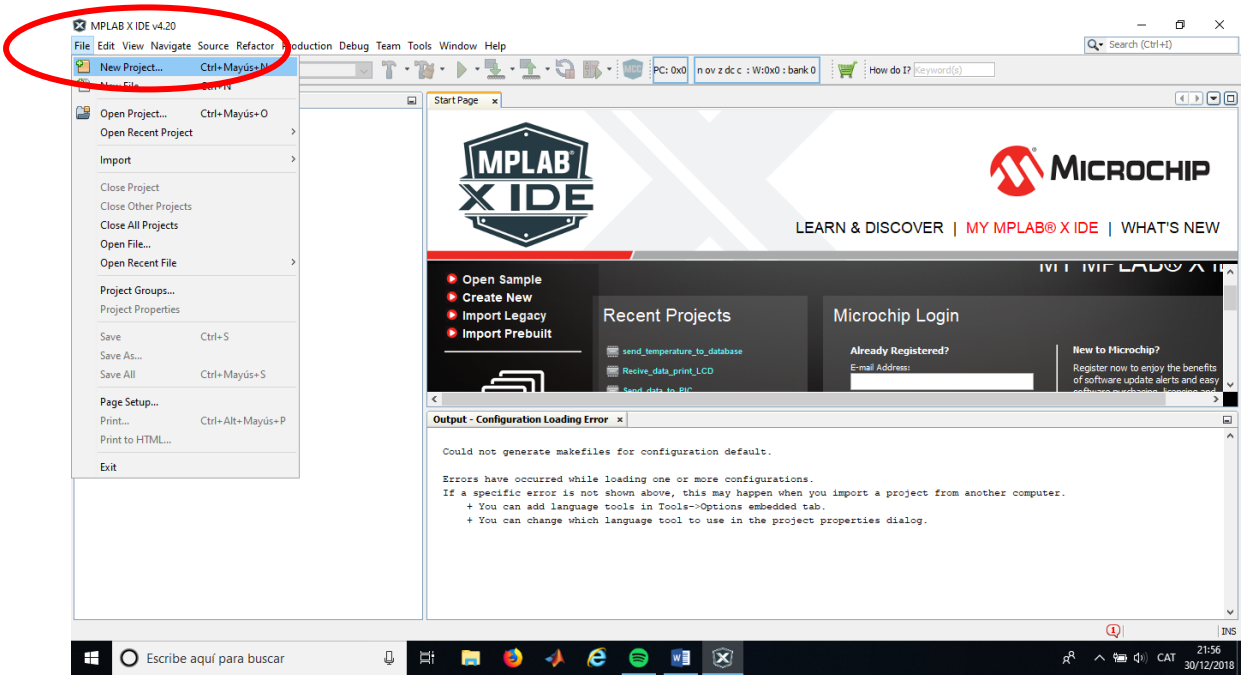


Fig. 24. MPLAB creació d'un nou projecte 1. Font: pròpia.

Posteriorment creem un projecte estàndard (Fig. 25) i cliquem "next". Després seleccionem el tipus de microcontrolador amb el qual treballarem, que en aquest cas és el PIC18F4520, i tornem a clicar "next" (Fig. 26).

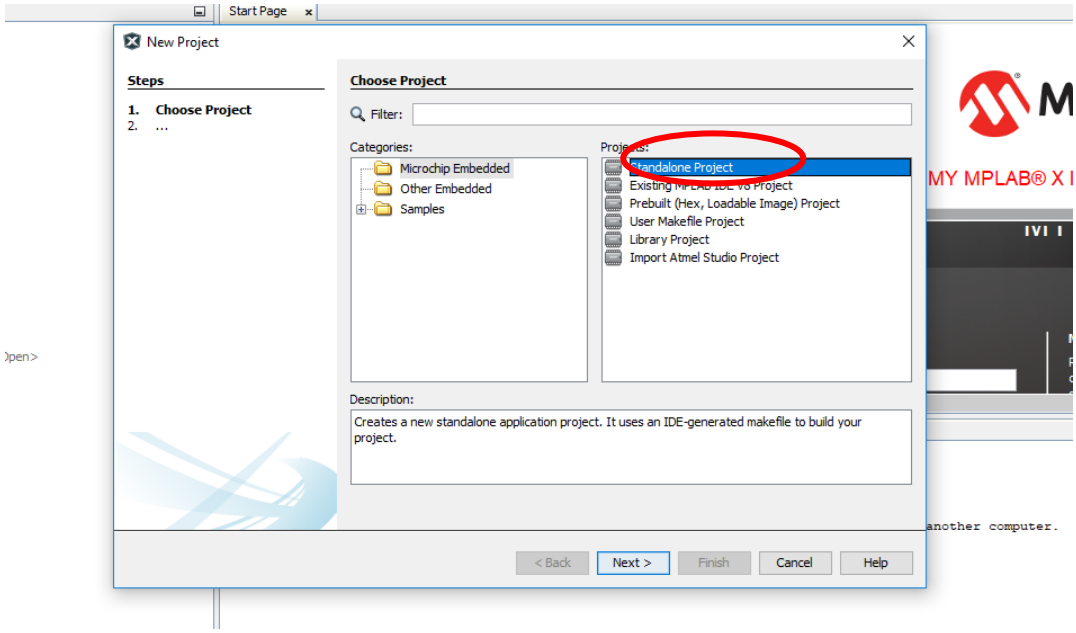


Fig. 25. MPLAB creació d'un nou projecte 2. Font: pròpia.

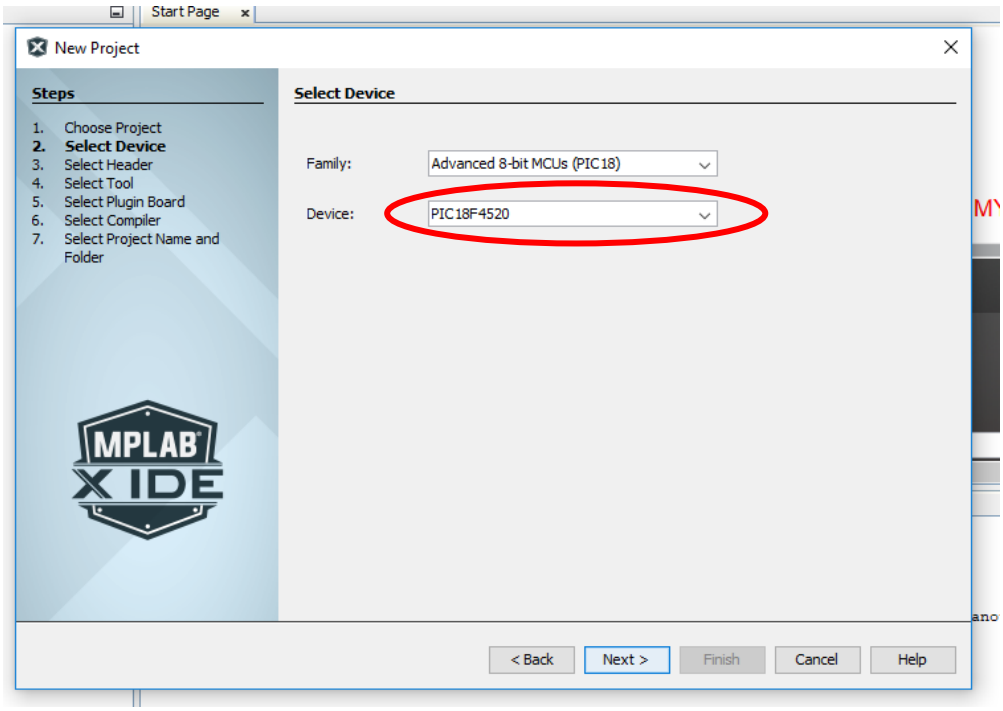


Fig. 26. MPLAB creació d'un nou projecte 3. Font: pròpia.

Posteriorment, seleccionem el programador que utilitzarem que en aquest cas és l'ICD 3 (Fig. 27), i el compilador que un altre cop, com ja s'ha dit, és l'XC8 (Fig. 28).

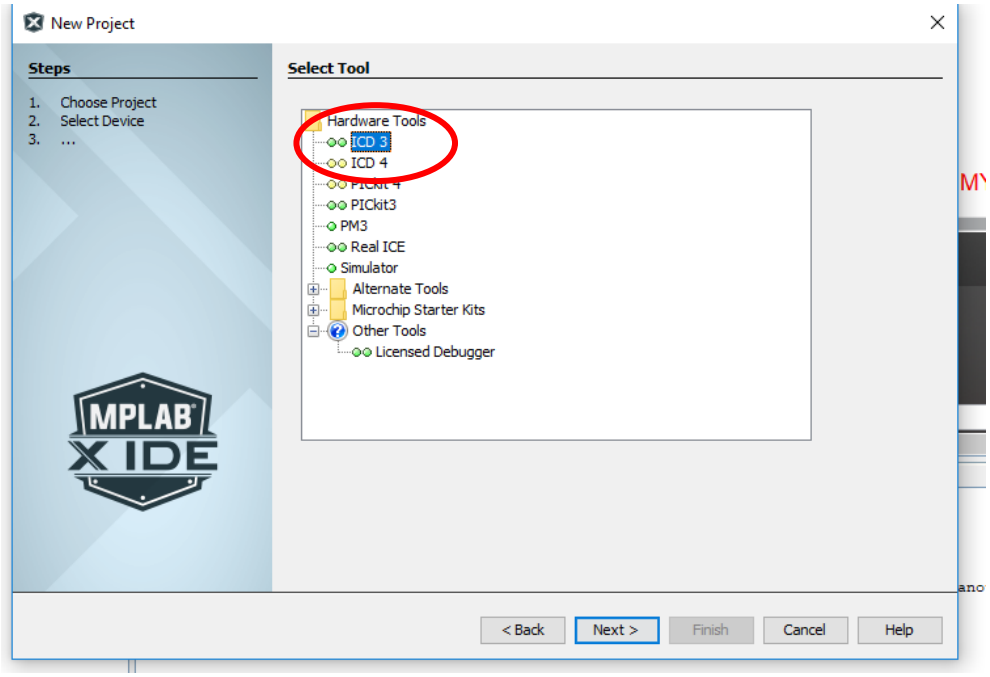


Fig. 27. MPLAB creació d'un nou projecte 4. Font: pròpia.

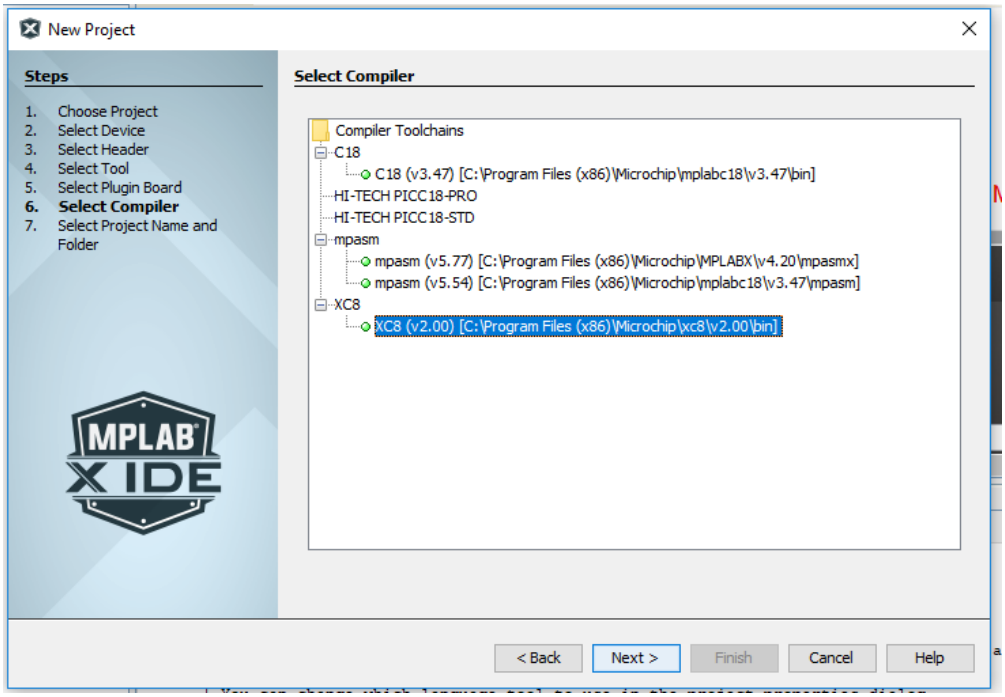


Fig. 28. MPLAB creació d'un nou projecte 5. Font: pròpia.

Per acabar, se li dona un nom al projecte i es selecciona la ubicació d'aquest (Fig. 29), i ja es pot clicar "Finish" per començar a treballar en el programa a realitzar.

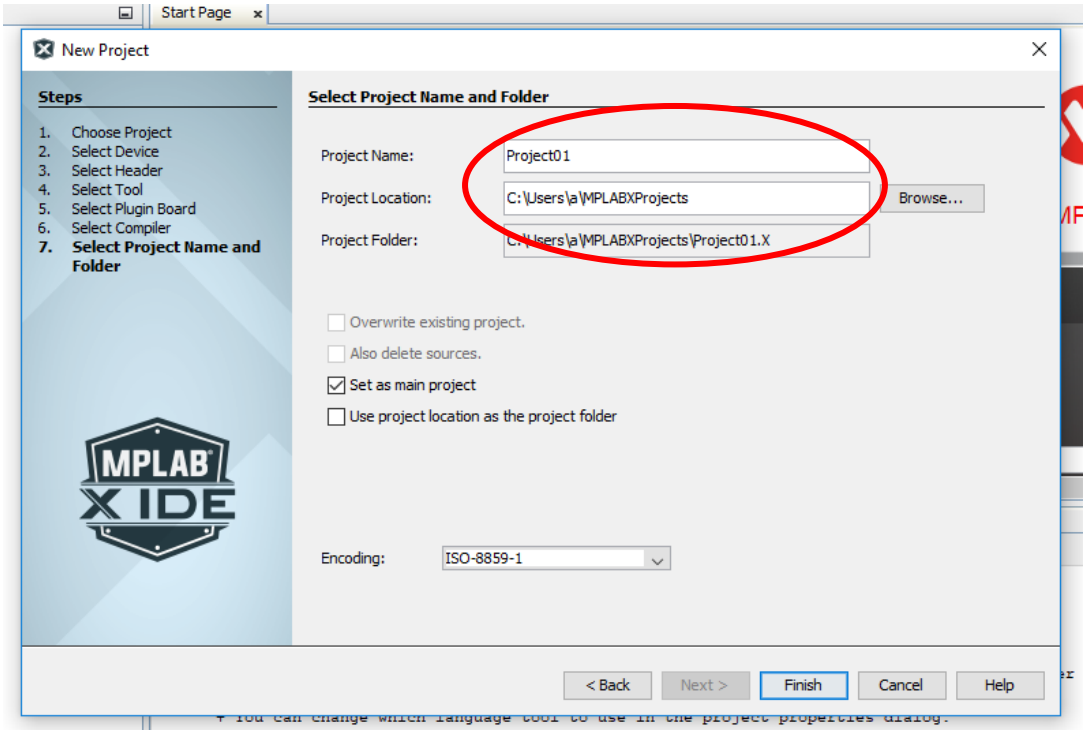


Fig. 29. MPLAB creació d'un nou projecte 6. Font: pròpia.

4.2. Funcions principals de l'MPLAB

La finestra principal de l'MPLAB es pot dividir en quatre grans parts, tot i que es pot configurar i afegir o retirar finestres, però, per defecte, ve configurada de la següent forma, i en principi, tal i com està per defecte, es pot treballar perfectament (Fig. 30).

En aquest es poden diferenciar 4 finestres:

1. A la primera finestra, tenim tots els arxius del programa, tant les llibreries com els arxius amb codi de programa.

2. Si seleccionem un arxiu concret de la primera finestra, podrem llegir el codi que hi ha en aquest a la segona finestra.

3. A la tercera finestra tenim informació de la memòria que ocupa el programa creat i altres informacions importants com el número de "break points" utilitzats (la funció d'aquests s'explicarà més endavant).

4. I per acabar, la quarta finestra es pot configurar perquè mostri informació rellevant, com el valor d'alguna variable en algun moment donat (si s'està treballant en mode "debugger"), o els errors que apareixen, en el cas que n'hi hagués algun, durant el moment de compilar el programa.

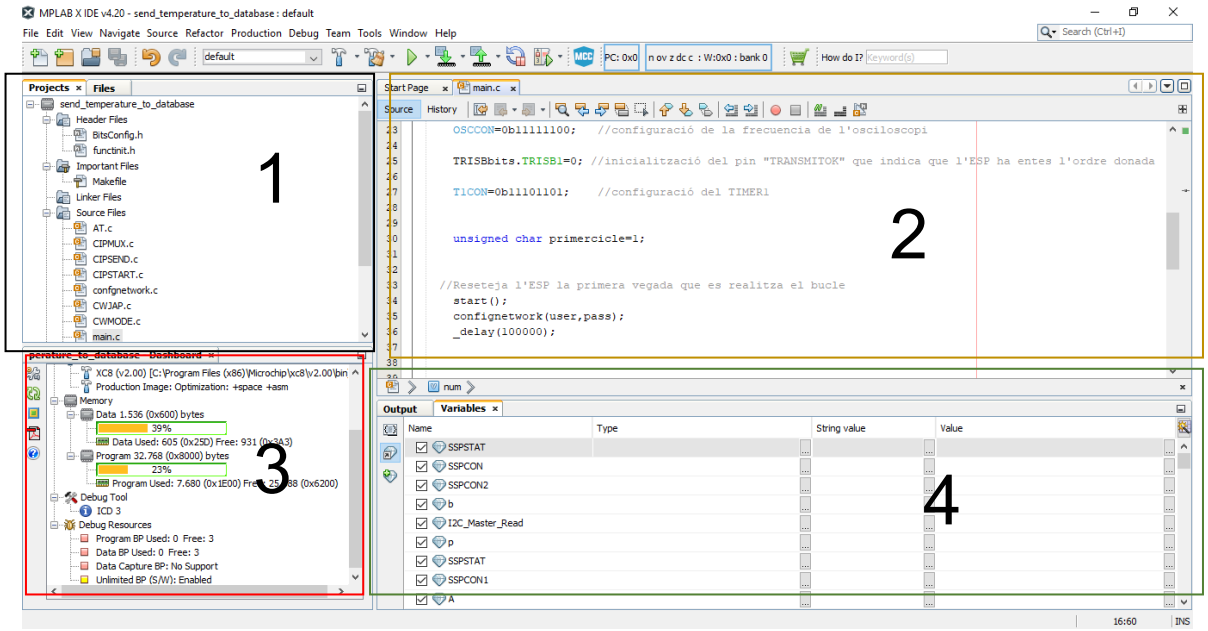


Fig. 30. Finestra principal de l'MPLAB 1. Font: pròpia.

A més, a la part superior de la finestra de l'MPLAB, tenim vàries icones força importants, per tal de fer funcionar el programa (Fig. 31).

1. La primera és la funció de compilar i netejar el programa, la qual revisa i tradueix el programa a un codi que el microcontrolador pot entendre.

2. La segona programa el microcontrolador amb el programa compilat anteriorment.

3. La tercera programa e inicia el microcontrolador en mode "debugger", i si s'ha situat algun "break point" en el programa, aquest s'atura en arribar en aquest punt, fet que permet revisar l'estat de les variables amb les quals s'està treballant durant el procés de creació del programa, i així determinar si està succeint allò que nosaltres volem que passi. Si volguessim introduir un "break point", només s'ha de donar un clic al número que apareix al costat de la línia de programa, on volguem que el programa s'aturi, tal com es mostra a la Fig. 32.

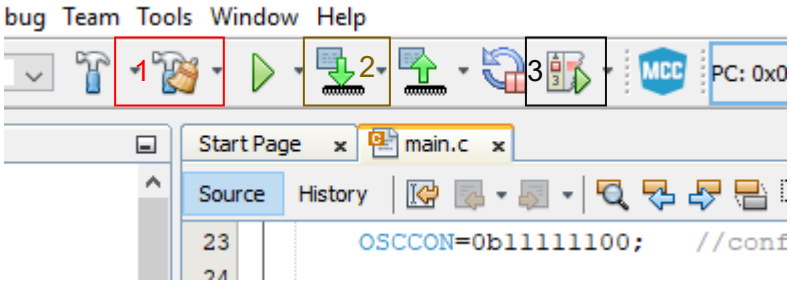


Fig. 31. Finestra principal de l'MPLAB 2. Font: pròpia.

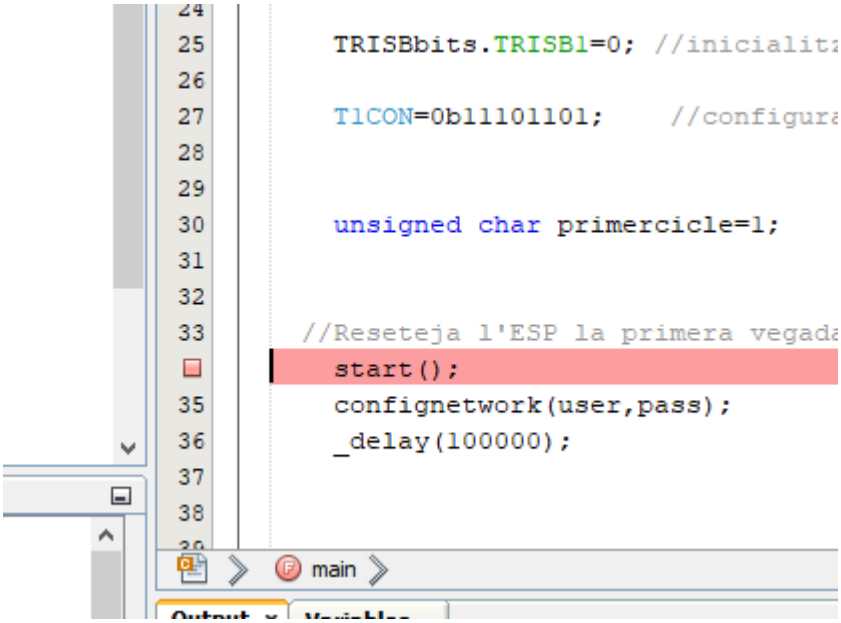


Fig. 32. Finestra principal de l'MPLAB 3. Font: pròpia.

5. Transmissió de dades de un µC a una base de dades

El primer objectiu del projecte, tal com s'ha dit a la introducció, és el de ser capaç de recollir dades mitjançant un sensor, i transmetre-les a un magatzem de dades mitjançant un microcontrolador i un mòdul wifi de baix cost.

Els arxius de codi es presenten a continuació:

5.1. Llibreries

Per tal de poder treballar de forma més còmode, a més de facilitar la programació, el que se sol fer, és escriure totes aquelles definicions o crides a diferents programes que es realitzin de forma continuada durant el programa, en una llibreria, i així, fer més simple el programa escrit.

Cal destacar també que aquestes funcions s'han d'afegir a la carpeta "Header Files" del menú de la finestra on teníem tots els arxius que s'ha comentat anteriorment (finestra 1 de la Fig. 30). En aquest cas, s'ha introduït dues llibreries externes, una per configurar els bits interns del microcontrolador, i l'altre per cridar totes les funcions que utilitza l'arxiu main.c.

A l'arxiu que configura els bits interns del microcontrolador (ANEX 1), cal destacar que s'ha configurat el microcontrolador perquè treballi amb l'oscil·lador intern "#pragma config OSC = INTIO7", i per tant, més endavant en el programa s'haurà de configurar la freqüència d'oscil·lació d'aquest.

També cal destacar que la resta de bits s'ha desactivat i per tant, opcions com el "watch dog timer", que és un comptador que evita que el programa es quedi clavat en un punt, i en el cas de que això passes reinicia el microcontrolador, està desactivat, a més de la resta de elements de control del bon funcionament del programa com el "burn out reset", que reinicia el microcontrolador quan el voltatge és inferior a un valor determinat, o les posicions de memòria prohibides, ja que en aquest cas, complicarien el programa, i per la funció que se li vol donar, no és necessari tenir aquestes precaucions, o en el cas que es tinguessin no aportarien gaire res al programa.

L'altra llibreria externa que s'ha introduït (Annex 2), cridar a les funcions que l'arxiu main necessita, ja que quan es programa en C, si es vol utilitzar una funció, cal haver-la cridat anteriorment.

5.2. Funcions

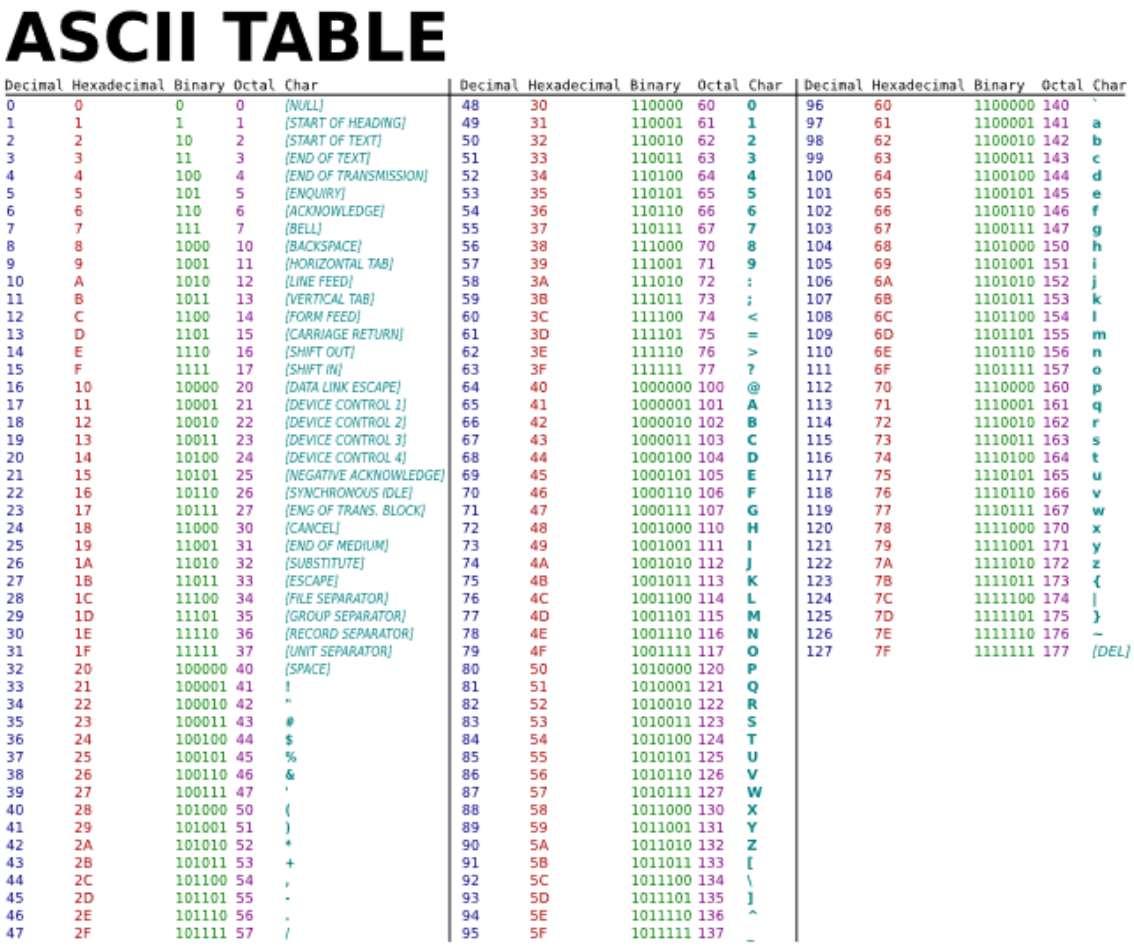
5.2.1. Funció main.c

La funció principal del programa és l'arxiu main.c, en el qual comença el programa, i va cridant a la resta de funcions. Aquestes es situen a la carpeta Source Files de la finestra 1 de la (Fig. 30).

A l'annex 3 es pot veure el codi d'aquesta funció. Aquesta comença cridant a les següents tres llibreries, les dos comentades en l'apartat anterior (entre cometes, ja que són llibreries externes creades per l'usuari), i la llibreria xc.h (entre signes de més gran que i menor que, ja que és una llibreria interna del compilador seleccionat), la qual incorpora funcions com la funció "_delay(X)", la qual realitza la funció d'esperar, durant un nombre de cicles de màquina determinats per el valor X, de dins els parèntesis.

S'ha de tenir en compte que un cicle de màquina, són quatre cicles de l'oscilador, i per tant, si la frecuencia de l'oscil·lador és de 8 MHz o 8.000.000 Hz, si triem el valor X=2.000.000, la funció s'esperarà un segon.

A continuació s'inicialitzen una sèrie de variables "globals" amb les que poden treballar tots els arxius o rutines que té el programa.
La primera és la variable *valor_temperatura*[9], la qual és una llista de nou posicions formada per *chars* (variable de 8 bits), el primer dels quals es comptabilitza com a part del número, de forma que es poden definir números del 0 al 255. En el cas contrari, si el *char* no és "unsigned", es poden definir números des de el -127 al 127.
Tot i això, donat que es treballa en ASCII, per tant, amb lletres, el valor de cada variable es comptabilitza de forma diferent (seguint el codi ASCII [Fig.33]), (igualment, s'han de definir així, per tal de poder treballar amb elles després).
La següent variable amb la qual es treballa és la variable *f*, que és una llista amb 100 posicions la qual es fa servir com a buffer per bolcar les dades que es van rebent del port UART.
La variable *user* és el nom de la connexió a la que es connectarà l'ESP, i la variable *pass* és la contrasenya de la connexió a la que ens volem conectar.
La següent variable és la direcció de la base de dades (link), i per acabar, la variable *num* s'utilitza per diferenciar diferents microcontroladors, en el cas que més d'un microcontrolador es connectessin a l'hora.



ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Fig. 33. Taula conversió binari ASCII. Font: [19].

A continuació entrem en el programa en sí.

El que es fa primer és configurar l'oscil·lador a 8 MHz de la forma que indica el manual del PIC18f4520 (Fig.34). Si mirem els bits del registre *OSCCON*, veurem que primer hi ha un 0b, lo cual indica que el valor està en codi binari, posteriorment tenim un 1 que, segons la taula del manual del PIC18f4520 (Fig. 35), indica que si aquest bit es configura com un 1, quan el microcontrolador se li indica que entri en el mode "sleep mode", el microcontrolador entra en el mode idle, el qual és un mode de baix consum que te el microcontrolador, i en canvi, si es configures amb un zero, el microcontrolador entraria en mode sleep, que és un altre mode de baix consum, i els dos es diferencien per la forma de despertar-se i el consum en aquests dos modes. En qualsevol cas, donat que en aquest programa no s'entra en el "sleep mode", no és molt important el valor d'aquest bit, però els següents tres bits si que són importants, ja que indiquen la velocitat a la qual ha de vibrar l'oscil·lador del microcontrolador, i donat que volem que treballi a la màxima velocitat possible, el

configurem a 8 MHz.
Els següents dos bits no són molt importats en aquest projecte, ja que són només de lectura, i en aquest cas, donat que l'oscil·lador funciona bé, no es revisen en cap cas.
Els dos últims bits però, si que són importants, ja que amb aquests s'indica al microcontrolador que s'ha de treballar amb l'oscil·lador principal que aquest porta incorporat, (el microcontrolador porta incorporat més d'un oscil·lador internament).
A continuació es configura el pin número 33 (Fig. 2) del microcontrolador com a sortida, de forma que internament es pot generar 3,3 V o 0 V per encendre o apagar un LED que porta incorporat la placa OpenPIC18F4520, i així determinar que el programa treballa correctament.

Posteriorment, es configura el registre *T1CON* que configura el "Timer 1" del microcontrolador (un altre oscil·lador diferent del principal), i aquest, es fa servir com a rellotge per determinar en certs moments del programa quant temps ha passat des d'un moment concret, i així, es pot introduir una condició que digui que si ha passat més d'un temps determinat, i la funció no ha respòs el que tocava, es pot realitzar alguna rutina que corregeixi aquest possible error.
Tal com s'han configurat els bits en aquest cas, es pot veure a la Fig. 35 que un altre cop, el 0b del principi indica que el valor de a continuació s'ha de considerar un valor binari de 8 bits, el primer 1 configura el timer perquè treballi amb una variable de 16 bits, el segon valor no és rellevant en el nostre cas, ja que és un valor de només lectura, i per tant, no és pot configurar.
Els següents dos bits configuren el "prescaler", lo qual fa que cada X cicles del rellotge es comptabilitzin com un cicle, i per tant sumin un valor al rellotge (en aquest cas s'ha configurat que cada quatre cops del rellotge, es comptabilitzin com un +1 al valor del rellotge).
El següent bit activa el funcionament del "timer", el valor del tercer per la cua no te rellevància, ja que el rellotge es configura perquè treballi amb una oscil·lador intern, el penúltim bit indica que l'oscil·lador seleccionat és l'intern, i l'últim inicia el "timer".

A continuació, el programa main.c realitza una sèrie de rutines que es comenten en els següents apartats.

PIC18F2420/2520/4420/4520

REGISTER 2-2: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-1	R/W-0	R/W-0	R(1)	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN**: Idle Enable bit
1 = Device enters an Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>**: Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz(3)
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)(2)

bit 3 **OSTS**: Oscillator Start-up Timer Time-out Status bit(1)
1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running
0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready

bit 2 **IOFS**: INTOSC Frequency Stable bit
1 = INTOSC frequency is stable
0 = INTOSC frequency is not stable

bit 1-0 **SCS<1:0>**: System Clock Select bits
1x = Internal oscillator block
01 = Secondary (Timer1) oscillator
00 = Primary oscillator

Note 1: Reset state depends on state of the IESO Configuration bit.
2: Source selected by the INTSRC bit (OSCTUNE<7>), see text.
3: Default output frequency of INTOSC on Reset.

Fig. 34. Configuració dels bits del registre OSCCON. Font: [11].

REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **RD16**: 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN**: Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS<1:0>**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Fig. 35. Configuració dels bits del registre T1CON. Font: [11].

5.2.2. Funció start.c

La funció start.c (Annex 4) inicialitza el protocol UART del microcontrolador i s'assegura de que la comunicació entre el microcontrolador i el mòdul wifi és correcte.

El codi comença cridant la llibreria *xc.h,* la qual inclou la funció "delays" que s'utilitza per implementar període d'espera per tal de permetre treballar a l'ESP.

Posteriorment es defineixen les sortides B2 i B3 amb el nom de “reseting” i “reading”. Això el que farà és que en el moment d’igualar aquestes variables a 1, s’encendrà un LED de la placa (el B2 o el B3), i quan s’iguala a zero, aquest s’apaga.
Així,mitjançant l’encesa i apagada de LEDs,es pot veure que el programa està realitzant les tasques que toca.
Posteriorment s’inicialitzen les funcions que utilitzarà la rutina (és una tasca que el compilador utilitzat t’obliga a fer en cada arxiu de codi del programa).

Posteriorment, s’inicia la rutina d’inicialitzar l’ESP i la rutina de inicialitzar la configuració de la comunicació UART. El primer que fa la rutina *start* és inicialitzar el bit B3 com a sortida, de forma que internament es pot posar aquests bits a 3,3 V o 0 V (el cas contrari seria configurar aquets bits com a entrada).

Després, el microcontrolador s’espera cinc segons a que el mòdul wifi s’encengui.

Seguidament inicia la rutina *“startUART()”,* que és la següent funció dins l’arxiu *start.c*, la qual es dedica a inicialitzar el perifèric UART del microcontrolador en les condicions adequades.

Primer es configuren els dos bits amb els que treballa el perifèric UART del microcontrolador (bits C6 i C7) com a entrada (el manual del microcontrolador indica que per treballar amb aquest perifèric s’han de configurar com a entrada els dos, i el microcontrolador ja els reconfigurarà com a sortida quan tingui d’enviar dades).

Posteriorment es configuren els registres SPBRGH i SPBRG amb els valors que indica el manual per tal que les dades s’enviïn i es llegeixin a 9.600 bauds/s. Segons la taula de la (Fig. 36) el valor de l’SPBRG ha de ser 12, i per tant en binari, el valor al qual s’ha de configurar és 00001100. El valor de l’SPBRGH es queda a zero, ja que el registre SPBRG és un registre de 16 bits format pels 8 bits superiors (SPBRGH) i els 8 bits inferiors (SPBRG), i per definir-lo amb el valor 12, amb el 8 bits inferiors ja fem. Si suposem el cas en el que necessitem configurar el SPBRG a un valor superior a 255, llavors sí que seria necessari el registre SPBRGH, però en aquest cas no ho és.

Un cop configurats els registres SPBRG, s’han de configurar els registres TXSTA i el RCSTA, per tal de que la comunicació UART es produeixi de forma adequada.

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)	Actual Rate (K)	% Error	SPBRG Value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

Fig. 36. Configuració dels registres SPBRGH i SPBRG. Font: [11].

El registre TXSTA (Fig. 37) es configura de la següent forma. El bit 7 a zero, ja que el mode de comunicació és asíncron, i per tant el valor d'aquest bit no importa. El bit 6 es posa a zero, ja que el mode de transmissió és de 8 bits, el bit 5 es posa a 1 i així s'habilita la transmissió de dades, el bit 4 es configura a zero per treballar en mode asíncron, el bit 3 es configura a zero, ja que no interessa que es trenqui la comunicació entre el PIC i l'ESP al final de cada comunicació. El bit 2 es configura a 0, ja que per treballar a 9600 bauds/s és necessari que estigui configurat així, el bit 1 no interessa ara perqué és només de lectura, i el bit 0 tampoc interessa, ja que es treballa amb mode de comunicació de 8 bits i aquest bit configura el novè bit.

Posteriorment es configura el registre RCSTA. El bit 7 del registre RCSTA configura com a ports els pins TX i RX (bits C6 i C7) de comunicació en sèrie, el bit 6 configura la comunicació amb paraules de 8 bits, el bit 5 no importa, ja que el mode de comunicació és asíncron, el bit 4 habilita la recepció de dades i per tant, es configura a valor positiu.

El bit 3 no importa, ja que té a veure amb la configuració de la comunicació de 9 bits per paraula, i els següents tres bits tampoc importen, ja que són només de lectura.

El bit 5 i el 4 es configuren a zero, ja que no es treballa amb valors invertits, el valor 3,3 V és l'1 lògic i el 0 V és el 0 lògic.

El bit 3 no importa gaire, ja que tant el registre SPBRGH i el registre SPBRG ja s'han configurat anteriorment. El bit 1 es configura a 1 i així, el microcontrolador llegeix contínuament si s'està rebent alguna dada. El bit 0 es configura a 0 per deshabilitar el càlcul automàtic del "baudrate" (ja que ja s'ha configurat anteriorment).

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data
Can be address/data bit or a parity bit.

Fig. 37. Configuració dels registre TXSTA. Font: [11].

REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPEN**: Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9**: 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-Bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-Bit (RX9 = 0):
Don't care.

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be cleared by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit, CREN)
0 = No overrun error

bit 0 **RX9D**: 9th Bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

Fig. 38. Configuració dels registre RXSTA. Font: [11].

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **ABDOVF:** Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL:** Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **RXDTP:** Data/Receive Polarity Select bit
Asynchronous mode:
1 = Receive data (RX) is inverted (active-low)
0 = Receive data (RX) is not inverted (active-high)
Synchronous mode:
1 = Data (DT) is inverted (active-low)
0 = Data (DT) is not inverted (active-high)

bit 4 **TXCKP:** Clock and Data Polarity Select bit
Asynchronous mode:
1 = Idle state for transmit (TX) is a low level
0 = Idle state for transmit (TX) is a high level
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

Fig. 39. Configuració dels registre BAUDCON. Font: [11].

5.2.2.1. Comunicació UART

Donat que la comunicació entre el microcontrolador i el mòdul Wifi es produeix mitjançant el protocol sèrie asíncron (UART), és important tenir en compte quin format segueix aquest.

El primer que cal tenir en compte és que a l'hora de connectar els pins del microcontrolador amb els del mòdul s'ha de connectar el pin RX del mòdul amb el TX del microcontrolador, ja que el TX transmet dades i el RX rep les dades, i el TX del mòdul s'ha de connectar amb el RX del microcontrolador pel mateix motiu.

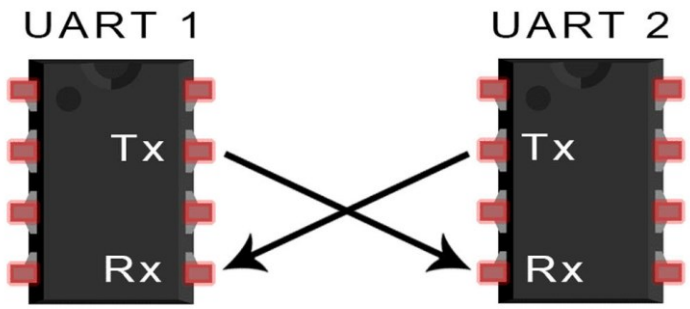


Fig. 40. Connexió dels pins RX i TX. Font: [20].

Posteriorment s'ha de saber que la comunicació es produeix amb 10 bits, 8 de dades un d'inici i un d'aturada.

Els cables, en estat de repòs es troben a valor lògic positiu (3,3 V), i la comunicació comença quan l'element que vol transmetre la dada baixa el bus a valor lògic zero durant un temps concret (depenent del "baudrate" seleccionat). Posteriorment s'envien els 8 bits que formen la dada a enviar. Un cop enviats els 8 bits, el microcontrolador que envia la dada posa el bus a valor alt (bit de stop), i posteriorment el torna a baixar per iniciar la transmissió del següent byte, i així fins enviar la paraula sencera que s'ha d'enviar.

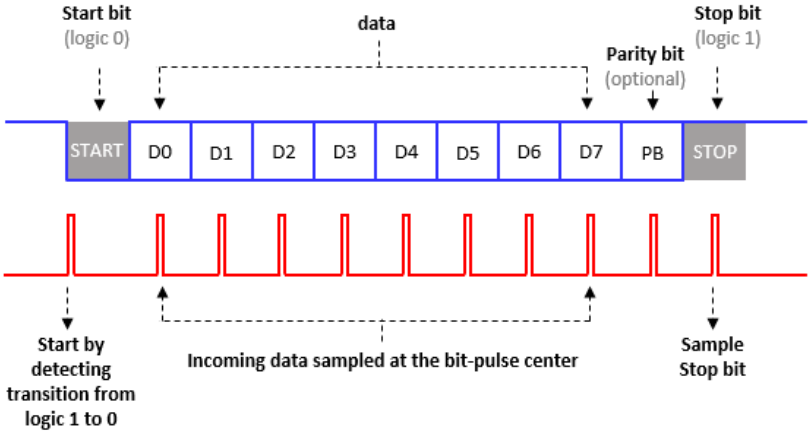


Fig. 41. Protocol UART. Font: [21].

Un cop configurada la comunicació, el microcontrolador realitza un altra aturada, i posteriorment envia el comandament AT al mòdul wifi (al següent apartat es parla dels comandaments AT i de la rutina "sendstring"), després llegeix la resposta del mòdul wifi, i posteriorment torna a enviar el mateix comandament i torna a llegir la resposta.

Aquest procés es realitza dues vegades, ja que durant la implementació del projecte, en alguns casos, la resposta que donava el mòdul wifi no era l'esperada, i realitzant aquest procés d'enviar la mateixa dada dues vegades, s'evitaven problemes.

Després es porta el bit CREN (bit 4 del RCSTA [Fig.38]) a zero i posteriorment a un, per tal de que si s'ha produït algun problema durant la comunicació en aquestes dues primeres transicions (ja s'ha dit que a vegades la primera transmissió donava problemes), netejar els possibles "flags" o "warnings" que dona el microcontrolador i així poder començar a comunicar dades sense cap problema.

Per acabar amb la rutina start, s'ha de dir que a priori pot semblar una mica caòtica, ja que incorpora vàries rutines de "delay" i l'enviament del mateix missatge vàries vegades, però s'ha de dir que s'ha arribat a ella mitjançant la prova i error, i tal i com està ara, funciona correctament (s'ha determinat el valor dels "delays" a prova i error, no els comandaments ni els valors d'aquests, ja que las configuracions vénen fixats pel manual del microcontrolador).

I l'últim que fa la rutina és apagar el LED que indica que s'està dins de la rutina d'inici (led B3).

5.2.3. Comandaments AT

El mòdul wifi, tal i com el seu manual indica [7], entén una sèrie d'ordres anomenades comandaments AT. A la taula mostrada a la Fig. 42 es poden trobar alguns d'aquests (els utilitzats durant la realització d'aquest projecte).

AT	No té una funció concreta, s'utilitza per determinar el correcte funcionament de l'ESP, a l'enviar aquest comandament, l'ESP contesta OK.
AT+RST	El comandament reinicia l'ESP i posteriorment es connecta a la última xarxa utilitzada.
AT+CWJAP=<ssid>,<pass>	El mòdul wifi es connecta a la xarxa indicada.
AT+CIFSR	El mòdul contesta l'adreça IP del de la connexió a la que està connectada o de la connexió de la que és punt d'accés.
AT+CWSAP=<ssid>,<pass>,<ch>,<encrypt>	Configura la connexió de l'ESP en el cas que treballi com a punt d'accés.
AT+CWMODE=X	Si X=1, el mòdul treballa com a client d'una connexió, si X=2, el mòdul treballa com a punt d'accés, i si X=3 treballa com els dos a la vegada.
AT+CIPMUX=X	Si X=0, el mòdul es connecta a només una base de dades, si X=1, el mòdul pot connectar-se a més d'un link (fins a 5).
(Si CIPMUX=0) AT+CIPSTART=<type>,<adress>,<port>	Aquest comandament es connecta a una base de dades mitjançant la selecció del protocol de comunicació (normalment TCP), l'adreça de la base de dades i el port (que normalment és 80).
(Si CIPMUX=1) AT+CIPSTART=<id>,<type>,<adress>,<port>	En aquest cas, els valors són els mateixos que els anteriors, a més de que s'ha de seleccionar un valor d'entre 0-4 com a id.

(SI CIPMUX=0) AT+CIPSEND=<len>	Aquest comandament inicia la transmissió de X=len caràcters a la base de dades.
(SI CIPMUX=1) AT+CIPSEND=<id>,<len>	Aquest comandament fa el mateix que l'anterior però també se li ha d'indicar el valor seleccionat per la base de dades concreta a la que volem enviar la dada (ja que en aquest cas ens hem connectat a més d'una base de dades).
AT+UART=<baud>,8,1,0,0	Aquest comandament, tal i com s'ha vist a l'apartat d'actualitzar l'ESP, configura la velocitat de comunicació del protocol UART.

Fig. 42. Comandaments AT. Font: pròpia.

*Tots els missatges han d'acabar amb \r\n, ja que és la forma que té el microcontrolador d'entendre que el missatge s'ha acabat.

5.2.4. Sendstring.c i Recive.c

Els dos comandaments que s'utilitzen per transmetre dades entre el microcontrolador i el mòdul wifi són la rutina "sendstring" i la rutina "recive".

El primer que es comenta és la rutina "sendstring" (Annex 5), que comença, a l'igual que tots els arxius, cridant a la llibreria *xc.h*, que inclou totes les funcions i dades del compilador XC8. Posteriorment, s'inicialitza el comptador i la variable *c*. Després s'agafa la comanda que ha d'enviar el microcontrolador a l'ESP, i es va enviant lletra per lletra.

Per enviar una lletra, cal escriure-la al registre TXREG, i esperar a que el "flag" passi a valor zero i així confirmar que la dada ja s'ha enviat. Per acabar, se suma un al comptador i, i s'envia la següent lletra, fins que no queden més lletres a l'ordre, o el que és el mateix, la lletra que llegeix el microcontrolador a l'intentar llegir l'ordre a enviar és '\0'.

La rutina "recive" funciona de manera semblant a la rutina "sendstring".

Aquesta es pot dividir en dues rutines, tot i que abans d'això, igual que en les rutines anteriors, s'inicia la llibreria *xc*, i es defineix LED, per indicar el bon funcionament del programa. Posteriorment s'inicia un "buffer" on bolcar tota la informació que es va rebent (a la llista f), i una sèrie de comptadors, a més d'una variable anomenada *IN*, utilitzada per

determinar si s'ha de sortir del programa, o encara s'està rebent informació i la inicialització del timer1 que es dedica a comptar el temps que ha passat des de la inicialització d'aquest.

El funcionament de la rutina es pot dividir en dues parts, com s'ha dit, una d'espera, i l'altre de lectura del registre RCREG, posició de memòria (registre) en la que es guarden els caràcters que es van rebent.

El funcionament de la rutina d'espera és simple, l'únic que fa és llegir contínuament el registre timer1, i si han passat més de 30000 cicles, acaba el programa (ja que s'ha determinat que aquest temps és suficient per rebre un caràcter, i si fa més d'aquest temps que no s'ha rebut cap caràcter, es pot determinar que la comunicació s'ha acabat).

La rutina de lectura del registre RCREG també és força simple. El primer que fa és llegir el "flag" RCIF, que s'activa quan s'ha rebut un caràcter, i si s'ha activat, comença la rutina de lectura llegint el registre RCREG. Posteriorment es guarda al buffer *f* la lletra rebuda seguida del caràcter \0, que serà sobreescrit per la pròxima lletra rebuda en el cas que es rebi alguna lletra més o quedarà com a últim caràcter del "string".

Per acabar se suma u al comptador i, es reinicia el timer1 i es passa a la rutina d'espera fins que arribi el següent caràcter.

5.2.5. Confignetwork.c

La funció "configNetwork", com el seu no indica, es dedica a configurar la connexió a la que s'ha de connectar l'ESP (Annex 7).

Per tal de realitzar aquesta connexió, requereix quatre passos. El primer consisteix en assegurar-se de que el mòdul wifi està ben connectat al microcontrolador mitjançant la funció AT (Annex 8) que envia el comandament AT i llegint la resposta d'aquest. Si la resposta és OK, això implica que la comunicació s'està realitzant de forma correcta, i per tant la connexió entre els dos elements és correcte.

El següent pas és configurar el mode de treball de l'ESP (Annex 8), donat que treballarà com a client d'una xarxa, es configura el comandament CWMODE=1 i un altre cop s'assegura que la tasca ha estat realitzada correctament.

El tercer pas que es realitza és configurar el nombre de connexions a les que es connectarà (Annex 10), i donat que només és una, es configura la funció CIPMUX=0.

Per acabar, es connecta a la xarxa indicada al principi de l'arxiu *main.c* mitjançant la funció CWJAP.c (Annex 11).

Per realitzar aquesta connexió, el que es fa primer és muntar l'ordre amb el nom de la connexió donada. Posteriorment s'envia l'ordre a l'ESP, i per acabar s'espera i s'assegura a que la connexió s'hagi realitzat de forma correcta.

Un cop connectat l'ESP a la xarxa indicada, la rutina "configNetwork" s'ha acabat.

5.3. Lectura i enviament de dades

Un cop connectat l'ESP a la xarxa, el programa entra en el bucle infinit "while(1)" de l'arxiu main.c (Annex 3) i per tant, les següents accions s'aniran repetint de forma indefinida fins que s'apagui el microcontrolador.

5.3.1. Readtemp.c

El primer que es fa en el bucle "while(1)" és llegir la temperatura mitjançant el sensor amb la funció "readtemp.c" (Annex 12).

Aquesta funció ja la tenia el Departament d'Electrònica, i per tant no l'he escrita jo. Tot i això s'hi ha fet algunes modificacions, per tal que connectes correctament amb la resta de funcions. El que s'ha fet sobretot és modificar les funcions perquè puguin treballar amb el compilador XC8, i modificar el format amb el qual retorna el valor de temperatura a llegir i així facilitar el seu posterior tractament (s'ha fet que el valor de temperatura retornat estigui en codi ASCII).

Per llegir la temperatura, el que fa la funció és seguir els passos que indica el manual del DS18B20 (el sensor de temperatura utilitzat), una còpia del qual es pot trobar a [5].

Per llegir la temperatura, el microcontrolador realitza quatre passos. El primer és assegurar-se que el sensor està connectat a la placa mitjançant la funció "Detect_Slave_Device()".

Si el sensor està connectat, el microcontrolador dóna l'ordre de mesurar la temperatura al sensor mitjançant la transmissió de l'ordre 0xCC, que segons el manual indica al sensor que no llegeixi la direcció ROM, i posteriorment envia l'ordre 0x44, que demana al sensor que llegeixi la temperatura.

El protocol utilitzat per enviar aquestes ordres és el "1-Wire", tal i com s'ha comentat anteriorment.

Un cop donades aquestes ordres, el microcontrolador s'espera a que la lectura s'hagi acabat amb la funció BusyDS18B20().

Un cop s'ha acabat mesurament l'adquisió de la temperatura per part del sensor, el

microcontrolador llegeix aquesta temperatura mitjançant l'ordre 0xCC, per tornar a evitar la lectura del valor ROM i posteriorment se li dóna la comanda 0xBE, que demana al sensor que contesti el valor adquirit de temperatura.

Un cop mesurada la temperatura, cal convertir-la a unitats d'enginyeria mitjançant la funció "DS18B20_decode_temp(scratchpad)".

Un cop descodificada la temperatura, es passa a codi ASCII, i es guarda la temperatura en la variable *valor_temperatura*. En el cas que el sensor no estigues connectat, el valor retornat és "--------".

5.3.2. Cipstart.c

Un cop llegit el valor de la temperatura per part del sensor, s'inicia la rutina CIPSTART, que es dedica a obrir la connexió entre el microcontrolador i la base de dades (Annex 13).

El que fa aquesta funció és iniciar una connexió TCP amb la base de dades (el link donat a l'inici del document main.c), seleccionant el port 80.

La funció en general realitza el mateix que les anteriors. Primer genera l'"string" amb l'ordre *cipstart* i les dades que necessita aquesta ordre, després l'envia i per acabar s'assegura que el mòdul wifi ha rebut l'ordre correctament.

5.3.3. Cipsend.c

Per acabar la rutina main, posteriorment a la inicialització de la connexió entre el microcontrolador i la base de dades, el microcontrolador s'espera (només la primera vegada que es realitza el cicle) a que la connexió s'estableixi, i després envia el valor de temperatura a la base de dades mitjançant la funció *cipsend* (Annex 14).

El primer que fa la funció és muntar l'"string" amb l'ordre a donar a l'ESP i les dades donades pel microcontrolador (link de la base de dades i el número del microcontrolador definit al principi per evitar problemes en el cas de que més d'un microcontrolador es connecti a la vegada a la base de dades). Posteriorment, conta la longitud d'aquest "string" i envia l'ordre de començar la transmissió dels X caràcters comptats. Posteriorment, envia l'ordre "GET" (la funció de la qual es comenta a continuació) amb el valor de la temperatura mesurat a la base de dades.

Un cop enviada aquesta dada, el microcontrolador torna a començar la rutina de mesura de la temperatura, connexió... Durant aquest procés, el led B1 es va encenent i apagant per indicar el correcte funcionament del programa.

6. Base de dades

A part de totes les tasques que realitza el microcontrolador, el gestor de la base de dades també ha de ser capaç d'entendre les dades que va rebent del microcontrolador i ha de ser capaç de guardar-les de forma correcta.

Abans de començar això, cal crear la base de dades, i per això, es pot utilitzar el programa XAMPP, que es pot descarregar del link [8] de la bibliografia.

Un cop descarregat e instal·lat aquest programa, s'obra el programa i s'inicialitza la base de dades clicant el botó *start* del mòdul Apache (Fig. 43).

Posteriorment, un cop iniciada la base de dades, clicant la pestanya *Netstat* es pot veure la direcció o link que cal per connectar el microcontrolador a la base de dades (Fig. 44).

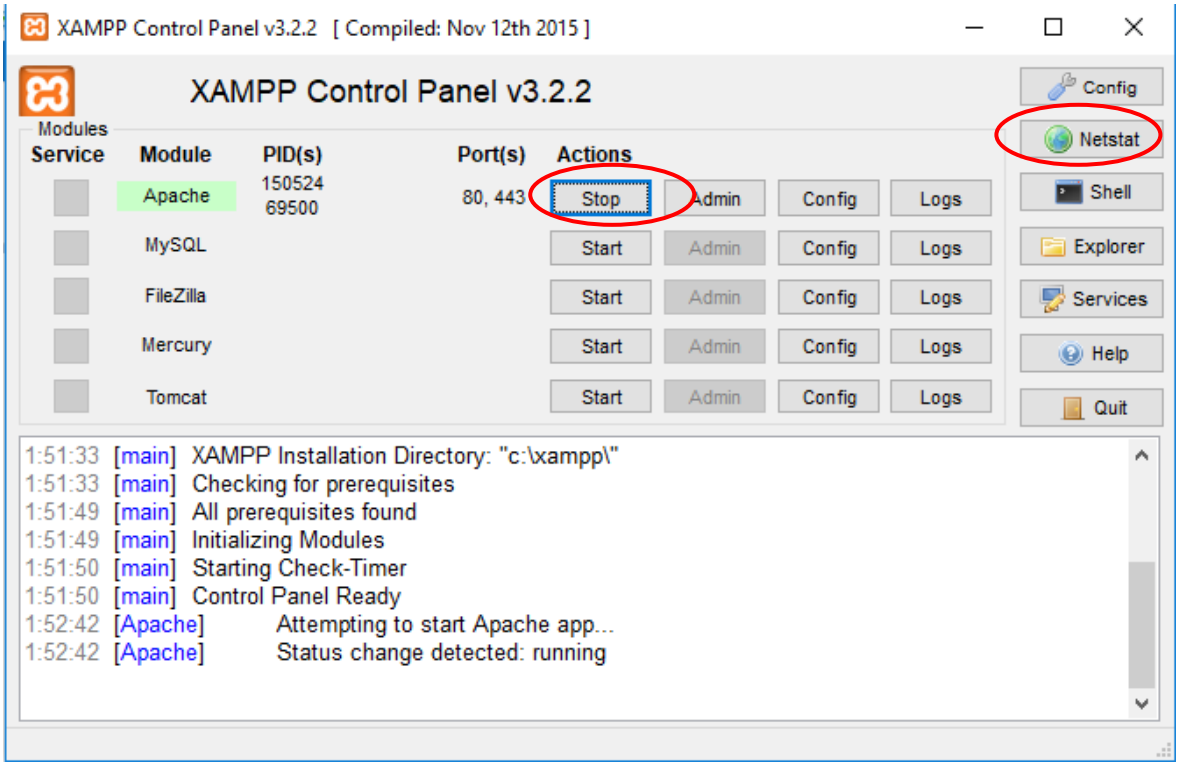


Fig. 43. Panell del XAMPP 1. Font: pròpia.

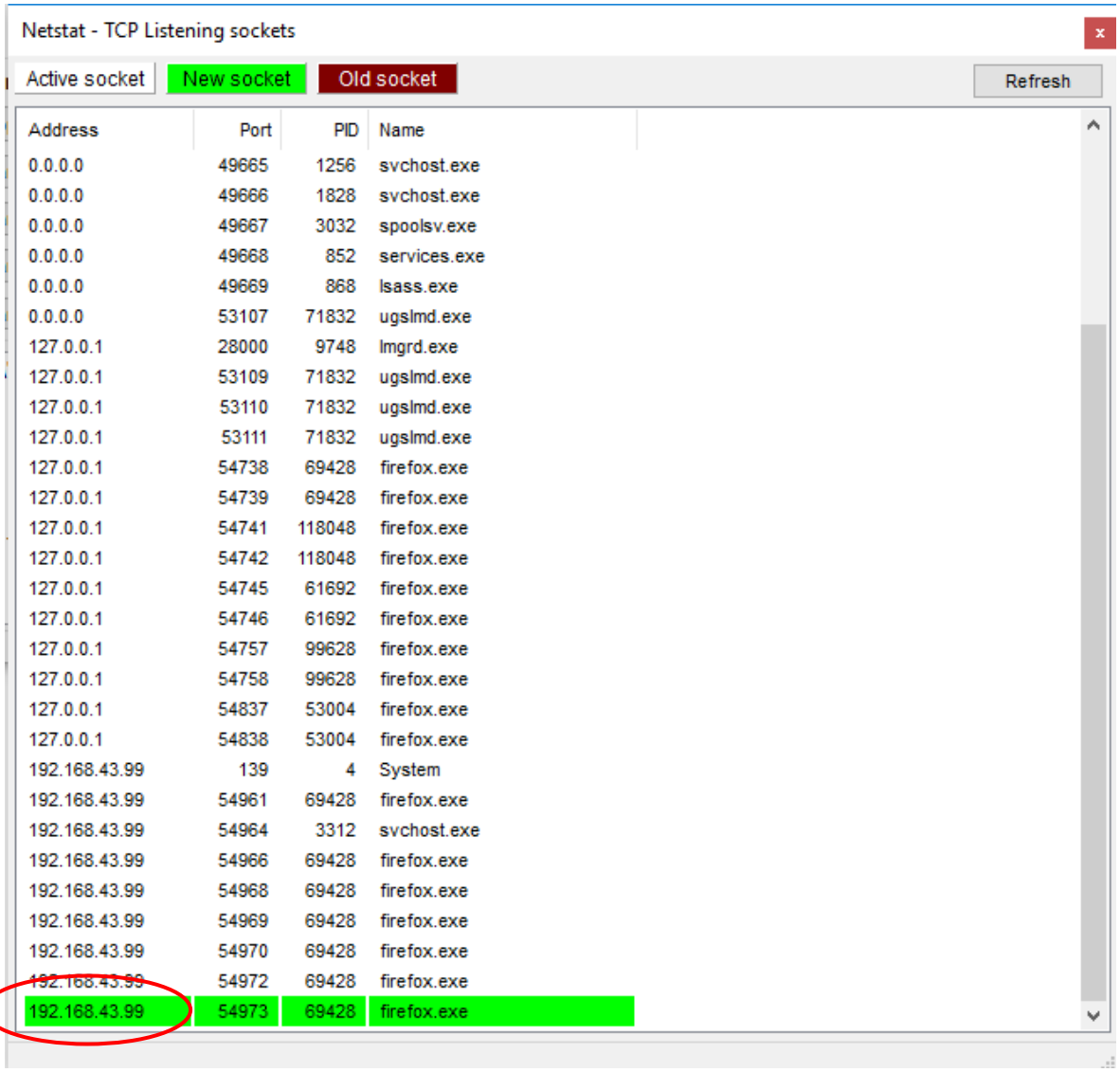


Fig. 44. Panell del XAMPP 2. Font: pròpia.

Un cop configurada la base de dades, s'escriu un arxiu escrit en lenguatge PHP capaç de llegir les dades que li envia el microcontrolador mitjançant la funció GET, que és una funció que serveix per fer una sol·licitud a un servidor, de forma semblant a quan s'escriu un link al navegador i es clica *enter*.

El que fa el microcontrolador és enviar una sol·licitud GET en la que demana a la base de dades que obri l'arxiu *read.php* i se li dóna la temperatura a aquest arxiu.

El que fa aquest arxiu a l'executar-lo és agafar la temperatura rebuda i guardar-la en un arxiu de text anomenat "data.txt".

Per acabar, per mostrar les dades que es van rebent de forma còmoda, s'utilitza el programa de lliure distribució KST Plot 2.0 [9].

Per inicialitzar el programa KST, s'ha d'arrossegar l'arxiu de text *data.txt* o es van bolcant les dades de temperatura (Fig. 45), i posteriorment es clica *next* (Fig 46).

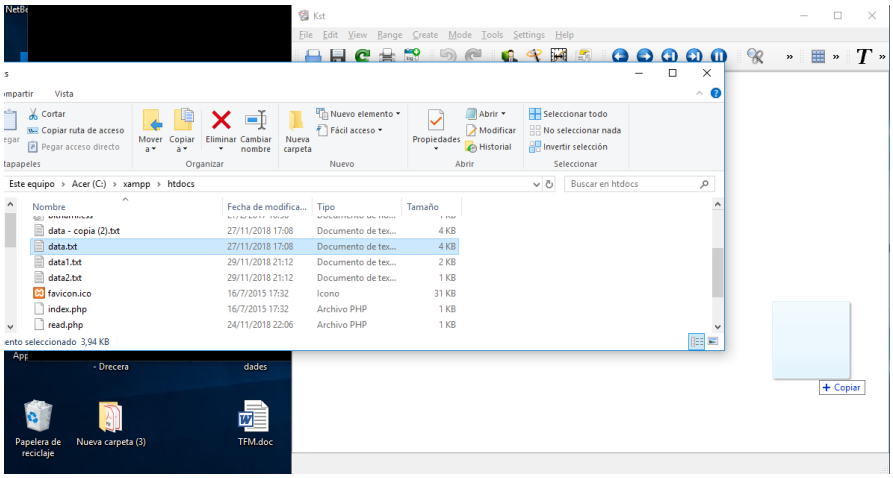


Fig. 45. Configuració del programa KST 1. Font: pròpia.

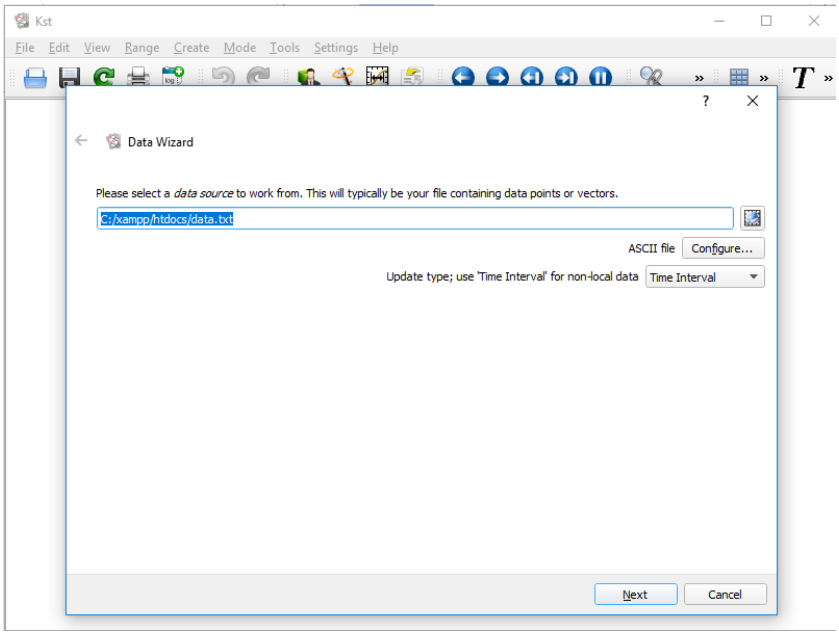


Fig. 46. Configuració del programa KST 2. Font: pròpia.

Després es selecciona la columna 1 i es clica *next*, posteriorment es configura la següent pantalla de la mateixa manera que la Fig. 47, un altre cop es prem *next*, i es torna a configurar la finestra que surt igual que la Fig. 48.

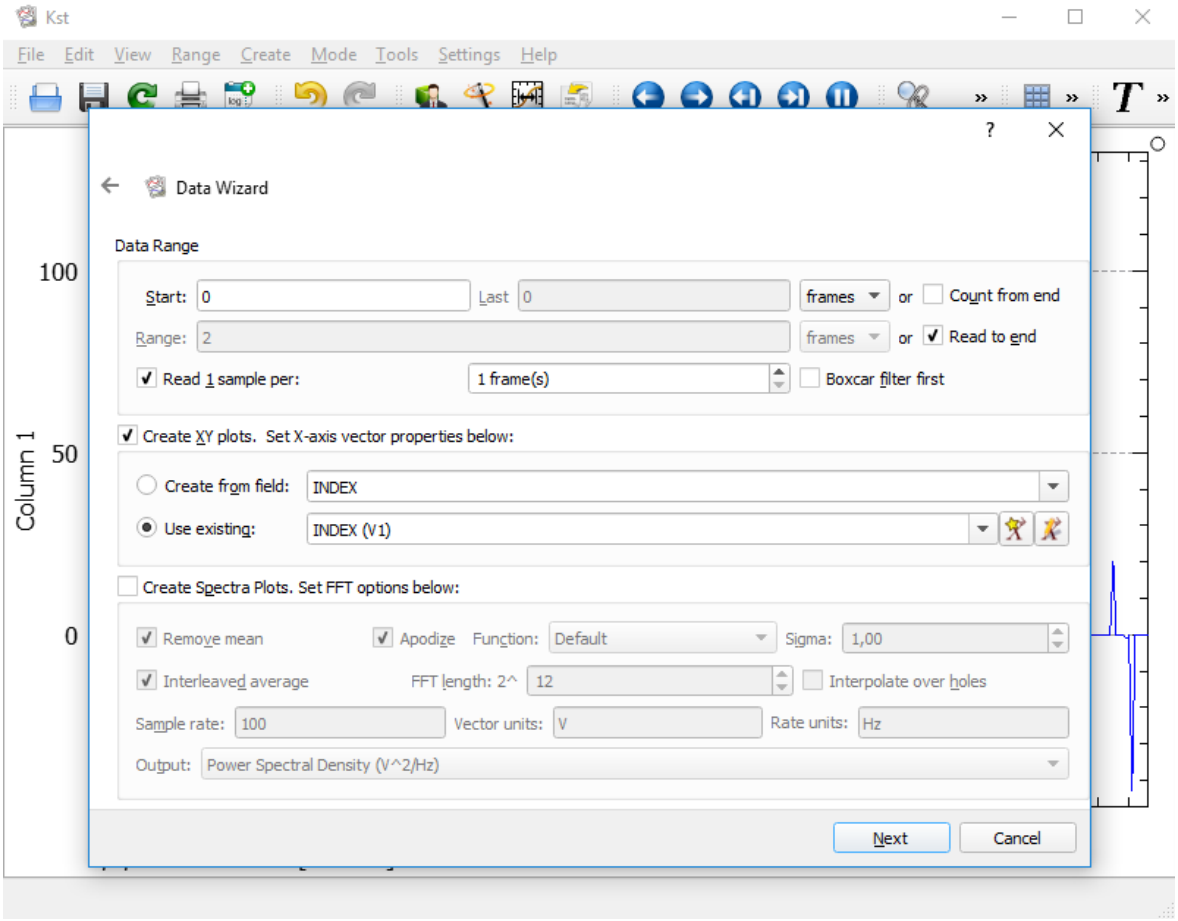


Fig. 47. Configuració del programa KST 3. Font: pròpia.

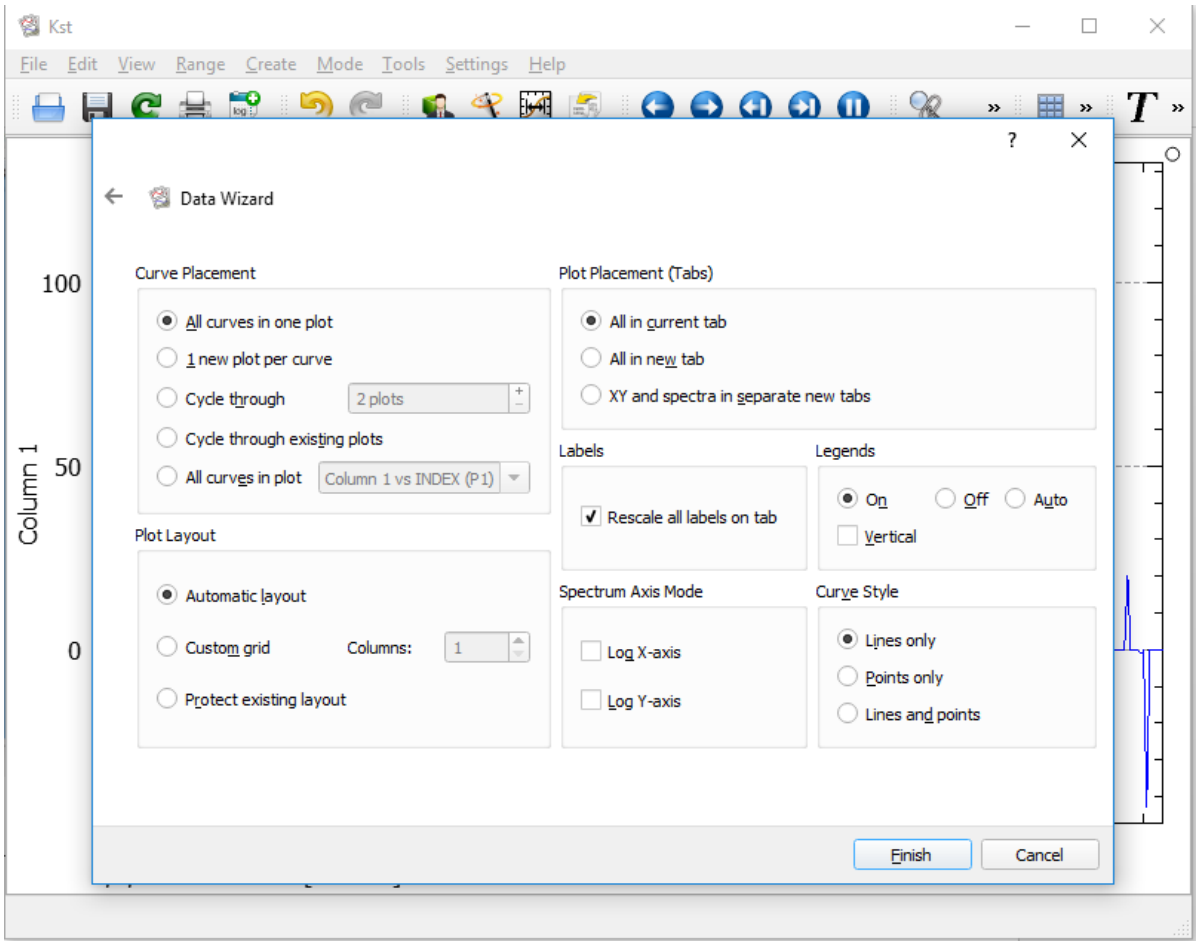


Fig. 48. Configuració del programa KST 4. Font: pròpia.

Un cop configurat el KST, el mateix programa s'encarrega de llegir el valor de la temperatura que es va actualitzant aproximadament cada 4 segons i mostrar-lo en temps real (complint així el primer objectiu del projecte, que és llegir la temperatura mitjançant un sensor, i enviar-la mitjançant una xarxa wifi a una base de dades i mostrant-la en temps real) (Fig. 49).

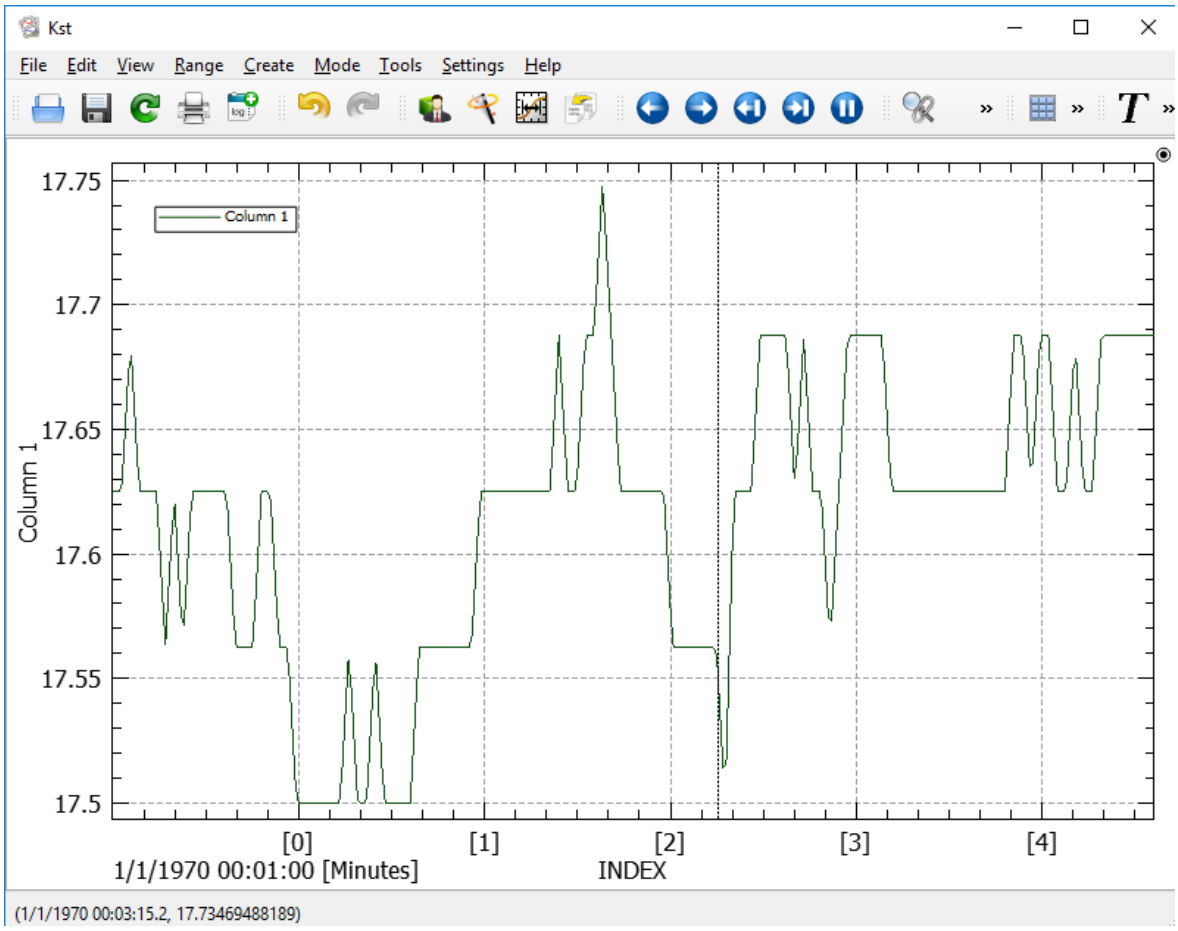


Fig. 49. Configuració del programa KST 5. Font: pròpia.

7. Comunicació entre dos microcontroladors

El segon objectiu del projecte, donat que un dels principis de l'Internet of things industrial és la comunicació entre màquines, és aconseguir que dos microcontroladors siguin capaços de comunicar-se entre ells i enviar-se dades. Per tant, aprofitant el codi ja escrit fins al moment per enviar dades a una base de dades, s'escriu un segon programa per a un microcontrolador, de forma que aquest sigui capaç de rebre la dada enviada mitjançant el programa anterior, i mostrar-la a un pantalla LCD.

7.1. Llibreries

A l'igual que en el cas anterior, el programa incorpora una sèrie de llibreries per tal de simplificar el programa principal, e igual que en el cas anterior el programa incorpora la llibreria *ConfigBits.h,* que configura els bits del microcontrolador (Annex 1), i la llibreria *definicions.h*, que en aquest cas s'hi ha fet unes lleugeres modificacions respecte al cas anterior, ja que les funcions cridades no són les mateixes (Annex 16).

A més s'inclouen vàries llibreries neccesàries per fer funcionar la pantalla en les que es defineixen variables i funcions a l'igual que en les anteriors (Asciihex8x16.h (Annex 17)), (clk.h (Annex 18)), (lcd_22.h (Annex 19)) i (System.h (Annex 20)).

Aquestes llibreries les ofereix el fabricant de la pantalla [4] igual que el programa necessari per fer funcionar la pantalla, per tant no té molt sentit descriure en detall el funcionament de cada una d'elles.

7.2. Main.c del PIC receptor de dades i encarregat de la LCD

L'arxiu principal del programa que es dedica a mostrar la dada rebuda a la pantalla, igual que qualsevol programa per a controlar un microcontrolador, és l'arxiu main.c. En aquest cas, igual que amb el programa anterior, aquest arxiu (Annex 21), al principi del programa crida a totes les llibreries que necessita el programa (les comentades en l'apartat anterior), i posteriorment inicialitza una sèrie de variables, entre elles el "buffer" *f*, una llista anomenada link que posteriorment s'utilitzarà per guardar la IP del microcontrolador que treballa com a port, i una llista anomenada temperatura, que s'utilitza per crear l'"string" que es mostra a la pantalla amb la temperatura.

Posteriorment, tenim una rutina anomenada "initializef", que neteja el buffer *f* i després comença la rutina *main*.

Dins aquesta rutina, es comença configurant el registre *OSCCON*, igual que abans per tal de configurar l'oscil·lador intern del programa a 8 MHz.

Posteriorment es configuren els ports A i D com a sortides (excepte el bit A1 que es configura com a entrada). Després es configura el timer1 igual que abans i a més es configura el registre *ADCON1* perquè els pins A1, A2, A3, A5, E0, E1 i E2 es comportin com entrades analògiques.

Un cop inicialitzats els registres anteriors, s'inicialitza la pantalla amb la rutina *init_lcd,* que està a l'arxiu lcd_22.c (Annex 22), i el que fa aquesta rutina és inicialitzar el protocol de comunicació SPI del controlador de forma semblant als casos anteriors.

El protocol SPI és un altre protocol de comunicació semblant als anteriors (comunicació sèrie asíncrona i 1 Wire), que es caracteritza per treballar amb 3 cables, un de transmissió de dades i un altre de recepció de dades (igual que protocol amb la UART), però en aquest cas, aquest protocol també té un senyal anomenat "clock", que es dedica a donar els senyals de rellotge a l'element connectat al microcontrolador, de forma que no s'ha de configurar una velocitat de transmissió, a més de que permet treballar a una velocitat major que protocol la comunicació sèrie asíncrona basat en UART (en el cas de que hi hagi més d'un element connectat al microcontrolador, s'introdueix un cable/senyal més per element connectat per així poder seleccionar l'element al que es transmeten les dades) (Fig. 50).

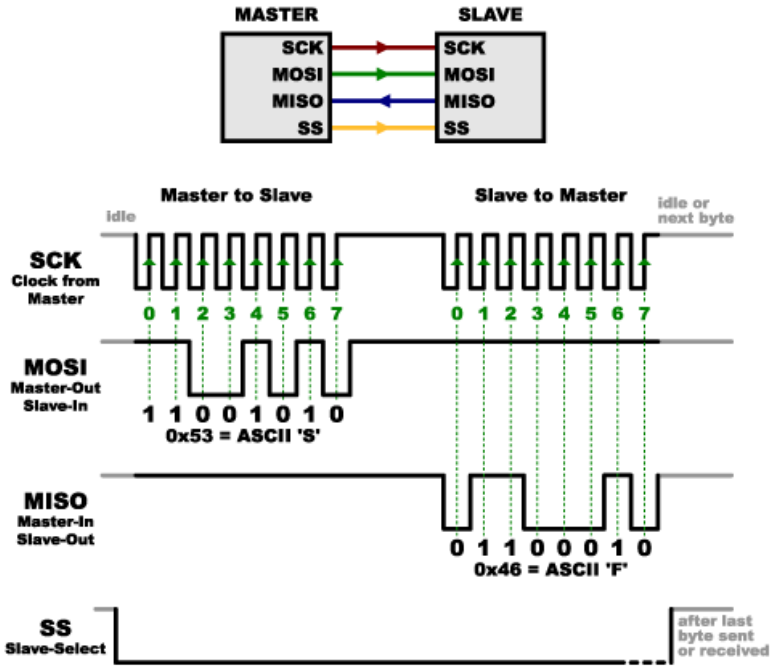


Fig. 50. Protocol SPI. Font: [22].

Després d'inicialitzar la comunicació amb la pantalla, s'executa una rutina *LCD_test* en la que posa el fons de pantalla de color groc.

Posteriorment a això, s'encenen els leds de la placa per indicar que no s'ha produït cap problema, i després s'escriu a la pantalla la frase "Starting ESP" amb la funció "DisplayString".

Aquesta funció també està a l'arxiu *lcd_22.c* (Annex 22), que com ja s'ha dit, l'ofereix el propi fabricant de la pantalla [4], i per tant, donat que bàsicament s'ha incorporat al programa, no té sentit comentar-la a fons.

Un cop escrit l'anterior string, s'inicialitza la rutina "start", que és la mateixa que en el cas anterior on s'inicialitza el perifèric UART (Annex 4).

Després d'això es neteja el "buffer" *f* amb la rutina "initializef", i es mostra a la pantalla l'"string" ESP8266 "started".

Posteriorment, es mostra la frase "Configurating ESP", i s'inicialitza la rutina ESPconfig, que en general fa el mateix que el cas de la rutina configNetwork del programa anterior.

Primer, igual que en el cas anterior, el microcontrolador envia el comandament AT, i espera la resposta. Si la resposta és OK, això implica que la comunicació s'està produint de forma correcta, i per tant es pot procedir a donar les següents ordres.

Primer s'envia l'ordre de configurar l'ESP com a punt d'accés amb el comandament CWMODE=3. Posteriorment es dóna l'ordre de que més d'un element es pugui connectar a la xarxa creada per l'ESP (CPMUX=1), i per acabar es configura el valor del port amb el comandament CIPSERVER=1,80.

Després d'això, l'ESP ja està configurat, i es mostra a la pantalla el comandament "ESP configurated".

Després, donat que per connectar-se al servidor creat per l'ESP, el microcontrolador amb el sensor necessita saber la direcció IP del receptor de les dades, s'inicialitza la rutina *CheckIP* (Annex 24).

Aquesta funció, el que fa és enviar el comandament CIFSR per determinar la direcció IP del mòdul wifi, i posteriorment llegeix la resposta i crea un "string" amb aquest valor. A més, aquesta funció també determina la direcció i la contrasenya de la connexió wifi creada per tal de que posteriorment ens puguem connectar a aquesta. Un cop determinats aquests valors es mostra a la pantalla a continuació i s'inicia la rutina de sincronització.

Per determinar si algun element s'ha connectat a l'ESP, el que fa el microcontrolador és enviar el comandament CWLIF que determina la IP dels clients connectats a l'ESP, si la IP retornada és 0.0.0.0, significa que no hi ha ningú connectat, però si la resposta és una IP més normal, és a dir, una que comença per 192.X.X.X, significa que algú s'ha connectat.

Un cop connectat el microcontrolador amb el sensor al microcontrolador amb la pantalla, apareix el missatge "sensor Connected" a la pantalla, i s'inicia la transmissió d'informació.

Per a la transmissió d'informació, el que fa el microcontrolador amb la pantalla és entrar en un bucle que es va repetint de forma contínua en el que s'espera a rebre un missatge, el llegeix, descodifica d'aquest missatge el valor de la temperatura, i el mostra a la pantalla (Fig. 51).

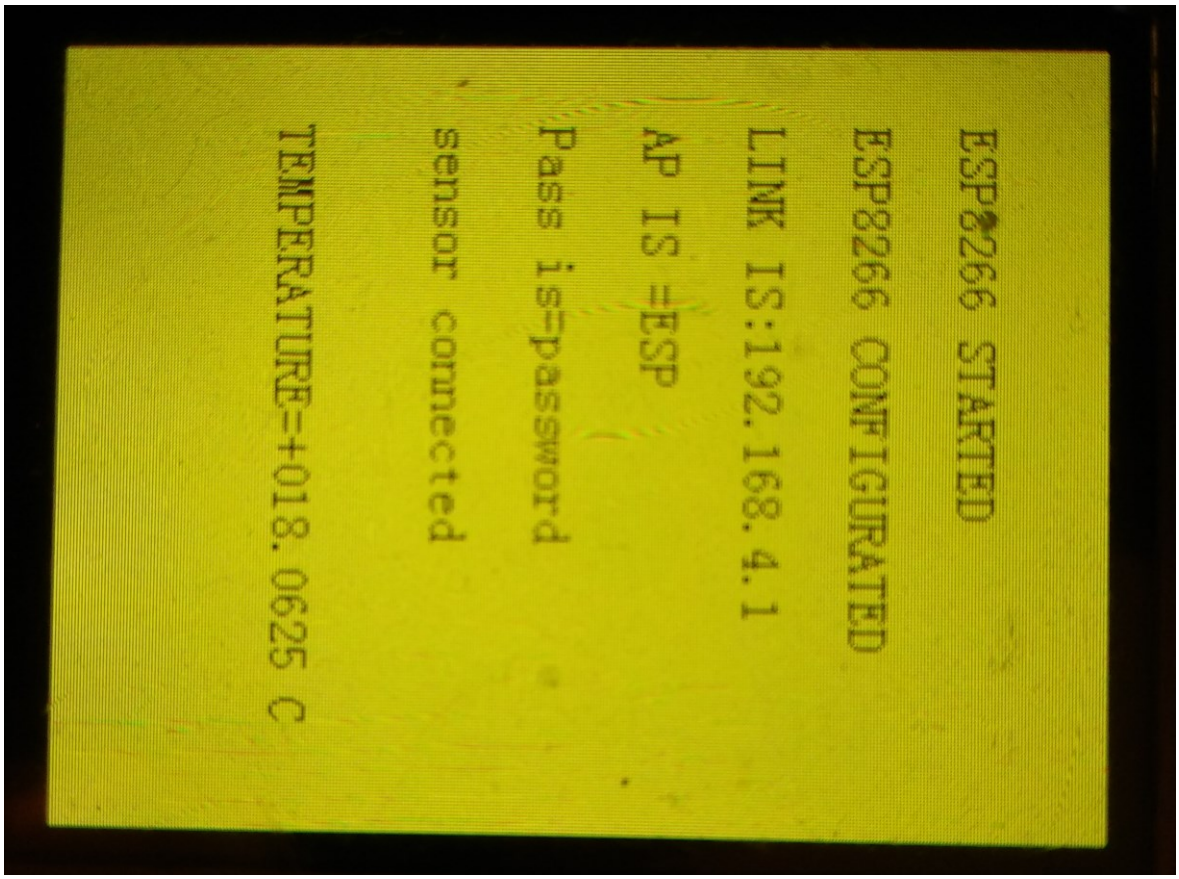


Fig. 51. Temperatura rebuda mostrada a la pantalla LCD. Font: pròpia.

8. Main.c del µC emissor al µC amb la LCD

Per acabar, s'ha hagut d'escriure un últim programa capaç de connectar-se amb el microcontrolador amb la pantalla LCD i enviar-li les dades de la temperatura.

Aquest és molt semblant a l'escrit al principi per enviar dades a la base de dades, i només s'ha introduït uns petits canvis per poder seleccionar el moment en el que s'envien les dades (Annex 26).

Els canvis que s'han introduït són la utilització del botó *S1* que incorpora la placa utilitzada per determinar el moment en que s'envien les dades a la placa amb la pantalla, a més dels canvis en el link, el nom d'usuari i la contrasenya de la xarxa a la qual es connecta el microcontrolador.

9. Manual simplificat

9.1. Comunicació entre el µC i la base de dades

El primer que s'ha de fer és crear un nou projecte, tal i com s'indica a l'apartat 3.1, e importar els arxius de les llibreries (BitsConfig.h i functinit.h) a la carpeta header files deixant l'opció "copy" seleccionada.

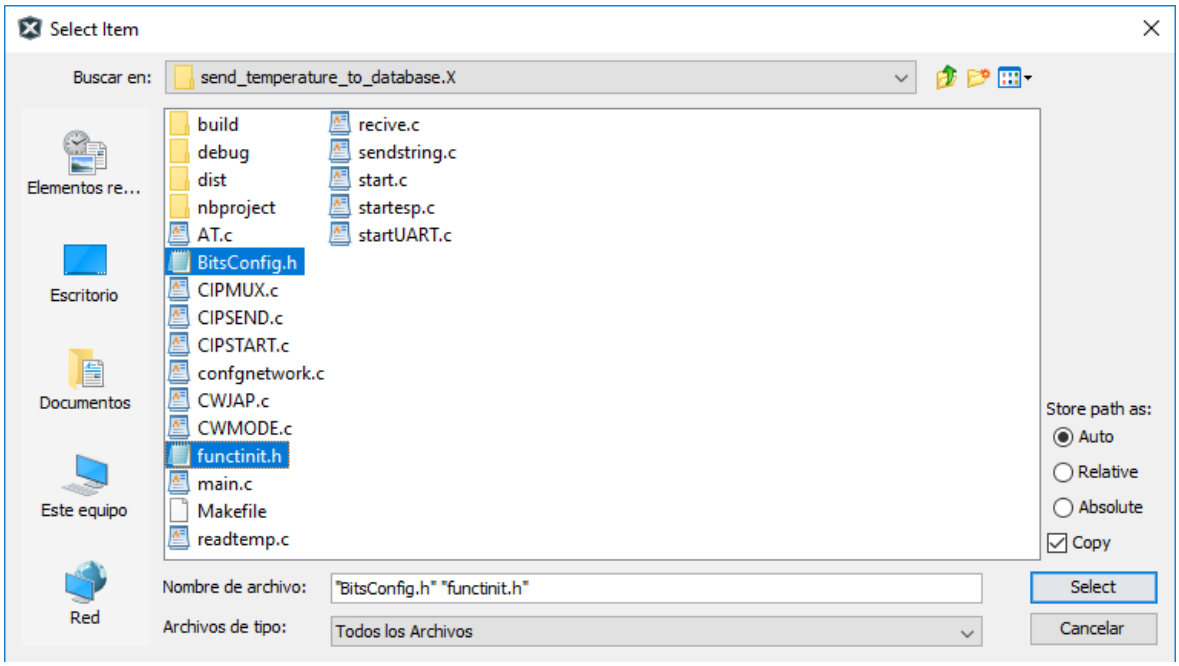


Fig. 52. Llibreries. Font: pròpia.

Posteriorment, anem a la carpeta *source files* i afegim els arxius de programa, tal i com es pot veure a la Fig. 53, deixant l'opció "copy" un altre cop seleccionada.

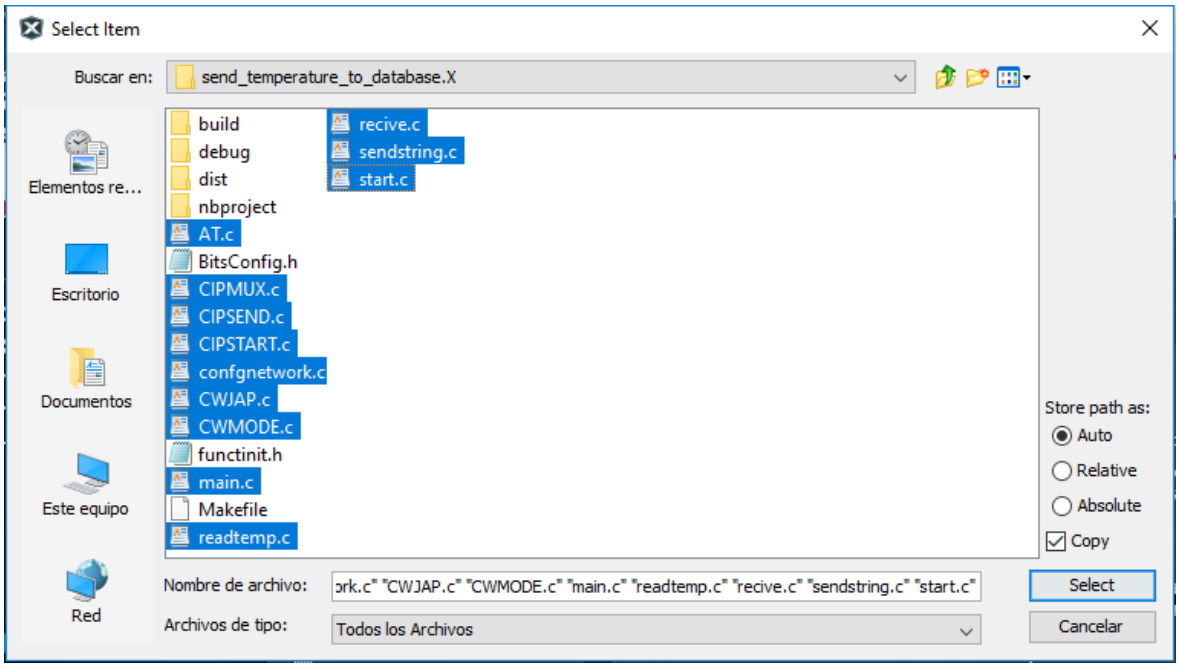


Fig. 53. Arxius del programa. Font: pròpia.

Un cop incorporats els arxius que requereix el programa, es configuren les variables de la connexió wifi (en nom del punt d'accés, la contrasenya i la direcció IP de la base de dades).

A més, s'ha de seleccionar un número de l'1 al 9 per poder diferenciar els microcontroladors, en el cas que més d'un es connectés a la base de dades alhora (Fig 54).

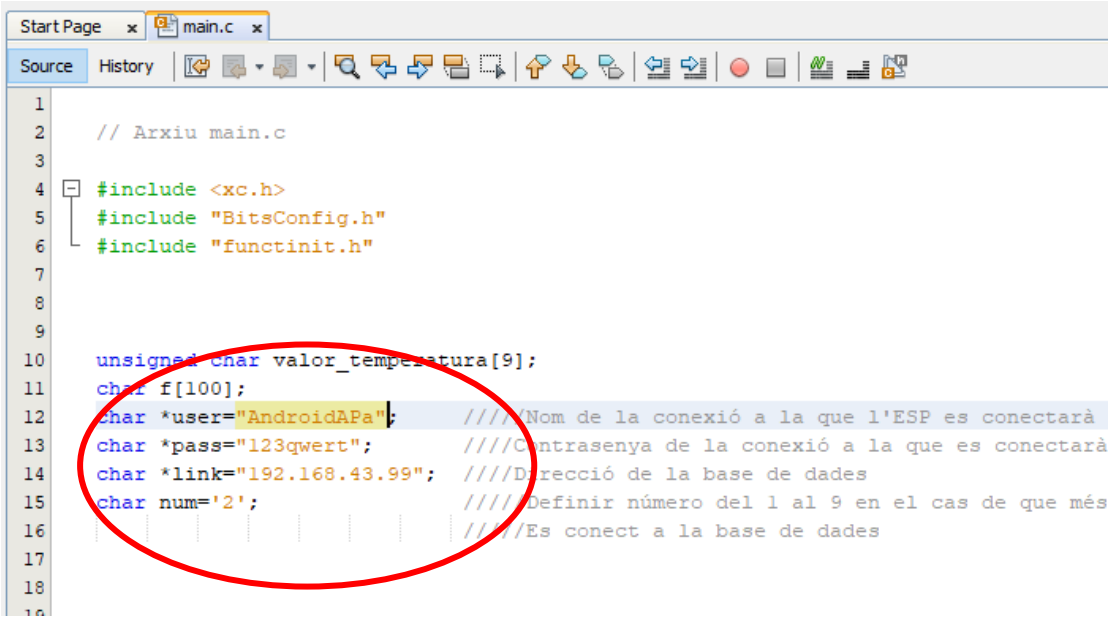


Fig. 54. Arxius del programa. Font: pròpia.

Paralel·lamen a això cal deixar configurada la base de dades.

Primer cal tenir connectat l'ordinador a la xarxa wifi en la que es produirà la comunicació i instal·lat el programa XAMPP.

Posteriorment, s'obre el programa XAMPP i s'inicia un servidor Apache (Fig. 55) i a la pestanya *Netstat* es pot veure el link al que es pot connectar el microcontrolador (Fig. 56).

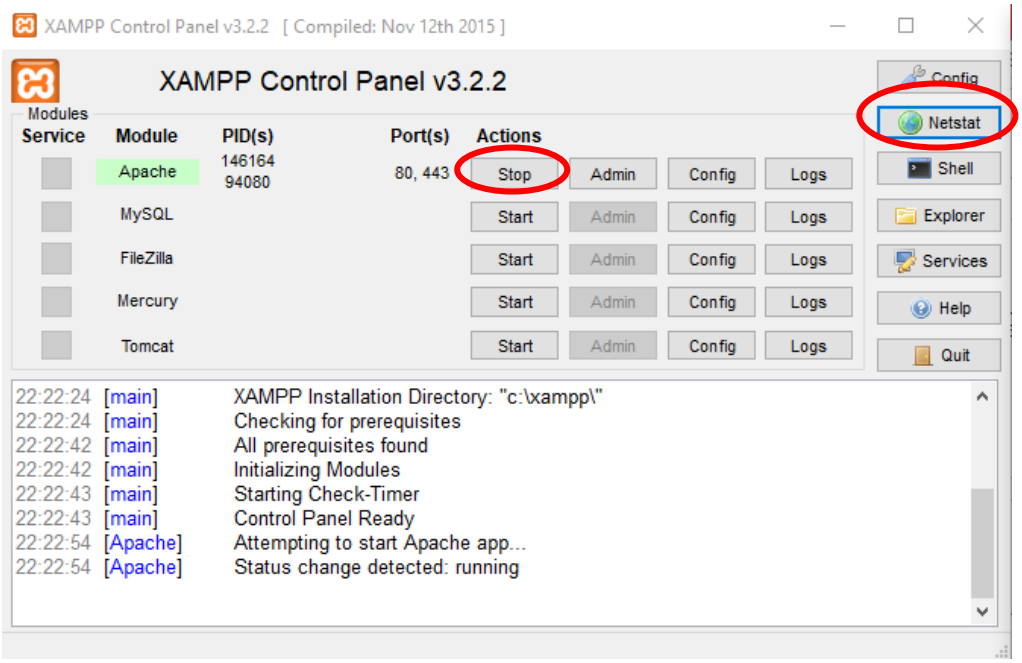


Fig. 55. XAMPP. Font: pròpia.

Un cop iniciat el servidor, cal tenir en compte que els arxius *php* que utilitza el servidor per executar l'ordre de guardar les dades estiguin presents a la carpeta adequada.

Aquests han d'estar a la carpeta *htdocs* dins la carpeta xampp (Fig. 57).

Per acabar i mostrar les dades enviades, s'utilitza el programa KST 2, tal i com s'indica a l'apartat 5 i així mostrar en temps real els valors de temperatura mesurats.

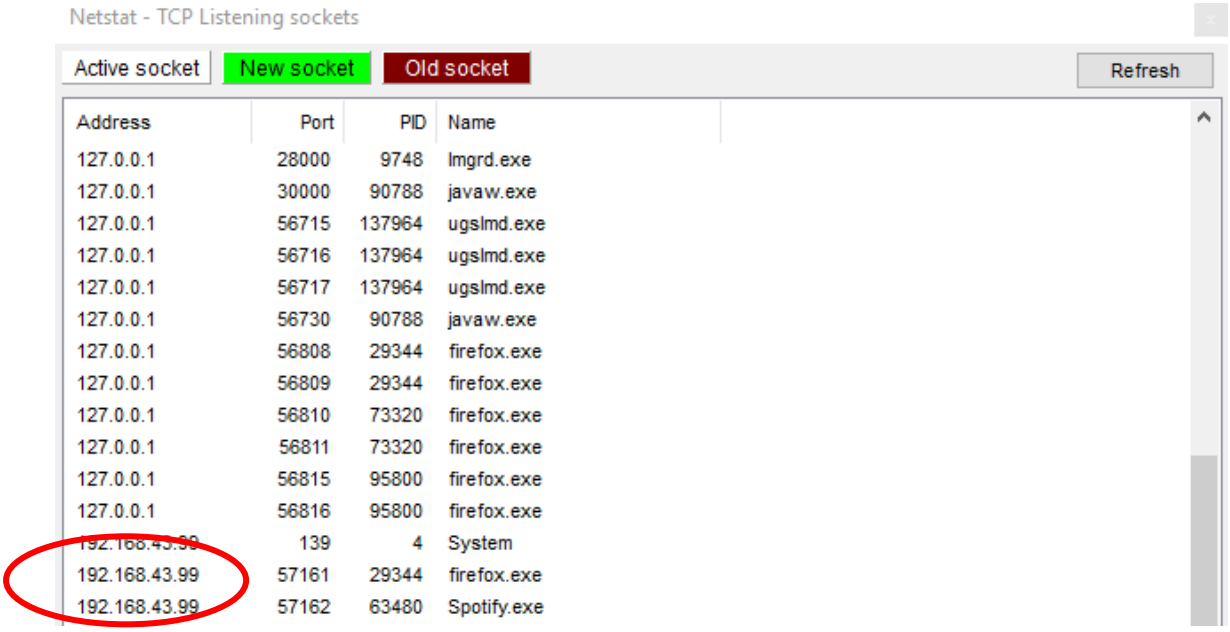


Fig. 56. XAMPP Netstat. Font: pròpia.

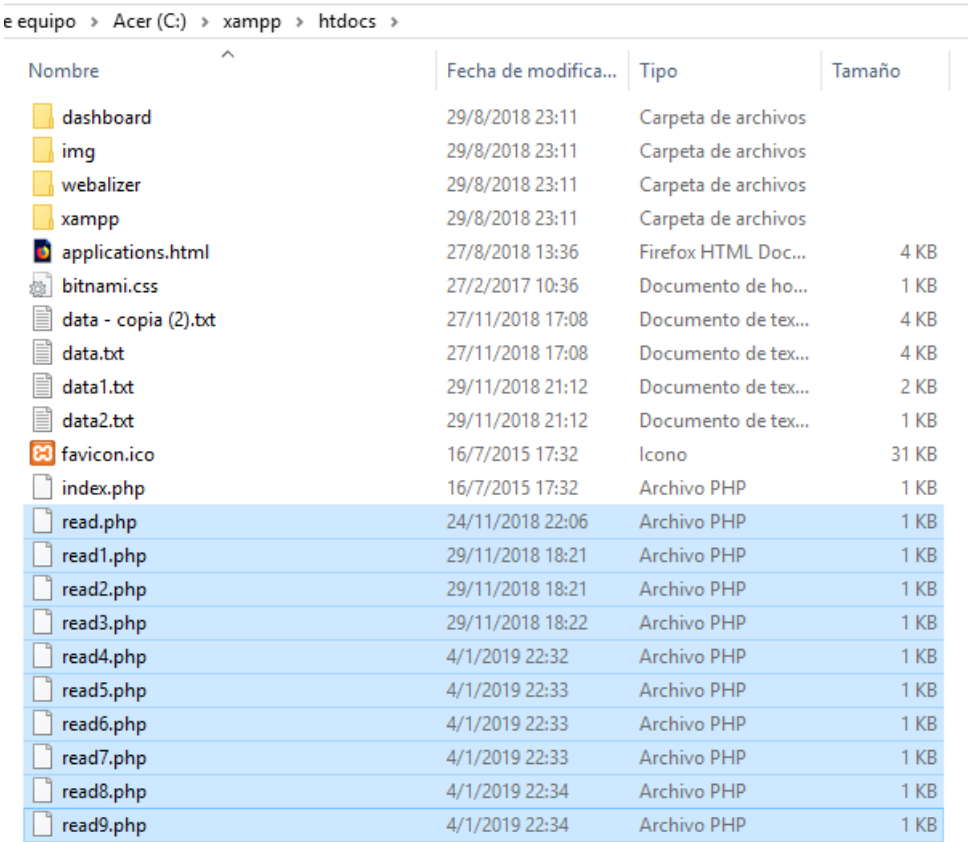


Fig. 57. Direcció dels arxius read.php. Font: pròpia.

9.2. Comunicació entre dos µC

En aquest cas, primer s'ha de crear un altre projecte igual que abans, i s'ha d'afegir els següents arxius a la carpeta header files (Fig. 58).

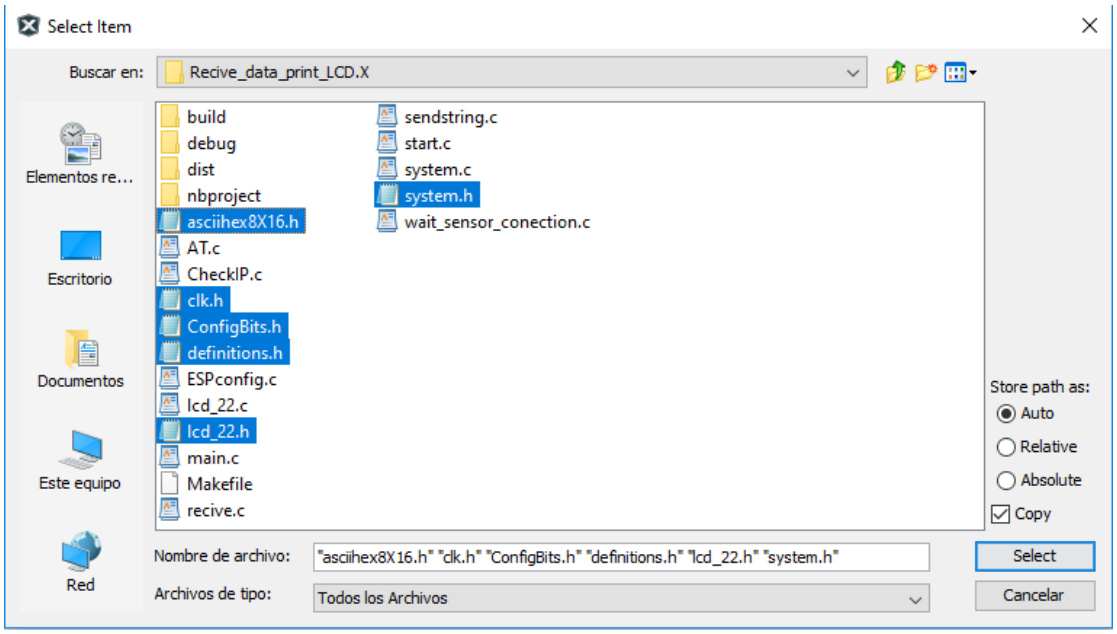


Fig. 58. Llibreries. Font: pròpia.

Posteriorment, s'afegeixen els arxius de programa a la carpeta source files (Fig. 59), i ja es pot programar el microcontrolador.

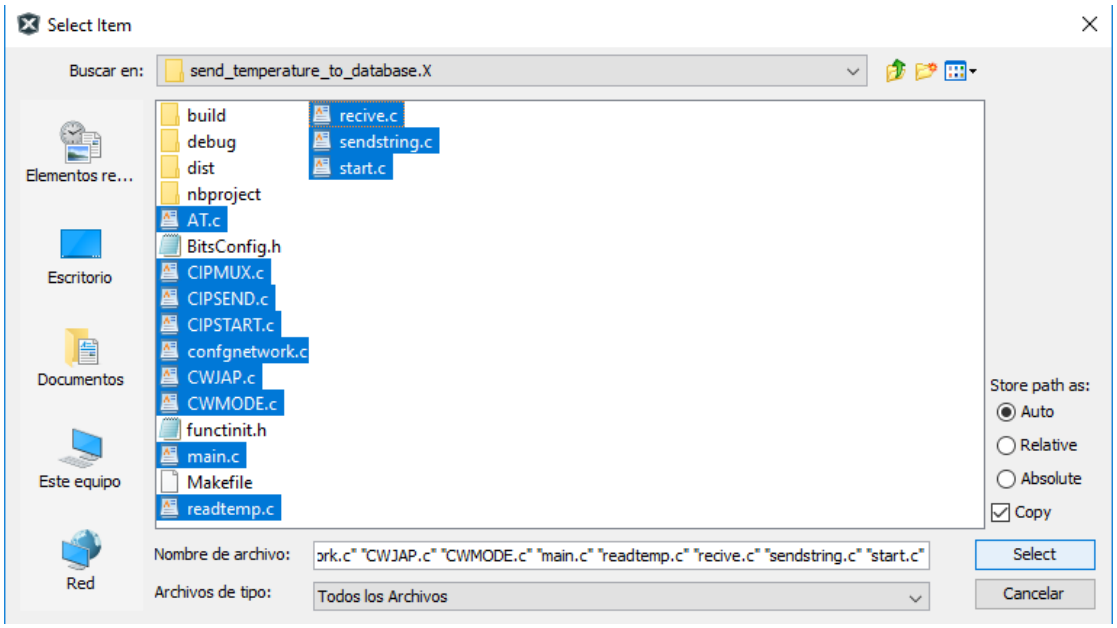


Fig. 59. Arxius de programa. Font: pròpia.

Paralel·lament, també s'ha de configurar l'altre microcontrolador amb la direcció IP del receptor, el nom de la xarxa wifi i la contrasenya.

Per facilitar aquesta tasca, el primer controlador, el que té la pantalla, mostra en aquesta els valors que necessitem. Per exemple, en la Fig. 60 es mostra que la contrasenya és "password", l'usuari és "ESP", i la IP és "192.168.4.1".

Per tant, el que es pot fer és, un cop programat el microcontrolador amb la pantalla, inicialitzar per veure aquestes variables que es necessiten i un cop determinades, modificar el programa de l'emissor i posteriorment programar-lo.

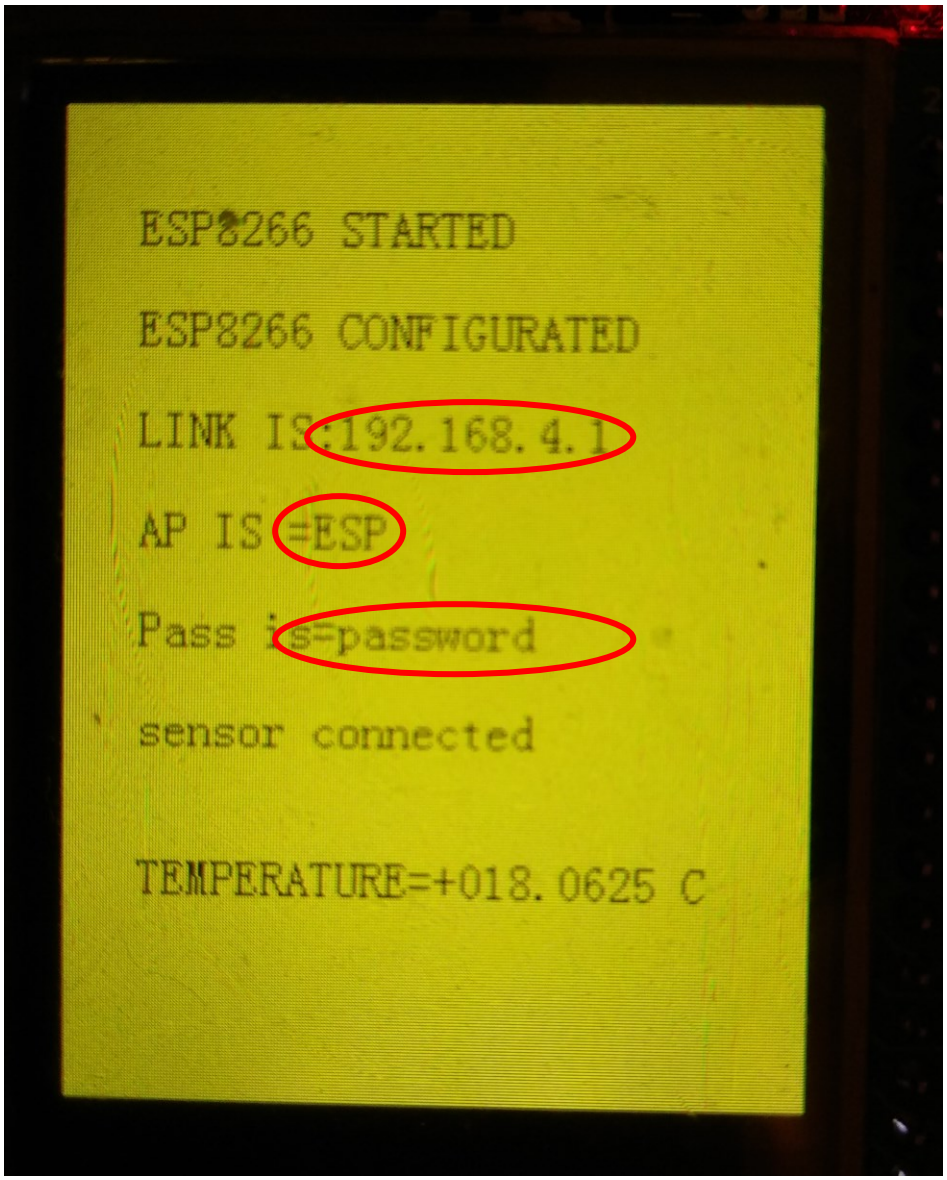


Fig. 60. Pantalla LCD. Font: pròpia.

Posteriorment, s'afegeixen les mateixes llibreries que en el primer cas (Fig. 61) a la carpeta *header files* i els arxius de programa de la Fig. 62 a la carpeta *source files*.

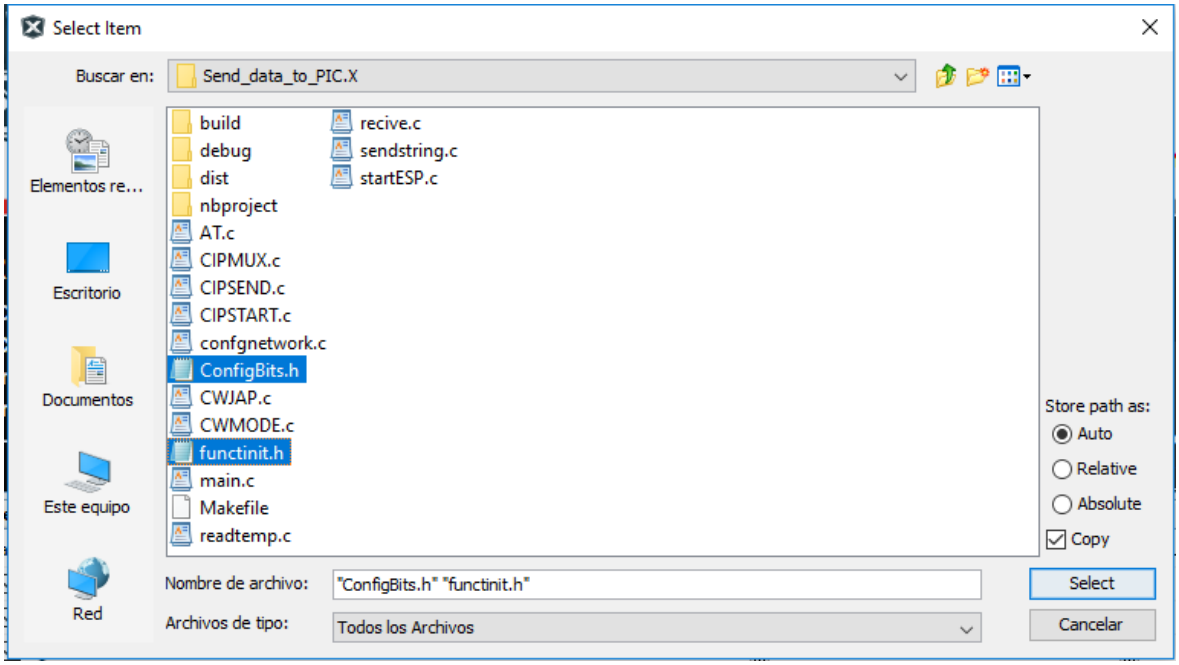


Fig. 61. Llibreries. Font: pròpia.

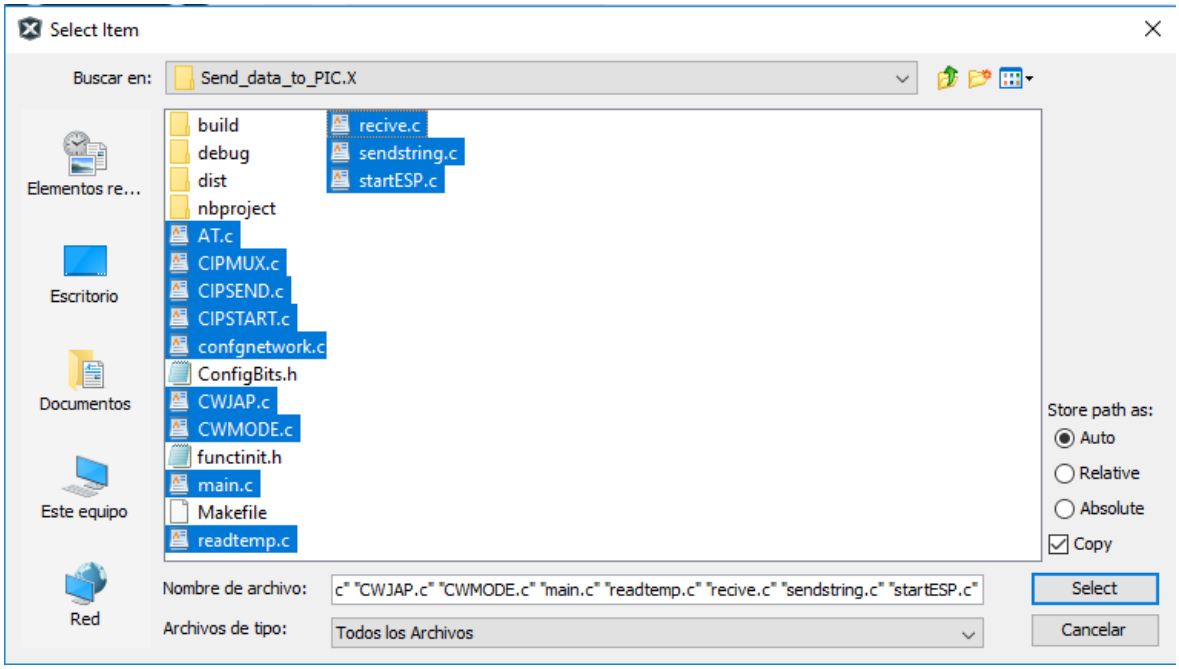


Fig. 62. Arxius de programa. Font: pròpia.

Posteriorment, a l'arxiu *main.c* es configuren els valors determinats anteriorment de la connexió creada (usuari, contrasenya i link), i ja es pot programar el microcontrolador.

Un cop programats els dos microcontroladors, cal sincronitzar la transmissió de dades.

Per tal de fer això, cal que inicialment els dos microcontroladors estiguin apagats (un cop determinats els valors necessaris per programar el segon microcontrolador, es pot apagar el primer, ja que tant la IP com les dades de la connexió no es modifiquen a l'apagar-lo).

El que s'ha de fer primer és encendre el microcontrolador amb la pantalla, i exactament (si passa un segon no passa res) en el moment en el que apareix el missatge "waiting sensor" encendre el segon microcontrolador (Fig. 64).

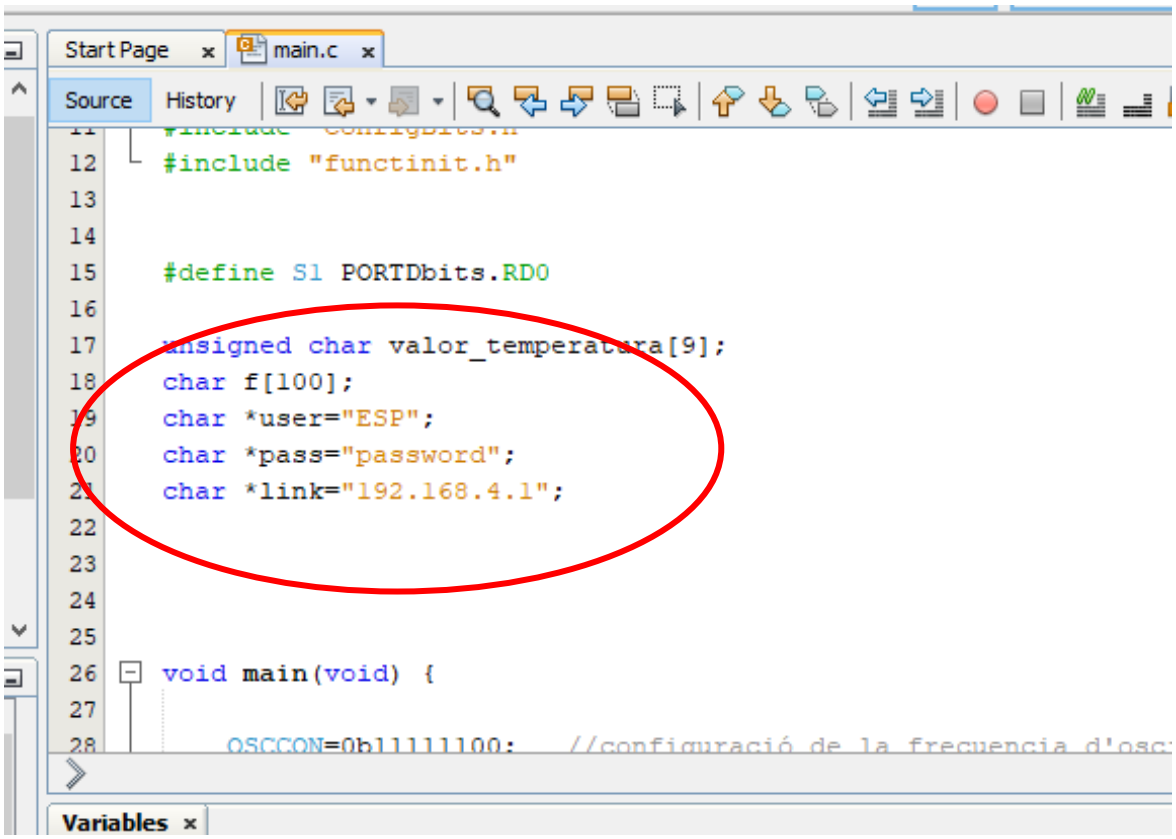


Fig. 63. Dades a modificar. Font: pròpia.

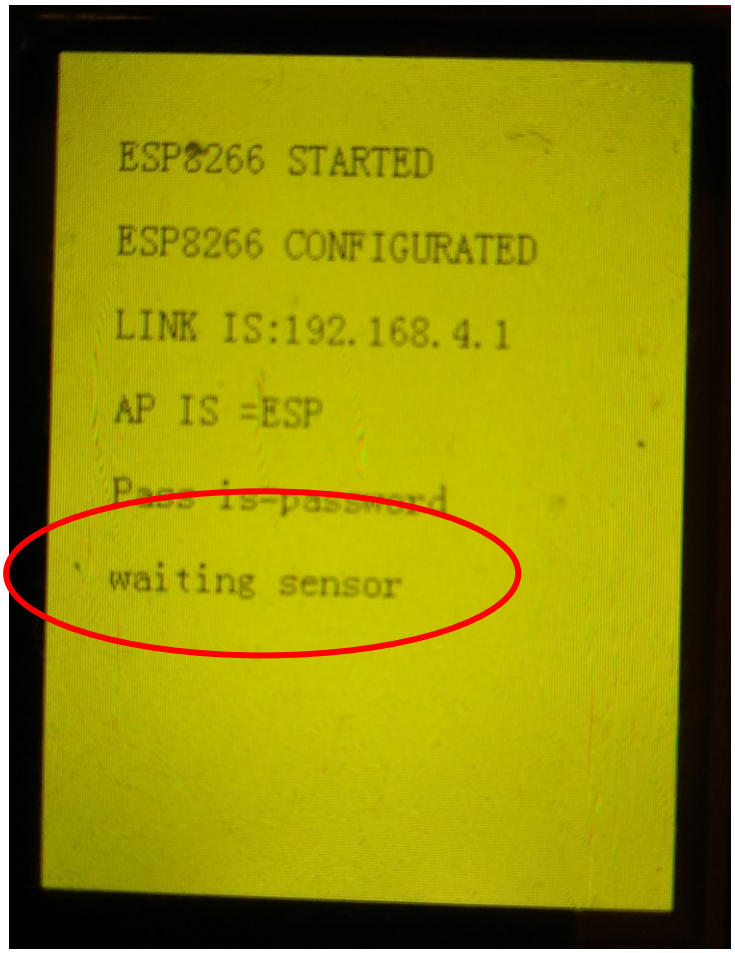


Fig. 64. Pantalla LCD 2. Font: pròpia.

Posteriorment, ens hem d'esperar aproximadament 40 segons fins que tots els LEDs de control de la placa amb el microcontrolador emissor de dades s'encenguin (Fig. 65).

Un cop passi això, automàticament cada 10 o 12 segons, el microcontrolador sensor llegirà i enviarà la dada de temperatura al segon microcontrolador. Sinó, també es pot prémer el botó S1 de la placa emissora de dades quan tots els LEDs estan encesos, per enviar el valor de la temperatura al segon microcontrolador.

A vegades pot haver-hi problemes a l'hora de sincronitzar els dos microcontroladors i per tant, pot ser que necessitem realitzar més d'una vegada el procés d'apagar i encendre en el moment exacte els dos microcontroladors.

Per tal d'assegurar-nos de que no s'estan produint aquests problemes, podem connectar el conversor USB-TTL amb el pin GND a ground i el pin RXD al pin 26 del microcontrolador i així veure que s'estan transmetent dades i no s'estan produint errors (Fig. 66).

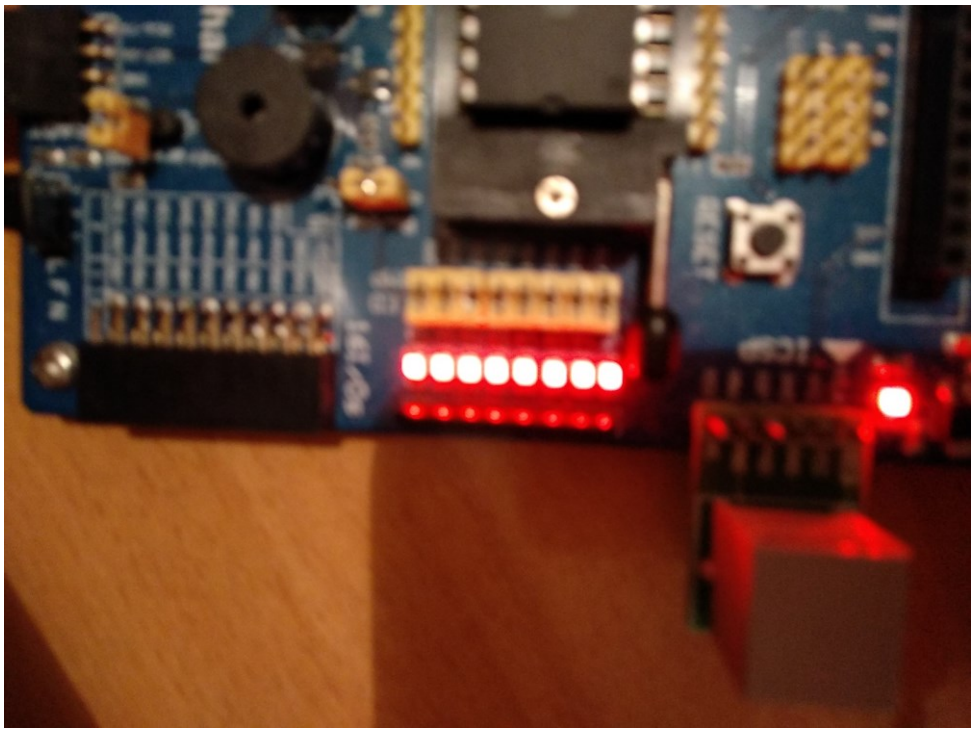


Fig. 65. Leds. Font: pròpia.

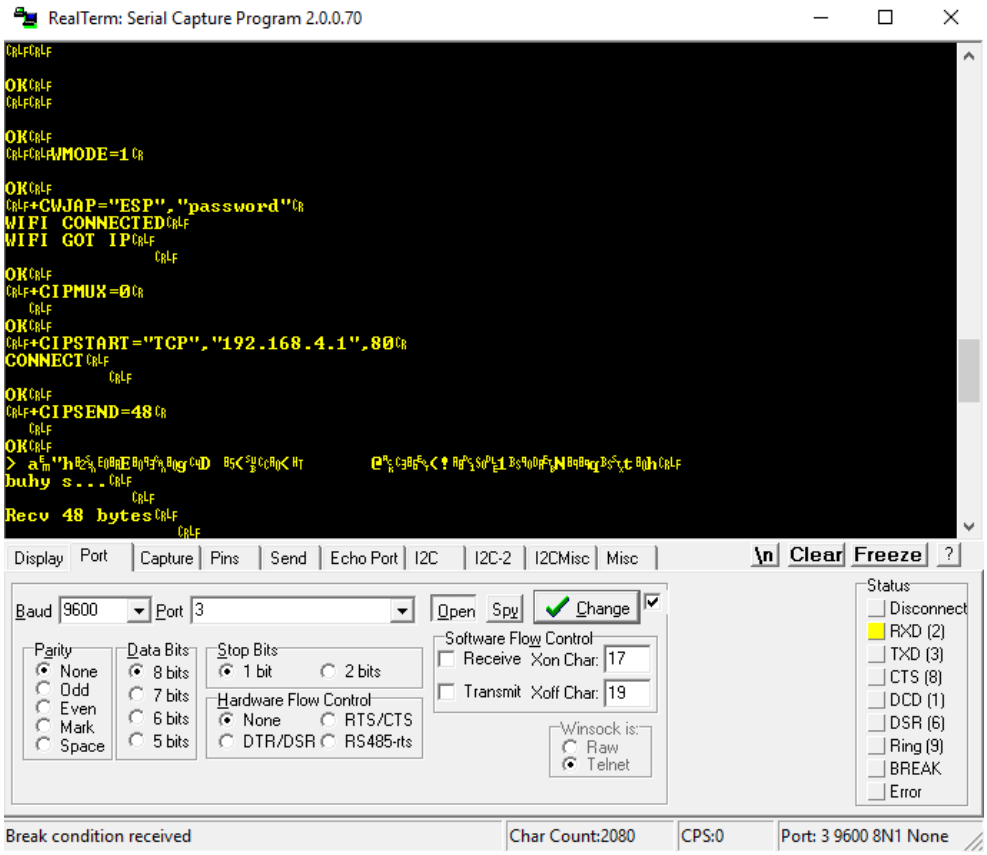


Fig. 66. Procés de sincronització entre els dos microcontroladors. Font: pròpia.

10. Pressupost

Com s'ha dit a la introducció del projecte, un dels objectius del projecte era mantenir els costos el més baixos possible. A continuació es presenten els resultats.

Donat que en un possible cas d'implementar aquest concepte en una aplicació "real" s'ha de separar el microcontrolador de la placa OpenPIC, per determinar el pressupost s'ha determinat que en lloc de tenir en compte el preu d'aquesta placa, s'utilitzarà una placa impresa, que amb els circuits impresos pot sortir per aproximadament 2 €.

A més, cal tenir en compte el preu del microcontrolador, que aproximadament podria ser de 1,5 €.

El preu del mòdul wifi és d'aproximadament 2 €, que igual que en els casos anteriors, dependrà d'el volum comprat, però un valor aproximat podria ser aquest.

Posteriorment, també s'ha utilitzat una pantalla LCD valorada en aproximadament 10 €, i els sensors de temperatura que es poden trobar per un preu aproximat de 1 € la unitat.

També serà necessària una bateria de 3,3 V, que té un preu aproximat de 3 € i un connector per poder carregar la bateria d'un cost aproximat de 0,5 €.

Per acabar es necessiten LEDs (0,1 €/per placa), i una carcassa amb un cost aproximat de 0,5 €.

10.1. Emissor de dades a una base de dades

Cost	
Placa	2 €
Microcontrolador	1,5 €
Modul Wifi	2 €
Bateria	3 €
Connector	0,5 €
Leds	0,1 €
Carcaça	0,5 €
Sensor	1 €
Total	**10,60 €**

Fig. 67. Cost unitari cas 1. Font: pròpia.

10.2. Microcontrolador receptor de dades + emissor

Cost	
Placa	2 x 2 €
Microcontrolador	2 x 1,5 €
Modul Wifi	2 x 2 €
Bateria	2 x 3 €
Connector	2 x 0,5 €
Leds	2 x 0,1 €
Carcaça	2 x 0,5 €
Sensor	1 €
LCD	10 €
Total	**30,10 €**

Fig. 68. Cost unitari cas 2. Font: pròpia.

10.3. Programació

El cost de programació és el més elevat, ja que s'ha hagut que dedicar moltes hores al programa (provablement en un cas real no serien necessàries tantes, ja que un projecte com aquest se li donaria a una persona amb més experiència), a més d'utilitzar tant el programador com la placa OpenPIC18F4520, que tenen un cost elevat.

Cost	
Hores de programació	400 x 10 €
OpenPIC	2 x 50 €
USB-TTL	2 €
ICD 3	100 €
Cables	0,5 €
Total	**4202,50 €**

Fig. 69. Cost de programació. Font: pròpia.

11. Declaració d'impacte ambiental

Com qualsevol altre producte creat a l'actualitat, la producció d'aquest requereix tant durant l'etapa de disseny com les etapes de fabricació, us i fi de la vida útil un cert consum de recursos, i per tant, aquest deixa una certa petjada ambiental.

Per determinar la petjada ambiental d'aquest producte es realitza un petit estudi del cicle de vida del producte.

Per començar, durant l'etapa de disseny, el major recurs consumit és l'energia elèctrica utilitzada per fer funcionar l'ordinador, durant el procés de disseny del programa, la qual s'estima a 100 W x 400 h= 40 kWh, i segons el factor d'emissió publicat per la generalitat de catalunya, es pot afirmar que aquest consum implica unes emissions de 0,392 kg CO_2/kWh x 40 kWh = 15,68 kg CO_2.

Per a l'etapa de fabricació del producte no es tenen valors concrets, ja que es tracta d'un prototip, i per tant, el possible procés de fabricació del producte no s'ha definit. Tot i això, donat que guarda forces similituds amb la fabricació d'un telèfon mòbil, a més de que per aquests productes es troba força informació relacionada amb la petjada ambiental, ja que les empreses estan obligats a publicar-la, s'ha assumit que per determinar un valor aproximat de la petjada ambiental del producte, considerar un consum de recursos semblant al d'un telèfon mòbil.

Els dos telèfons mòbils que s'han seleccionat per realitzar la comparació són l'Iphone 8 i el Sony Xperia T.

Per l'Iphone 8, s'estimen unes emissions de CO_2 equivalents de 47,31 kg [23], i pel Sony Xperia T s'estimen unes emissions de 44 kg de CO_2 [24]. Per tant, un valor aproximat de les emissions del procés de fabricació de cada placa amb els diferents components que les componen podria ser de 45 kg de CO_2.

Durant el funcionament de la placa, s'ha mesurat un consum de 190 mA aproximadament, i multiplicat per els 5 V als que treballa,s'obté un consum de potència de 0,95 W.

Estimant una vida útil de 2 anys x 365 dies x 24 hores, l'energia consumida total durant l'etapa de funcionament és d'aproximadament 4,38 kWh multiplicada pel factor d'emissions anterior (0,392 Kg CO_2/kWh), s'obtenen unes emissions de 6,52 kg de CO_2.

Per acabar, es torna a estimar el valor de les emissions creades durant el procés de destrucció del producte mitjançant la comparació amb els telèfons mòbils anteriors, i s'arriba a la conclusió de que aquest valor és d'1 kg de CO_2 aproximadament.

Amb tot això, s'arriba a la conclusió de que durant el procés de disseny s'han emés aproximadament 15,68 kg de CO_2 i per cada placa amb el mòdul wifi incorporat fabricada, s'estimen unes emissions de 52,52 kg de CO_2.

En quant a altres elements nocius que es poguessin crear durant el procés de fabricació d'aquest, s'ha de dir que tots els elements utilitzats per la realització del projecte, són productes comercialitzats per altres empreses, i per tant, s'entén que han hagut de passar una sèrie de proves abans de la seva sortida al mercat, i se sobreentén que no provoquen efectes nocius als usuaris finals.

En quant a la petjada ecològica del producte, s'ha de dir que tenint en compte el model energètic actual a l'estat espanyol, les emissions de CO_2 són inevitables, i per tant, tot i que es podria treballar en un programa que consumis menys energia, o utilitzar elements que consumeixin menys, el problema de les emissions seguiria existint.

A més, com passa amb tots els elements electrònics, un cop acabada la vida útil del producte, no es reciclen, tot i que tècnicament sí que es poden reciclar, però el cost de recuperar les peites quantitats d'elements reciclables que hi ha a les plaques impreses fa que no sigui rentable el procés.

En resum, el producte en si té una certa petjada ecològica negativa, però la seva implementació en un procés industrial per tal d'optimitzar un procés podria implicar una reducció del consum de recursos d'aquest, i per tant, al final es podria donar el cas de que es tingues un impacte positiu sobre el medi ambient.

12. Conclusions

La principal conclusió a la qual es pot arribar és que s'han complert els objectius marcats al principi de programar un microcontrolador i aconseguir que realitzi allò en el que es basa l'IoT, a més, s'ha aconseguit realitzar-ho a un cost raonable, i per tant, també s'ha complert el segon objectiu.

Seguint amb l'objectiu del cost, cal dir, que s'ha de tenir en compte un possible cost d'ensamblatge de cada element, i per tant és probable que el cost de la placa augmenti, tot i això, també és cert que els costos dels materials utilitzats està força inflat, pel fet de que només s'han comprat lots petits, i per tant, si això es volgués produir a un nivell industrial, és probable que el cost dels materials caigues dràsticament, a més que si s'automatitza el procés de muntatge, podria arribar a ser possible un cost del conjunt de uns 5 €.

També s'ha de tenir en compte que els possibles avantatges que pot donar la implementació del IoT en una planta industrial compensen de forma força important la inversió en les plàques (sempre i quan es realitzi de forma correcta).

Per acabar, cal dir que en certa forma pot semblar força "inútil" el fet de mesurar una temperatura i guardar-la en una base de dades, però l'objectiu principal del projecte era obtenir els coneixements necessaris per implementar una aplicació relacionada amb l'IoT, ja que el principi en el que es basen aquestes aplicacions és força semblant sempre (lectura – tractament – emissió – recepció – tractament – enmagatzematge) i per tant, en general ha sigut un projecte força interessant tant de dur a terme com en els objectius assolits. A més, donada la naturalesa del projecte, podria arribar a ser interesant la seva possible aplicació amb finalitats didàctiques en assignatures com Ampliació a l'Electrònica del màster d'Enginyeria Industrial de l'ETSEIB, ja que en aquesta assignatura es realitza un projecte en el que s'utilitza un sensor de temperatura i un microcontrolador per determinar el moment en el que s'ha d'encendre un ventilador i evitar el sobreescalfament d'una resistència, i si a més s'introduis el mòdul wifi programat en aquest projecte, es podria veure en una ordinador (mitjansant el programa KST) el valor de la temperatura mesurada per el sensor.

13. Annexos

13.1. Annex 1 BitsConfig.h

```
// PIC18F4520 Configuration Bit Settings
// 'C' source line config statements
// CONFIG1H
#pragma config OSC = INTIO7     // Oscillator Selection bits (Internal oscillator block, CLKO function on RA6, port function on RA7)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)
// CONFIG2L
#pragma config PWRT = OFF       // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled))
#pragma config BORV = 3         // Brown Out Reset Voltage bits (Minimum setting)
// CONFIG2H
#pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits (1:32768)
// CONFIG3H
#pragma config CCP2MX = PORTC   // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF     // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
#pragma config LPT1OSC = OFF    // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)
#pragma config MCLRE = ON       // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)
// CONFIG4L
#pragma config STVREN = OFF     // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause Reset)
#pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))
```

```
// CONFIG5L
#pragma config CP0 = OFF        // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)
#pragma config CP1 = OFF        // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)
#pragma config CP2 = OFF        // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)
#pragma config CP3 = OFF        // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)
// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code-protected)
// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)
// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write-protected)
// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)
```

```
// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)
```

13.2. Annex 2 Functinit.h

```
/*Inicialització de les funcions que utilitza l'arxiu main.c*/
void sendstring(char*);
void start (void);
void configNetwork(char*,char*);
void recive(void);
void CIPSTART(void);
void CIPSEND(void);
void readtemp(void);
```

13.3. Annex 3 main.c

```
// Arxiu main.c
#include <xc.h>
#include "BitsConfig.h"
#include "functinit.h"
unsigned char valor_temperatura[9];
char f[100];
char *user="AndroidAPa";     /////Nom de la conexió a la que l'ESP es conectarà
char *pass="123qwert";       ////Contrasenya de la conexió a la que es conectarà l'ESP
char *link="192.168.43.99";  ////Direcció de la base de dades
char num='2';     /////Definir número del 1 al 9 en el cas de que més d'un microcontrolador
                  /////Es connecti a la base de dades
void main (void)
{
   OSCCON=0b11111100;   //configuració de la freqüència de l'oscil·lador 8 MHz
   TRISBbits.TRISB1=0;   //inicialització del pin "TRANSMITOK" que indica que l'ESP ha entes l'ordre donada
   T1CON=0b11101101;   //configuració del TIMER1
   unsigned char primercicle=1;
  //Reseteja l'ESP la primera vegada que es realitza el bucle i s'assegura que la transmissió de dades es realitza de forma correcte
   start();
```

```
    //Conexió a la xarxa wifi definida anteriorment
    configNetwork(user,pass);
    _delay(100000);
    while(1)  //Bucle que llegeix i envia la temperatura
    {   //Bit que indica que el programa funciona correctament
        LATBbits.LATB1=1;
        //Lectura temperatura amb el sensor
        readtemp();
        CIPSTART();
        //El primer cicle s'espera una mica més de temps del normal per permetre que
        //L'ESP es connecti a la base de dades (CIPSTART), una vegada connectat, ja no caldrà esperar tant cada vegada que es repeteixi el cicle.

        if (primercicle==1)
            {_delay(3000000);
             _delay(3000000);
             _delay(3000000);
             _delay(3000000);
             _delay(3000000);
             primercicle=0; }
        //Bit que indica que el programa funciona correctament
        LATBbits.LATB1=0;
        _delay(5000000);
        //Enviament de la dada mesurada a la base de dades(CIPEND)
        CIPSEND();
        _delay(500000);
    }
}
```

13.4. Annex 4 start.c

```
/*Espera l'inicialització de l'ESP, inicialitza l'uart*/

#include <xc.h>

#define RESETING LATBbits.LATB3


void startUART(void);
```

```
void recive(void);
void sendstring(char*);

void start (void)
{
   TRISBbits.TRISB2=0;
   TRISBbits.TRISB3=0;
   RESETING=1;
   _delay(5000000);
   _delay(5000000);
   _delay(500);
   startUART();
   _delay(3000000);
   sendstring("AT\r\n\0");
   recive();
   _delay(300000);
   sendstring("AT\r\n\0");
   recive();
   _delay(3000000);
   CREN=0;
   _delay(100);
   CREN=1;
   RESETING=0;
}

void startUART(void)
{
   TRISCbits.TRISC7=1;
   TRISCbits.TRISC6=1;
   SPBRGH=0b00000000;
   SPBRG=0b00001100;
   TXSTA=0b00100000;
   RCSTA=0b10010000;
   BAUDCON=0b00000010;
   _delay(500);
}
```

13.5. Annex 5 Sendstring.c

```
/*Escriptura de dades a l'UART*/

#include <xc.h>

void sendstring(char* strat){
    unsigned char i=0;
    unsigned char c;
    while(strat[i]!='\0')
    {
    c=strat[i];
    TXREG=c;
    while(!TXIF);
    i++;
    }
}
```

13.6. Annex 6 Recive.c

```
/* Lectua de l'UART*/
#include <xc.h>
#define READING LATBbits.LATB2


void recive(void)
{   TRISBbits.TRISB2=0;
    extern char f[100];
    READING=1;

    char datA;
    char i=0;
    unsigned long time=0;
    char IN=1;

    WRITETIMER1(0);
    while (IN)
    {
        if(RCIF)
```

```
            {
            datA=RCREG;
            f[i]=datA;
            f[i+1]='\0';
            i++;

            /*if(OERR)
               {
               datA=RCREG;
               //IN=0;
               }*/
            WRITETIMER1(0);
            }
         else
            {
            time=READTIMER1();
            if(time>30000)
               {
               IN=0;
               }
            }
      }
      READING=0;
}
```

13.7. Annex 7 Configgetwork.c

```
/*Realitza els diferents pasos per configurar la conexió*/

#include <xc.h>

void start (void);
unsigned char AT(void);
unsigned char CIPMUX(void);
unsigned char CWMODE(void);
unsigned char CWJAP(char*,char*);

unsigned char pas1=1;
unsigned char pas2=1;
unsigned char pas3=1;
```

```
unsigned char pas4=1;
unsigned char errorcount=3;
//unsigned char errorcount2=0;


void configNetwork(char *user1,char *pass1)
{

   while(pas1)
   {_delay(500);
    pas1=AT();
    _delay(500000);
   }
   while(pas2)
   {_delay(500);
   pas2=CWMODE();
   _delay(500000);
   }
   while(pas3)
   {_delay(50000);
    pas3=CIPMUX();
   _delay(5000000);
   _delay(5000000);
   }
   while(pas4&&errorcount)
   {
   _delay(500);
   pas4=CWJAP(user1,pass1);
   _delay(5000000);
   _delay(5000000);
   _delay(5000000);
   _delay(5000000);
   errorcount=errorcount-1;
   }
}
```

13.8. Annex 8 AT.c

```
/*Comprovació de que la comunicació entre la UART i l'ESP és correcte*/
#include <xc.h>
void sendstring(char*);
void recive (void);

unsigned char AT(void)
{
 extern char f[100];
 _delay(5000);
 sendstring("AT\r\n\0");
 recive();
 _delay(50);

 if (f[7]=='O'&&f[8]=='K')
 {return 0;}
 else
    {return 1;}
}
```

13.9. Annex 9 CWMODE.c

```
/*Configura l'ESP com a client*/
#include <xc.h>
void sendstring(char*);
void recive (void);

unsigned char CWMODE(void)
{ extern char f[100];
 _delay(5000);
 sendstring("AT+CWMODE=1\r\n\0");
 recive();

 if (f[16]=='O'&&f[17]=='K')
 {return 0;}
 else
    {return 1;}
}
```

13.10. Annex 10 CIPMUX.c

```
/*desactiva la possibilitat de que multiples clients es conectin al punt d'acces*/
#include <xc.h>
void sendstring(char*);
void recive (void);

unsigned char CIPMUX (void)
{
   extern char f[100];
   sendstring("AT+CIPMUX=0\r\n\0");
   recive();
   _delay(5000);
   return 0;
}
```

13.11. Annex 11 CWJAP.C

```
/*Donat un user i un pass, es conecta a la conexió triada, i s'espera fins que aquesta
s'hagui realitzat*/

#include <xc.h>
char z[100];
void sendstring(char*);
void recive (void);

unsigned char CWJAP(char *user, char *pass)
{
   extern char f[100];
   unsigned long t1=0;
   unsigned long t2=0;
   unsigned long t3=30000;
   unsigned char l=0;
  // _delay(5000);
   char *p1="AT+CWJAP=\"";
   _delay(500);
   for(unsigned char i=0;i<10;i++)
   {
      z[l]=p1[i];
      l++;}
```

```
   for(unsigned char i=0;user[i]!='\0';i++)
   {
      z[l]=user[i];
      l++;
   }
   char *p2="\",\"";
   for(unsigned char i=0;i<3;i++)
   {
      z[l]=p2[i];
      l++;
   }
   for(unsigned char i=0;pass[i]!='\0';i++)
   {
      z[l]=pass[i];
      l++;
   }
   char *p3="\"\r\n";
   for(unsigned char i=0;i<3;i++)
   {
      z[l]=p3[i];
      l++;
   }
   z[l]='\0';
_delay(500);
sendstring(z);
recive();
WRITETIMER1(0);

while(!RCIF&&t1)
{
    t2=READTIMER1();
    if(t2>t3)
    {t1=0;}
    _delay(500);
}
if(RCIF)
{recive();}
```

```
if (f[12]=='F'||f[13]=='A'||f[14]=='I'||f[15]=='L')
{return 1;}

 //LATDbits.LATD3=1;
WRITETIMER1(0);
t2=0;
t1=1;
while(!RCIF&&t1)
  { t2=READTIMER1();
     if(t2>t3)
       {t1=0;}
     _delay(50);}
  if(RCIF)
    {recive();
    t2=0;
    t1=1;}
  WRITETIMER1(0);
  while(!RCIF&&t1)
    { t2=READTIMER1();
     if(t2>t3)
        {t1=0;}
     _delay(50);}

  if(RCIF)
   {recive();
   t2=0;
   t1=1;}
  WRITETIMER1(0);
  while(!RCIF&&t1)
     {t2=READTIMER1();
        if(t2>t3)
           {t1=0;}
        _delay(50);}
  if(RCIF)
     {recive();
     t2=0;
     t1=1;}
   //LATDbits.LATD3=0;
  return 0;}
```

13.12. Annex 12 Readtemp.c

```
/*
 * File:   readtemp.c
 * Author: a
 *
 * Created on 9 / de novembre / 2018, 21:01
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

#define OW_TRIS TRISBbits.TRISB0    //Port mode register, 1=input,0=output
#define OW_PORT PORTBbits.RB0       //Pin connected to 1-wire bus
#defineHIGH  1
#defineLOW          0
#defineOUTPUT       0
#defineINPUT 1
#defineSET          1
#defineCLEAR        0

#define OW_PIN_DIRECTION        TRISBbits.TRISB0
#define OW_WRITE_PIN            LATBbits.LATB0
#define OW_READ_PIN                     PORTBbits.RB0

unsigned char scratchpad[9];
unsigned char a;
unsigned char sign;
unsigned char digit;
unsigned int  decimal;

void drive_OW_high (void);
void drive_OW_low (void);
unsigned char OW_reset_pulse(void);
unsigned char read_OW (void);
unsigned char OW_read_bit (void);
unsigned char OW_read_byte (void);
void OW_write_bit (unsigned char write_bit);
void OW_write_byte (unsigned char write_data);
```

```
unsigned char Detect_Slave_Device(void);
void DS18B20_convert_temperature(void);
unsigned char DS18B20_read_scratchpad(unsigned char *sp);
void DS18B20_decode_temp(unsigned char *sp);
unsigned char BusyDS18B20(void);
unsigned char calc_crc(unsigned char buff[], unsigned char num_vals);

void readtemp(void) {

   extern unsigned char valor_temperatura[9];

   if(Detect_Slave_Device())
      {
       DS18B20_convert_temperature();

       while(BusyDS18B20())
          {Nop();}

       _delay(2500);
       a=DS18B20_read_scratchpad(scratchpad);
       DS18B20_decode_temp(scratchpad);
   /*Guardat del valor de la temperatura a la variable valor_temperatura en format ASCII
(string)*/
       valor_temperatura[0]=sign;
       valor_temperatura[1]=(digit/100)|0x30;
       valor_temperatura[2]=((digit%100)/10)|0x30;
       valor_temperatura[3]=((digit%10)%10)|0x30;
       valor_temperatura[4]='.';
       valor_temperatura[5]=(decimal/1000)|0x30;
       valor_temperatura[6]=((decimal%1000)/100)|0x30;
       valor_temperatura[7]=(((decimal%1000)%100)/10)|0x30;
       valor_temperatura[8]=(((decimal%1000)%100)%10)|0x30;
      }
   else
   {  valor_temperatura[0]='-';
      valor_temperatura[1]='-';
      valor_temperatura[2]='-';
      valor_temperatura[3]='-';
```

```
        valor_temperatura[4]='-';
        valor_temperatura[5]='-';
        valor_temperatura[6]='-';
        valor_temperatura[7]='-';
        valor_temperatura[8]='-';
    }
}
void drive_OW_high (void)
{
        OW_WRITE_PIN = HIGH;
        OW_PIN_DIRECTION = OUTPUT;
}

void drive_OW_low (void)
{
        OW_WRITE_PIN = LOW;
        OW_PIN_DIRECTION = OUTPUT;
}

unsigned char OW_reset_pulse(void)
{
        unsigned char presence_detect;

        drive_OW_low();                         // Drive the bus low
        _delay(240);                            // Delay 480 microseconds
        _delay(240);
        _delay(240);                            // Delay 480 microseconds
        _delay(240);
        drive_OW_high ();                       // Release the bus
        _delay(70);                                     // Delay 70 microseconds
        _delay(70);                                     // Delay 70 microseconds
        presence_detect = read_OW();    // Sample for presence pulse from slave
        _delay(205);                            // Delay 410 microseconds
        _delay(205);
        _delay(205);                            // Delay 410 microseconds
        _delay(205);
        drive_OW_high ();                       // Release the bus
        return presence_detect;                 // Return the presence pulse
}
```

```
unsigned char read_OW (void)
{
        OW_WRITE_PIN = INPUT;

         if (HIGH == OW_READ_PIN)
         {
                return(SET);
         }
         else
         {
                return(CLEAR);
         }
}
unsigned char OW_read_bit (void)
{
        unsigned char read_data;
        //reading a bit
        drive_OW_low();                         // Drive the bus low
        _delay(12);                                     // delay 6 microseconds
        drive_OW_high ();                       // Release the bus
        Nop();                                          // delay 3 microseconds
        Nop();
        Nop();
        read_data = read_OW();                  // Read the status of OW_PIN
        _delay(110);                                    // Delay 55 microseconds
        return read_data;
}

unsigned char OW_read_byte (void)
{
        unsigned char loop, result=0;
        for (loop = 0; loop < 8; loop++)
        {
                result >>= 1;  // Shift the result to get it ready for the next bit to receive
                if (OW_read_bit())
                result |= 0x80;                 // If result is one, then set MS-bit
        }
        return result;
}
```

```
void OW_write_bit (unsigned char write_bit)
{
        if (write_bit)
        {
                //writing a bit '1'
                drive_OW_low();             // Drive the bus low
                _delay(12);                         // Delay 6 microseconds
                drive_OW_high ();           // Release the bus
                _delay(128);                        // Delay 64 microseconds
        }
        else
        {
                //writing a bit '0'
                drive_OW_low();             // Drive the bus low
                _delay(120);                        // Delay 60 microseconds
                drive_OW_high ();           // Release the bus
                _delay(120);                        // Delay 10 microseconds for recovery
        }
}

void OW_write_byte (unsigned char write_data)
{
        unsigned char loop;
        for (loop = 0; loop < 8; loop++)
        {
                OW_write_bit(write_data & 0x01);    // Sending LS-bit first
                write_data >>= 1;           // Shift the data byte for the next bit to send
      _delay(100);
        }
}

/*********************************************************************
* Function:       unsigned char Detect_Slave_Device(void)
* PreCondition:    None
* Input:            None
* Output:                 0 - Not Present   1 - Present
* Overview:       Check the presence of slave device.
*********************************************************************/
```

```
unsigned char Detect_Slave_Device(void)
{
	if (!OW_reset_pulse())
		return 1;
	else
		return 0;
}

/*********************************************************************
* Function:	void DS18B20_convert_temperature(void)
* PreCondition:	None
* Input:		None
* Output:		None
* Overview:	Send a command to the DS18B20 to convert temperature.
*********************************************************************/
void DS18B20_convert_temperature(void)
{
	OW_reset_pulse();		// Reset pulse
	OW_write_byte(0xCC);		// Skip ROM command
	OW_write_byte(0x44);// Send a command to convert temperature
}

/*********************************************************************
* Function:	unsigned char DS18B20_read_scratchpad(unsigned char *sp)
* PreCondition:	None
* Input:		None
* Output:		pointer to a buffer containing data from device
*			This function return 1 if CRC is OK, 0 otherwise
* Overview:	Read scratchpad from DS18B20 device.
*********************************************************************/
unsigned char DS18B20_read_scratchpad(unsigned char *sp)
{
	unsigned char i,crc;
	// Sequence to read scratchpad
	OW_reset_pulse();
	OW_write_byte(0xCC);
	OW_write_byte(0xBE);
	for(i=0;i<9;i++)
	{
```

```
            sp[i] = OW_read_byte();
        }
        // Check CRC
        crc = calc_crc(sp,8);
        if(crc == sp[8])                    // CRC OK from scratchpad
        {
            return(1);
        }
        else
        {
            return(0);                      // CRC wrong from scratchpad
        }
}


/*********************************************************************
* Function:       void DS18B20_decode_temp(unsigned char *sp, char *buffer)
* PreCondition:    None
* Input:            pointer to data (sp) containing raw data
* Output:                 pointer to a buffer containing string with converted temperature
* Overview:       Decode temperature into string (degrees Celsius) from raw data read from
DS18B20 device.
*********************************************************************/
void DS18B20_decode_temp(unsigned char *sp)
{

    unsigned int temperature;

    temperature = ((unsigned int)sp[1])<<8 | sp[0];
    if((sp[1] & 0x80) == 0x80)
    {
        sign = '-';
        temperature = (temperature ^ 0xFFFF) + 1;
    }
    else
    {
        sign = ' ';
    }
    digit = (temperature >> 4) & 0x7F;
```

```
        decimal = (temperature & 0x0F) * 625;
    _delay(500);
}

/*************************************************************************
* Function:       unsigned char BusyDS18B20(void)
* PreCondition:    None
* Input:            None
* Output:                 This function return 1 if device is busy, 0 otherwise
* Overview:       Check status(busy/idle) of the device.
*************************************************************************/
unsigned char BusyDS18B20(void)
{
        return(!OW_read_bit());
}

/*************************************************************************
* Function:       unsigned char calc_crc(unsigned char buff[], unsigned char num_vals)
* PreCondition:    None
* Input:            pointer to data buffer and number of data values
* Output:                 Calculated CRC
* Overview:       Compute CRC from data.
*************************************************************************/
unsigned char calc_crc(unsigned char buff[], unsigned char num_vals)
{
  unsigned char shift_reg=0, data_bit, sr_lsb, fb_bit, i, j;

  for (i=0; i<num_vals; i++) /* for each byte */
  {for(j=0; j<8; j++)   /* for each bit */
    {
      data_bit = (buff[i]>>j)&0x01;
      sr_lsb = shift_reg & 0x01;
      fb_bit = (data_bit ^ sr_lsb) & 0x01;
      shift_reg = shift_reg >> 1;
      if (fb_bit)
      {
        shift_reg = shift_reg ^ 0x8c;
      }}}
  return(shift_reg);}
```

13.13. Annex 13 CIPSTART.c

```
/*Inicialitza la comunicació entre el punt d'accés i el client*/

#include <xc.h>

char w[100];

void sendstring(char*);
void recive (void);

void CIPSTART(void)
{
    extern char f[100];
    extern char *link;
    unsigned char u=0;
    unsigned char counter=4;
    unsigned char repeat=1;
    char *w1="AT+CIPSTART=\"TCP\",\"";
    char *w2="\",80\r\n";

   while (repeat&&counter){

   for(unsigned char i1=0;i1<19;i1++)
   {   repeat=0;
       w[u]=w1[i1];
       u++;
   }

   for(unsigned char i2=0;link[i2]!='\0';i2++)
   {
       w[u]=link[i2];
       u++;
   }

   for(unsigned char i3=0;i3<6;i3++)
   {
       w[u]=w2[i3];
       u++;
   }
```

```
 w[u]='\0';
 _delay(500);
 sendstring(w);
 recive();
 unsigned long t2=0;
 unsigned long t3=30000;
 unsigned char t1=1;
 if(f[39]=='A'||f[40]=='T')
 {t1=0;}
WRITETIMER1(0);
 while(!RCIF&&t1)
 {
  t2=READTIMER1();
  if(t2>t3)
     {t1=0;}
  _delay(50);
 }
 if(RCIF)
   {recive();}

if (f[2]=='E'||f[3]=='R'||f[4]=='R'||f[5]=='O'||f[6]=='R')
{repeat=1;
counter=counter-1;
 }
_delay(5000);
 }
 }
```

13.14. Annex 14 Cipsend.c

```
/*crea i envia el comandament l'ordre de enviar una dada al servidor, i crea i envia la dada*/

#include <xc.h>

void sendstring(char*);
void recive (void);
```

```
void CIPSEND (void)
{
   extern char f[100];
   extern unsigned char valor_temperatura[9];
   extern char *link;
   extern char num;
   char *data1="GET /read";
   char *data2=".php?val1=";
   char *data3=" HTTP/1.1\r\nHost: ";
   char *data4="\r\n\r\n\0";

   char data[100];
   char r[100];

   unsigned char i1=0;
   unsigned char i=0;
   for (i=0;data1[i]!='\0';i++)
      {
        data[i1]=data1[i];
        i1++;
      }
      data[i1]=num;
      i1++;
   for (i=0;data2[i]!='\0';i++)
      {
        data[i1]=data2[i];
        i1++;
      }
   for (i=0;i<9;i++)
      {
        data[i1]=valor_temperatura[i];
        i1++;
      }
   for (i=0;data3[i]!='\0';i++)
      {
        data[i1]=data3[i];
        i1++;
      }
   for (i=0;link[i]!='\0';i++)
```

```
    {
      data[i1]=link[i];
      i1++;
    }
for (i=0;data4[i]!='\0';i++)
    {
      data[i1]=data4[i];
      i1++;
    }
unsigned char j4=0;
char *r1="AT+CIPSEND=";

for(unsigned char j3=0;j3<11;j3++)
{
   r[j4]=r1[j3];
   j4++;
}
char j6[1];
unsigned char j5=1;
unsigned char j7=10;
unsigned char j9=1;

while(j5)
{
   j9=i1/10;
   j6[0]=i1/10;
   j6[0]=0b00110000|j6[0];
   _delay(500);
   if(j9<10)
   { if(j9==0)
      {
         j6[0]=i1%10;
         j6[0]=0b00110000|j6[0];
         r[j4]=j6[0];
       }
       else
        {
         r[j4]=j6[0];
        }
```

```
            j4++;
            i1=i1-j9*j7;
          }
          else
          {
             j7=j7*10;
          }

          j5=j9;
          _delay(500);
      }

      char *r2="\r\n";

      for(unsigned char j8=0;j8<2;j8++)
      {
          r[j4]=r2[j8];
          j4++;
      }
      r[j4]='\0';

      sendstring(r);
      recive();
     // _delay(2000000);
      _delay(30000);
      sendstring(data);
}
```

13.15. Annex 15 Read.php

```
<?php
$var1 = $_GET {'val1'};


$fileContent =" ".$var1." \n";

$fileStatus = file_put_contents('data.txt',$fileContent,FILE_APPEND);

if($fileStatus != false)
{
        //echo "OK";
}
else
{
        //echo "NOK";
}
?>
```

13.16. Annex 16 Definicions.h

```
//definició de les función utilitzades al main.c
void LCD_test(void);
void DisplayString(u8 *s,u8 x,u8 y,u8 Reverse);
void start (void);
void ESPconfig(void);
void CheckIP(void);
void wait_sensor_conection(void);
void recive(void);
void sendstring(char*);
```

13.17. Annex 17 Asciihex8x16.h

```
/*
*=====================================================================
==================================
*
* File              : asciihex8x16.h
* Hardware Environment:
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
*
*                          (c) Copyright 2011-2016, WaveShare
*                              http://www.waveShare.net
*                                All Rights Reserved
*
*=====================================================================
==================================
*/
const  unsigned char  ascii[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
          //0
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x18,0x3C,0x7E,0x7E,0x7E,0x3C,0x18,0x00,0x00,0x00,0x00,0x00,0x00,
0xFF,0xFF,0xFF,0xE7,0xC3,0x81,0x81,0x81,0xC3,0xE7,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x1F,0x05,0x05,0x09,0x09,0x10,0x10,0x38,0x44,0x44,0x44,0x38,0x00,0x00,0x00,
0x00,0x1C,0x22,0x22,0x22,0x1C,0x08,0x08,0x7F,0x08,0x08,0x08,0x08,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x03,0x1D,0x11,0x13,0x1D,0x11,0x11,0x11,0x13,0x17,0x36,0x70,0x60,0x00,0x00,0x00,
0x08,0x08,0x5D,0x22,0x22,0x22,0x63,0x22,0x22,0x22,0x5D,0x08,0x08,0x00,0x00,0x00,
          //0f
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0xFF,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
```

```
0x00,0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0x3F,0x1F,0x0F,0x07,0x03,0x01,0x00,0x00,
0x08,0x1C,0x2A,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x2A,0x1C,0x08,0x00,0x00,
0x00,0x24,0x24,0x24,0x24,0x24,0x24,0x24,0x24,0x24,0x00,0x00,0x24,0x24,0x00,0x00,
0x00,0x1F,0x25,0x45,0x45,0x45,0x25,0x1D,0x05,0x05,0x05,0x05,0x05,0x00,0x00,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0xF8,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x08,0x1C,0x2A,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x0F,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x00,0x00,0x00,0x00,0x00,0x04,0x02,0x7F,0x02,0x04,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x10,0x20,0x7F,0x20,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//1F
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,0x18,0x18,0x00,0x00,0x00,
0x12,0x36,0x24,0x48,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x24,0x24,0x24,0xFE,0x48,0x48,0x48,0xFE,0x48,0x48,0x48,0x00,0x00,0x00,
0x00,0x10,0x38,0x54,0x54,0x50,0x30,0x18,0x14,0x14,0x54,0x54,0x38,0x10,0x10,0x00,
0x00,0x00,0x44,0xA4,0xA8,0xA8,0xA8,0x54,0x1A,0x2A,0x2A,0x2A,0x44,0x00,0x00,0x00,
0x00,0x00,0x30,0x48,0x48,0x48,0x50,0x6E,0xA4,0x94,0x88,0x89,0x76,0x00,0x00,0x00,
0x60,0x60,0x20,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x02,0x04,0x08,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x08,0x08,0x04,0x02,0x00,0x00,
0x40,0x20,0x10,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x10,0x10,0x20,0x40,0x00,0x00,
0x00,0x00,0x00,0x10,0x10,0xD6,0x38,0x38,0xD6,0x10,0x10,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xFE,0x10,0x10,0x10,0x10,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x60,0x20,0xC0,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x60,0x00,0x00,0x00,
0x00,0x01,0x02,0x02,0x04,0x04,0x08,0x08,0x10,0x10,0x20,0x20,0x40,0x40,0x00,0x00,
0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x00,0x00,0x10,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x3C,0x42,0x42,0x42,0x04,0x04,0x08,0x10,0x20,0x42,0x7E,0x00,0x00,0x00,
0x00,0x00,0x3C,0x42,0x42,0x04,0x18,0x04,0x02,0x02,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0x04,0x0C,0x14,0x24,0x24,0x44,0x44,0x7E,0x04,0x04,0x1E,0x00,0x00,0x00,
0x00,0x00,0x7E,0x40,0x40,0x40,0x58,0x64,0x02,0x02,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0x1C,0x24,0x40,0x40,0x58,0x64,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
```

0x00,0x00,0x7E,0x44,0x44,0x08,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,0x00,
0x00,0x00,0x3C,0x42,0x42,0x42,0x24,0x18,0x24,0x42,0x42,0x42,0x3C,0x00,0x00,0x00,
0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x26,0x1A,0x02,0x02,0x24,0x38,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x10,0x20,0x00,
0x00,0x00,0x02,0x04,0x08,0x10,0x20,0x40,0x20,0x10,0x08,0x04,0x02,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0xFE,0x00,0x00,0x00,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x40,0x20,0x10,0x08,0x04,0x02,0x04,0x08,0x10,0x20,0x40,0x00,0x00,0x00,
0x00,0x00,0x3C,0x42,0x42,0x62,0x02,0x04,0x08,0x08,0x00,0x18,0x18,0x00,0x00,0x00,
0x00,0x00,0x38,0x44,0x5A,0xAA,0xAA,0xAA,0xAA,0xB4,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0x10,0x10,0x18,0x28,0x28,0x24,0x3C,0x44,0x42,0x42,0xE7,0x00,0x00,0x00,
0x00,0x00,0xF8,0x44,0x44,0x44,0x78,0x44,0x42,0x42,0x42,0x44,0xF8,0x00,0x00,0x00,
0x00,0x00,0x3E,0x42,0x42,0x80,0x80,0x80,0x80,0x80,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0xF8,0x44,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x44,0xF8,0x00,0x00,0x00,
0x00,0x00,0xFC,0x42,0x48,0x48,0x78,0x48,0x48,0x40,0x42,0x42,0xFC,0x00,0x00,0x00,
0x00,0x00,0xFC,0x42,0x48,0x48,0x78,0x48,0x48,0x40,0x40,0x40,0xE0,0x00,0x00,0x00,
0x00,0x00,0x3C,0x44,0x44,0x80,0x80,0x80,0x8E,0x84,0x44,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0xE7,0x42,0x42,0x42,0x42,0x7E,0x42,0x42,0x42,0x42,0xE7,0x00,0x00,0x00,
0x00,0x00,0x7C,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x3E,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x88,0xF0,0x00,
0x00,0x00,0xEE,0x44,0x48,0x50,0x70,0x50,0x48,0x48,0x44,0x44,0xEE,0x00,0x00,0x00,
0x00,0x00,0xE0,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x42,0xFE,0x00,0x00,0x00,
0x00,0x00,0xEE,0x6C,0x6C,0x6C,0x6C,0x54,0x54,0x54,0x54,0x54,0xD6,0x00,0x00,0x00,
0x00,0x00,0xC7,0x62,0x62,0x52,0x52,0x4A,0x4A,0x4A,0x46,0x46,0xE2,0x00,0x00,0x00,
0x00,0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0xFC,0x42,0x42,0x42,0x42,0x7C,0x40,0x40,0x40,0x40,0xE0,0x00,0x00,0x00,
0x00,0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0xB2,0xCA,0x4C,0x38,0x06,0x00,0x00,
0x00,0x00,0xFC,0x42,0x42,0x42,0x7C,0x48,0x48,0x44,0x44,0x42,0xE3,0x00,0x00,0x00,
0x00,0x00,0x3E,0x42,0x42,0x40,0x20,0x18,0x04,0x02,0x42,0x42,0x7C,0x00,0x00,0x00,
0x00,0x00,0xFE,0x92,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x38,0x00,0x00,0x00,
0x00,0x00,0xE7,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x3C,0x00,0x00,0x00,
0x00,0x00,0xE7,0x42,0x42,0x44,0x24,0x24,0x28,0x28,0x18,0x10,0x10,0x00,0x00,0x00,
0x00,0x00,0xD6,0x92,0x92,0x92,0x92,0xAA,0xAA,0x6C,0x44,0x44,0x44,0x00,0x00,0x00,
0x00,0x00,0xE7,0x42,0x24,0x24,0x18,0x18,0x18,0x24,0x24,0x42,0xE7,0x00,0x00,0x00,
0x00,0x00,0xEE,0x44,0x44,0x28,0x28,0x10,0x10,0x10,0x10,0x10,0x38,0x00,0x00,0x00,
0x00,0x00,0x7E,0x84,0x04,0x08,0x08,0x10,0x20,0x20,0x42,0x42,0xFC,0x00,0x00,0x00,
0x1E,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x1E,0x00,0x00,
0x00,0x40,0x40,0x20,0x20,0x10,0x10,0x10,0x08,0x08,0x04,0x04,0x04,0x02,0x02,0x00,
0x78,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x78,0x00,0x00,

```
0x1C,0x22,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,
0x60,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3C,0x42,0x1E,0x22,0x42,0x42,0x3F,0x00,0x00,0x00,
0x00,0x00,0xC0,0x40,0x40,0x40,0x58,0x64,0x42,0x42,0x42,0x64,0x58,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x22,0x40,0x40,0x40,0x22,0x1C,0x00,0x00,0x00,
0x00,0x00,0x06,0x02,0x02,0x02,0x1E,0x22,0x42,0x42,0x42,0x26,0x1B,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3C,0x42,0x7E,0x40,0x40,0x42,0x3C,0x00,0x00,0x00,
0x00,0x00,0x0F,0x11,0x10,0x10,0x7E,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3E,0x44,0x44,0x38,0x40,0x3C,0x42,0x42,0x3C,0x00,
0x00,0x00,0xC0,0x40,0x40,0x40,0x5C,0x62,0x42,0x42,0x42,0x42,0xE7,0x00,0x00,0x00,
0x00,0x00,0x30,0x30,0x00,0x00,0x70,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x0C,0x0C,0x00,0x00,0x1C,0x04,0x04,0x04,0x04,0x04,0x04,0x44,0x78,0x00,
0x00,0x00,0xC0,0x40,0x40,0x40,0x4E,0x48,0x50,0x68,0x48,0x44,0xEE,0x00,0x00,0x00,
0x00,0x00,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFE,0x49,0x49,0x49,0x49,0x49,0xED,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xDC,0x62,0x42,0x42,0x42,0x42,0xE7,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x42,0x42,0x3C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xD8,0x64,0x42,0x42,0x42,0x44,0x78,0x40,0xE0,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1E,0x22,0x42,0x42,0x42,0x22,0x1E,0x02,0x07,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xEE,0x32,0x20,0x20,0x20,0x20,0xF8,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3E,0x42,0x40,0x3C,0x02,0x42,0x7C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x10,0x10,0x7C,0x10,0x10,0x10,0x10,0x10,0x0C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC6,0x42,0x42,0x42,0x42,0x46,0x3B,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xE7,0x42,0x24,0x24,0x28,0x10,0x10,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xD7,0x92,0x92,0xAA,0xAA,0x44,0x44,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x6E,0x24,0x18,0x18,0x18,0x24,0x76,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xE7,0x42,0x24,0x24,0x28,0x18,0x10,0x10,0xE0,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x44,0x08,0x10,0x10,0x22,0x7E,0x00,0x00,0x00,
0x03,0x04,0x04,0x04,0x04,0x04,0x08,0x04,0x04,0x04,0x04,0x04,0x04,0x03,0x00,0x00,
0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x00,
0x60,0x10,0x10,0x10,0x10,0x10,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x60,0x00,0x00,
0x4C,0x43,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
/*
0x00,0x00,0x1F,0x23,0x41,0x40,0xFE,0x40,0xFE,0x40,0x40,0x22,0x1C,0x00,0x00,0x00,
        //128
0x7F,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x60,0x20,0x20,0x00,0x00,
```

```
0x00,0x07,0x04,0x04,0x08,0x0C,0x08,0x08,0x18,0x10,0x10,0x10,0x20,0x20,0xC0,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x6E,0x22,0x44,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x63,0x63,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x18,0x18,0x10,0x7E,0x10,0x18,0x18,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,
0x00,0x18,0x18,0x76,0x7E,0x18,0x10,0x10,0x10,0x10,0x18,0x76,0x5A,0x18,0x18,0x00,
0x00,0x30,0x50,0x88,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x30,0x48,0x48,0x48,0x49,0x32,0x04,0x08,0x11,0x22,0x42,0x82,0x02,0x01,0x00,0x00,
0x3C,0x00,0x3C,0x42,0x40,0x40,0x30,0x0C,0x06,0x02,0x02,0x42,0x7C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x10,0x20,0x60,0x60,0x20,0x10,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x3E,0x4A,0x48,0x48,0x4A,0x4E,0x48,0x48,0x48,0x49,0x3F,0x00,0x00,0x00,
0x7F,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,
0x14,0x08,0x00,0x7F,0x01,0x02,0x04,0x08,0x08,0x10,0x20,0x40,0x7F,0x00,0x00,0x00,
0x7F,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,
0x7F,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x60,0x60,0x20,0x40,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x01,0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x01,0x01,0x00,0x01,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x3C,0x3C,0x3C,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x60,0xB8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0xE9,0x4B,0x4B,0x5D,0x55,0x5F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x88,0x70,0x20,0x30,0x48,0x40,0x60,0x38,0x0C,0x4C,0x78,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x20,0x30,0x10,0x20,0x20,0x40,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x02,0x7D,0x49,0x4F,0x48,0x48,0x4D,0x36,0x00,0x00,0x00,0x00,
0x7F,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x28,0x10,0x00,0x7C,0x04,0x08,0x10,0x20,0x40,0x7C,0x00,0x00,0x00,
0x24,0x00,0x42,0x44,0x24,0x28,0x18,0x18,0x10,0x10,0x10,0x10,0x3C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x60,0x90,0x90,0x60,0x00,0x0E,0x11,0x20,0x20,0x20,0x20,0x20,0x11,0x0E,0x00,   //
0x00,0x60,0x90,0x90,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   //
0x00,0x00,0x00,0x00,0x20,0x20,0x00,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x00,   //
0x00,0x04,0x04,0x04,0x18,0x2E,0x4E,0x50,0x50,0x50,0x22,0x3C,0x40,0x40,0x40,0x00,
0x00,0x00,0x1E,0x22,0x20,0x20,0x20,0x38,0x20,0x20,0x21,0xFA,0x4C,0x00,0x00,0x00,
0x40,0x27,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x27,0x40,0x00,0x00,
0x00,0x00,0x62,0x24,0x24,0x14,0x18,0xFF,0x08,0xFF,0x08,0x08,0x3C,0x00,0x00,0x00,
0x00,0x40,0x40,0x40,0x40,0x40,0x40,0x00,0x00,0x40,0x40,0x40,0x40,0x40,0x40,0x00,
0x08,0x08,0x04,0x02,0x03,0x04,0x08,0x08,0x05,0x02,0x01,0x04,0x08,0x08,0x07,0x00,
```

0x00,0x00,0x0C,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x7E,0xA6,0xC1,0xC1,0xC1,0xA6,0x5A,0x24,0x00,0x00,0x00,0x00,0x00,
0x00,0xE0,0x90,0xF0,0x90,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x11,0x22,0x24,0x44,0x4C,0x24,0x22,0x12,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0xFF,0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x78,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x7E,0xAA,0xA5,0xB9,0xA9,0xA6,0x42,0x3C,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x30,0x48,0x48,0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x01,0x01,0x01,0x0F,0x01,0x01,0x01,0x00,0x0F,0x00,0x00,0x00,0x00,0x00,
0x00,0x70,0x90,0x10,0x20,0x50,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x70,0x10,0x20,0x10,0x10,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x10,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x22,0x22,0x22,0x22,0x22,0x2F,0x50,0x40,0x40,0x00,0x00,0x00,
0x00,0x3E,0xF4,0xF4,0xF4,0xF4,0x74,0x14,0x14,0x14,0x14,0x14,0x14,0x14,0x00,0x00,
0x00,0x00,0x00,0x00,0x03,0x07,0x07,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x10,0x60,0x00,
0x00,0x20,0x20,0x20,0x20,0x20,0x70,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x70,0x98,0x88,0x50,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x88,0x48,0x24,0x22,0x32,0x24,0x44,0x88,0x00,0x00,0x00,0x00,
0x00,0x42,0x44,0x44,0x48,0x48,0x50,0x12,0x26,0x26,0x27,0x40,0x00,0x00,0x00,0x00,
0x00,0x44,0x44,0x44,0x48,0x48,0x50,0x17,0x21,0x22,0x44,0x47,0x00,0x00,0x00,0x00,
0x00,0x62,0x24,0x44,0x28,0x28,0x50,0x12,0x16,0x26,0x27,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x10,0x10,0x00,0x00,0x10,0x20,0x20,0x44,0x44,0x3C,0x00,0x00,0x00,
0x10,0x00,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x10,0x20,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x28,0x00,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x0C,0x00,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x3C,0x00,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x28,0x18,0x10,0x18,0x28,0x28,0x28,0x24,0x3C,0x44,0x44,0x42,0xC7,0x00,0x00,0x00,
0x00,0x00,0x1E,0x18,0x28,0x28,0x2A,0x4E,0x78,0x48,0x48,0x89,0xDE,0x00,0x00,0x00,
0x00,0x00,0x1C,0x23,0x41,0x40,0x40,0x40,0x40,0x40,0x61,0x32,0x0C,0x08,0x08,0x00,
0x10,0x10,0x6E,0x42,0x40,0x44,0x44,0x7C,0x44,0x40,0x40,0x42,0xFE,0x00,0x00,0x00,
0x08,0x10,0x6E,0x42,0x40,0x44,0x44,0x7C,0x44,0x40,0x40,0x42,0xFE,0x00,0x00,0x00,
0x28,0x44,0x7E,0x42,0x40,0x44,0x44,0x7C,0x44,0x40,0x40,0x42,0xFE,0x00,0x00,0x00,
0x2C,0x00,0x7E,0x42,0x40,0x44,0x44,0x7C,0x44,0x40,0x40,0x42,0xFE,0x00,0x00,0x00,
0x10,0x10,0x38,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x08,0x10,0x38,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x28,0x00,0x38,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,

0x2C,0x00,0x38,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,0x00,
0x00,0x00,0x3C,0x22,0x22,0x21,0x21,0x31,0x21,0x23,0x22,0x26,0x78,0x00,0x00,0x00,
0x0C,0x00,0x42,0x62,0x62,0x52,0x52,0x4A,0x4A,0x46,0x46,0x42,0xE2,0x00,0x00,0x00,
0x10,0x10,0x2C,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x10,0x10,0x2C,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x28,0x44,0x3C,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x0C,0x00,0x3C,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x24,0x00,0x3C,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,0x00,
0x00,0x00,0x10,0x08,0x04,0x02,0x01,0x02,0x04,0x08,0x10,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x3C,0x42,0x46,0x4A,0x4A,0x53,0x52,0x62,0x62,0x44,0x38,0x00,0x00,0x00,
0x10,0x00,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x08,0x10,0x62,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x28,0x00,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x24,0x00,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x08,0x10,0x62,0x44,0x24,0x28,0x18,0x18,0x10,0x10,0x10,0x10,0x3C,0x00,0x00,0x00,
0x00,0x00,0x20,0x20,0x38,0x26,0x21,0x21,0x21,0x26,0x38,0x20,0x70,0x00,0x00,0x00,
0x00,0x00,0x1C,0x22,0x22,0x22,0x2C,0x22,0x22,0x22,0x22,0x3A,0x6C,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x38,0x44,0x44,0x44,0x44,0x44,0x3A,0x00,0x00,0x00,
0x00,0x00,0x00,0x08,0x18,0x00,0x38,0x44,0x44,0x44,0x44,0x44,0x3A,0x00,0x00,0x00,
0x00,0x30,0x28,0x44,0x00,0x00,0x7C,0x44,0x1C,0x64,0x44,0x45,0x3A,0x00,0x00,0x00,
0x00,0x20,0x5C,0x00,0x00,0x00,0x7C,0x44,0x1C,0x64,0x44,0x45,0x3A,0x00,0x00,0x00,
0x00,0x28,0x6C,0x00,0x00,0x00,0x7C,0x44,0x1C,0x64,0x44,0x45,0x3A,0x00,0x00,0x00,
0x00,0x38,0x28,0x10,0x00,0x00,0x7C,0x44,0x1C,0x64,0x44,0x45,0x3A,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x66,0x99,0x91,0x3F,0x50,0x90,0x99,0x6E,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x18,0x64,0x44,0x40,0x40,0x44,0x44,0x38,0x20,0x10,0x60,0x00,
0x00,0x00,0x00,0x10,0x10,0x00,0x3C,0x42,0x42,0x7C,0x40,0x22,0x1C,0x00,0x00,0x00,
0x00,0x00,0x04,0x08,0x08,0x00,0x3C,0x42,0x42,0x7C,0x40,0x22,0x1C,0x00,0x00,0x00,
0x00,0x00,0x18,0x38,0x44,0x00,0x3C,0x42,0x42,0x7C,0x40,0x22,0x1C,0x00,0x00,0x00,
0x00,0x00,0x2C,0x24,0x00,0x00,0x3C,0x42,0x42,0x7C,0x40,0x22,0x1C,0x00,0x00,0x00,
0x00,0x00,0x20,0x20,0x10,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x38,0x00,0x00,0x00,
0x00,0x00,0x08,0x10,0x10,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x38,0x00,0x00,0x00,
0x00,0x10,0x28,0x48,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x3C,0x00,0x00,0x00,
0x00,0x28,0x6C,0x00,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x3C,0x00,0x00,0x00,
0x00,0x00,0x24,0x18,0x28,0x04,0x3E,0x42,0x42,0x42,0x42,0x42,0x3C,0x00,0x00,0x00,
0x00,0x30,0x1C,0x00,0x00,0x00,0x7C,0x42,0x42,0x42,0x42,0x42,0xE6,0x00,0x00,0x00,
0x00,0x00,0x00,0x10,0x00,0x00,0x3C,0x42,0x42,0x42,0x42,0x66,0x3C,0x00,0x00,0x00,
0x00,0x00,0x00,0x0C,0x08,0x00,0x3C,0x42,0x42,0x42,0x42,0x66,0x18,0x00,0x00,0x00,
0x00,0x18,0x28,0x04,0x00,0x00,0x3C,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x30,0x1C,0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,

```
0x00,0x24,0x2C,0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x42,0x44,0x38,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x01,0x00,0x1F,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x3E,0x46,0x4A,0x52,0x62,0x44,0xB8,0x00,0x00,0x00,
0x00,0x00,0x00,0x10,0x00,0xC4,0x44,0x44,0x44,0x44,0x44,0x26,0x3A,0x00,0x00,0x00,
0x00,0x00,0x04,0x08,0x18,0xC4,0x44,0x44,0x44,0x44,0x44,0x26,0x3A,0x00,0x00,0x00,
0x00,0x18,0x28,0x44,0x00,0x02,0x42,0x42,0x42,0x42,0x42,0x46,0x38,0x00,0x00,0x00,
0x00,0x00,0x00,0x24,0x24,0xC4,0x44,0x44,0x44,0x44,0x44,0x26,0x3A,0x00,0x00,0x00,
0x00,0x0C,0x08,0x10,0x00,0x00,0x24,0x24,0x24,0x18,0x18,0x10,0x10,0x10,0x60,0x00,
0x00,0xC0,0x40,0x40,0x40,0x5E,0x22,0x42,0x42,0x22,0x22,0x54,0x48,0x40,0x60,0x00,*/
};
```

13.18. Annex 18 clk.h

```
/*
*====================================================================
==================================
*
* File              : clk.h
* Hardware Environment:
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
*
*                          (c) Copyright 2011-2016, WaveShare
*                              http://www.waveShare.net
*                                All Rights Reserved
*
*====================================================================
==================================
*/
#ifndef _CLK_H
#define _CLK_H
#include<pic18.h>
#include "lcd_22.h"
/*
#if defined HSI_16M
void CLK_Init(void)
{
        CLK_SWR=0xe1;
```

```
        CLK_CKDIVR=0x00;
        CLK_SWCR|=0x02;     //¿ªÆôÇÐ»»
        while((CLK_SWCR & 0x01)==0x01);
        CLK_SWCR&=(~0x02);     //¹Ø±ÕÇÐ»»
}
#elif defined HSE
void CLK_Init(void)
{
        //ÆôÓÃÍâ²¿¸ßËÙ¾§ÕñÇÒ0·ÖÆµ
        CLK_SWR=0xb4;
        CLK_CKDIVR=0x00;
        CLK_SWCR|=0x02;     //¿ªÆôÇÐ»»
        while((CLK_SWCR & 0x01)==0x01);
        CLK_SWCR&=(~0x02);     //¹Ø±ÕÇÐ»»
}
#elif defined HSI_2M

#else
        #error has not define Crystal
#endif*/

#endif /*clk_h*/
```

13.19. Annex 19 clk.h

```
/*
*====================================================================
==================================
*
* File              : lcd_22.h
* Hardware Environment:
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
*
*                          (c) Copyright 2011-2016, WaveShare
*                              http://www.waveShare.net
*                                All Rights Reserved
*
```

```
*=====================================================================
====================================
*/
#ifndef LCD_22_H
#define LCD_22_H

#include<xc.h>
#include "system.h"

#define LCD22_SPI
//#define LCD22_PAN
#define  PENIRQ                          RA0   //IN

#define en_touch()                       RA4=0;// T_CS
#define dis_touch()              RA4=1;

#define reset_clr                        RD0=0;//RST
#define reset_set                        RD0=1;//

#define en_lcd()                          RA1=0;// CS
#define dis_lcd()                         RA1=1;//

#define en_lcd_data()     RA2=1;// RS
#define en_lcd_index()              RA2=0;//



#ifdef   LCD22_PAN
#define LCD_DATA     PORTB//PB_ODR
//#define lcd_wr_clk()           {TRISA=0x10;RA3=0;RA3=1;}//hou sbi(PC_ODR,2);
#endif

#define SAMP_COUNT                        5
#define SAMP_THRESHOLD              5

#define TOUCH_CMD_X              0xD0
#define TOUCH_CMD_Y              0x90

#define COLOR_YELLOW             0xFFE0
```

```
#define COLOR_BLACK             0x0000
#define COLOR_WHITE             0xFFFF
#define COLOR_INIT              COLOR_YELLOW

#define DOT_WIDTH 4

typedef struct xy
{
        u16 x;
        u16 y;
}xy_t;

#define TOUCH_MAX_CACHE 8
volatile xy_t touch_xy_buffer[TOUCH_MAX_CACHE];
volatile u8 touch_wr_index;
volatile u8 touch_rd_index;
volatile u8 touch_counter;
/**/
void init_lcd_spi(void);
void init_touch_spi(void);
void post_data(u16 data);
void post_cmd(u16 index, u16 cmd);
/**/
u16 get_touch_data(u16 cmd);
xy_t get_touch_xy(void);
u8 get_point_xy(void);
u8 draw_lcd(void);
void init_lcd(void);

#endif
```

13.20. Annex System.h

```
/*****************************************************************************************************
*
*
* File              : system.h
* Hardware Environment:
* Build Environment   : ST Visual Develop 4.1.6
```

```
* Version            : V1.0
* By                 : Xiao xian hui
*
*                              (c) Copyright 2005-2010, WaveShare
*                                  http://www.waveShare.net
*                                    All Rights Reserved
*
*******************************************************************************************************
*/

#ifndef _SYSTEM_H
#define _SYSTEM_H

typedef signed long  s32;
typedef signed short s16;
typedef signed char  s8;

typedef signed long  const sc32;  /* Read Only */
typedef signed short const sc16;  /* Read Only */
typedef signed char  const sc8;   /* Read Only */

typedef volatile signed long  vs32;
typedef volatile signed short vs16;
typedef volatile signed char  vs8;

typedef volatile signed long  const vsc32;  /* Read Only */
typedef volatile signed short const vsc16;  /* Read Only */
typedef volatile signed char  const vsc8;   /* Read Only */

typedef unsigned long  u32;
typedef unsigned short u16;
typedef unsigned char  u8;

typedef unsigned long  const uc32;  /* Read Only */
typedef unsigned short const uc16;  /* Read Only */
typedef unsigned char  const uc8;   /* Read Only */

typedef volatile unsigned long  vu32;
typedef volatile unsigned short vu16;
```

```
typedef volatile unsigned char  vu8;

typedef volatile unsigned long  const vuc32;  /* Read Only */
typedef volatile unsigned short const vuc16;  /* Read Only */
typedef volatile unsigned char  const vuc8;   /* Read Only */

#define LCD_DELAY() _asm("nop");_asm("nop");_asm("nop");_asm("nop");_asm("nop");

/* io configs */
#define sbi(io,bit)                ( io  |=  (1<<bit) )
//example:sbi(PA_ODR,0);sbi(PA_DDR,0);
#define cbi(io,bit)                ( io  &= ~(1<<bit) )
//example:cbi(PA_ODR,0);cbi(PA_DDR,0);
#define gbi(pin ,bit)     ( pin &   (1<<bit) )
//example: gbi(PA_IDR,0);

void delay_ms(u16 ms);
void delay_us(u16 us);

#endif /*_SYSTEM_H*/
```

13.21. Annex 21 main.c (receptor de dades)

```
#include "clk.h"
#include <xc.h>
#include "ConfigBits.h"
#include "system.h"
#include "lcd_22.h"
#include "definitions.h"

char f[100];
char link[20];
char temperatura[ ]={'T','E','M','P','E','R','A','T','U','R','E','=','\0','\0','\0','\0','\0','\0','\0','\0',' ',
'C'};
char AccesPoint[15];
char Password[20];
```

```
// La funció initializef neteja la variable f
void initializef(void)
{   unsigned char i=0;
    while(i<100) {f[i]='\0';i++;}
}


void main(void)
{
    OSCCON=0b11111100;  //configuració de la frecuenia d'oscilació a 8 MHz
    ADCON1=0X07;
          TRISA=0x01;
          TRISD=0x00;
    T1CON=0b11101101;    //configuració del TIMER1


          init_lcd();
    LCD_test();
          TRISB=0;PORTB=0XF0;delay_ms(100);

          TRISB=0;PORTB=0X0f;delay_ms(100);

          delay_ms(5);
    DisplayString("STARTING ESP8266",3,2,0);
    start ();
    initializef();
    DisplayString("ESP8266 STARTED  ",3,2,0);
    DisplayString("CONFIGURATING ESP8266",3,4,0);
    _delay(1000000);
    ESPconfig();
    DisplayString("ESP8266 CONFIGURATED  ",3,4,0);
    _delay(1000);
    CheckIP();
    _delay(1000);
    DisplayString(link,3,6,0);
    _delay(1000);
    DisplayString(AccesPoint,3,8,0);
```

```
_delay(1000);
DisplayString(Password,3,10,0);
_delay(1000);
DisplayString("waiting sensor",3,8,0);
_delay(5000000);
wait_sensor_conection();
_delay(1000);
DisplayString("sensor connected",3,8,0);
initializef();
_delay(1000);
unsigned char i1=12;
unsigned char i2=17;
while(!RCIF);
recive();

//////////////////Rutina de recepció i escriptura de la temperatura//////////////////////
while(1){
   LATB=0XFF;
   while(!RCIF); //Espera a alguna dada del sensor
   recive();

   _delay(50);
   i1=12;
   i2=17;
  //Escriptura de la temperatura rebuda a la variable temperatura
   while (i1<21)
   {
   temperatura[i1]=f[i2];
   i1++;
   i2++;}
   //Escriptura de la temperatura a la pantalla
   _delay(500);
   DisplayString(temperatura,3,12,0);
   _delay(50000);
   //Neteja del registre CREN per tal de netejar el registre OERR per
   //per tal de poder seguir rebent dades mitjançant la uart en el cas
   //de que s'hagues produit un error durant la ransmissió
   _delay(500);
   CREN=0;
```

```
        _delay(100);
        CREN=1;
        _delay(100);
    }

}
```

13.22. Annex 22 lcd_22.c

```
/*
*=====================================================================
====================================
*
* File              : lcd_22.C
* Hardware Environment:
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
*
*                            (c) Copyright 2011-2016, WaveShare
*                                 http://www.waveShare.net
*                                   All Rights Reserved
*
*=====================================================================
====================================
*/
#include "lcd_22.h"
#include "asciihex8X16.h"
#include <xc.h>

const u16 colorfol[]={0xffe0,0xffe0,0xffe0,0xffe0,0xffe0,0xffe0,0xffe0,0xffe0};
void init_lcd_spi(void)
{
          //TRISC=0xD0;
    TRISCbits.TRISC5=0;
    TRISCbits.TRISC4=1;
    TRISCbits.TRISC3=0;
     SSPSTAT=0X80;
     SSPCON1=0X30;
    INTCON=0X00;
```

```
    PIR1=0X00;

}

void init_touch_spi(void)
{
   TRISCbits.TRISC5=0;
   TRISCbits.TRISC4=1;
   TRISCbits.TRISC3=0;
    SSPSTAT=0X80;
    SSPCON1=0X30;
   INTCON=0X00;
    PIR1=0X00;
}

#if defined(LCD22_SPI)
void post_data(u16 data)
{
        //u8 tmp;
        //while(!SSPIF);
        SSPBUF=(u8)(data>>8);
        while(!SSPIF);
        SSPIF=0;
        //tmp=SSPBUF;

   //while(!SSPIF);
        SSPBUF=(u8)(data);
        while(!SSPIF);
        SSPIF=0;
        //tmp=SSPBUF;
}


#endif

void lcd_rst(void)
{
        reset_clr;
        delay_us(3);
```

```
        reset_set;
        delay_us(3);
}

void post_cmd(u16 index, u16 cmd)
{
        en_lcd_index();
        post_data(index);
        en_lcd_data();
        post_data(cmd);
}

void init_lcd(void)
{
        u16 x, y;/*
        touch_counter = 0;
        touch_wr_index = 0;
        touch_rd_index = 0;*/

        init_lcd_spi();
        en_lcd();
        lcd_rst();

        post_cmd( 0x000, 0x0001 ); /* oschilliation start */
        delay_ms( 10 );
        /* Power settings */
        post_cmd( 0x100, 0x0000 ); /*power supply setup*/
        post_cmd( 0x101, 0x0000 );
        post_cmd( 0x102, 0x3110 );
        post_cmd( 0x103, 0xe200 );
        post_cmd( 0x110, 0x009d );
        post_cmd( 0x111, 0x0022 );
        post_cmd( 0x100, 0x0120 );
        delay_ms( 20 );

        post_cmd( 0x100, 0x3120 );
        delay_ms( 80 );
        /* Display control */
        post_cmd( 0x001, 0x0100 );
```

```
post_cmd( 0x002, 0x0000 );
post_cmd( 0x003, 0x1230 );
post_cmd( 0x006, 0x0000 );
post_cmd( 0x007, 0x0101 );
post_cmd( 0x008, 0x0808 );
post_cmd( 0x009, 0x0000 );
post_cmd( 0x00b, 0x0000 );
post_cmd( 0x00c, 0x0000 );
post_cmd( 0x00d, 0x0018 );
/* LTPS control settings */
post_cmd( 0x012, 0x0000 );
post_cmd( 0x013, 0x0000 );
post_cmd( 0x018, 0x0000 );
post_cmd( 0x019, 0x0000 );

post_cmd( 0x203, 0x0000 );
post_cmd( 0x204, 0x0000 );

post_cmd( 0x210, 0x0000 );
post_cmd( 0x211, 0x00ef );
post_cmd( 0x212, 0x0000 );
post_cmd( 0x213, 0x013f );
post_cmd( 0x214, 0x0000 );
post_cmd( 0x215, 0x0000 );
post_cmd( 0x216, 0x0000 );
post_cmd( 0x217, 0x0000 );

// Gray scale settings
post_cmd( 0x300, 0x5343);
post_cmd( 0x301, 0x1021);
post_cmd( 0x302, 0x0003);
post_cmd( 0x303, 0x0011);
post_cmd( 0x304, 0x050a);
post_cmd( 0x305, 0x4342);
post_cmd( 0x306, 0x1100);
post_cmd( 0x307, 0x0003);
post_cmd( 0x308, 0x1201);
post_cmd( 0x309, 0x050a);
```

```
        /* RAM access settings */
        post_cmd( 0x400, 0x4027 );
        post_cmd( 0x401, 0x0000 );
        post_cmd( 0x402, 0x0000 );   /* First screen drive position (1) */
        post_cmd( 0x403, 0x013f );   /* First screen drive position (2) */
        post_cmd( 0x404, 0x0000 );

        post_cmd( 0x200, 0x0000 );
        post_cmd( 0x201, 0x0000 );

        post_cmd( 0x100, 0x7120 );
        post_cmd( 0x007, 0x0103 );
        delay_ms( 10 );
        post_cmd( 0x007, 0x0113 );

        dis_lcd();
}
void LCD_test(void)
{
        u16  temp,num;
        u8 n,i;

        en_lcd();

        post_cmd(0x210,0x00);
        post_cmd(0x212,0x0000);
        post_cmd(0x211,0xEF);
        post_cmd(0x213,0x013F);

        post_cmd(0x200,0x0000);
        post_cmd(0x201,0x0000);

        en_lcd_index();
        post_data(0x202);
        en_lcd_data();
        for(n=0;n<8;n++)
        {
           temp=colorfol[n];
                for(num=40*240;num>0;num--)
```

```
                {
                        post_data(temp);
                }
        }
        delay_ms(500);
/*      for(n=0;n<1;n++)
        {
                post_cmd(0x210,0x00);
                post_cmd(0x212,0x0000);
                post_cmd(0x211,0xEF);
                post_cmd(0x213,0x013F);

                post_cmd(0x200,0x0000);
                post_cmd(0x201,0x0000);

                en_lcd_index();
                post_data(0x202);
                en_lcd_data();
           temp=colorfol[n];
                for(i=0;i<240;i++)
                {
                        for(num=0;num<320;num++)
                        {
                                post_data(temp);
                        }
                }
        //      delay_ms(50);
        }*/
        dis_lcd();
}


void DisplayChar(u8 casc,u8 postion_x,u8 postion_y)
{
        u8 i,j,b;
        u8 *p;

        en_lcd();
        post_cmd(0x210,postion_x*8);          //x start point
```

```
        post_cmd(0x212,postion_y*16);       //y start point
        post_cmd(0x211,postion_x*8+7);      //x end point
        post_cmd(0x213,postion_y*16+15);    //y end point
        post_cmd(0x200,postion_x*8);
        post_cmd(0x201,postion_y*16);

        en_lcd_index();
        post_data(0x202);
        en_lcd_data();
        p=ascii;
        p+=casc*16;
        for(j=0;j<16;j++)
        {
                b=*(p+j);
                for(i=0;i<8;i++)
                {
                        if(b&0x80)
                        {
                                post_data(COLOR_BLACK);
                        }
                        else
                        {
                                post_data(COLOR_YELLOW);
                        }
                        b=b<<1;

                }
        }
        dis_lcd();
}

void DisplayChar_Reverse(u8 casc,u8 postion_x,u8 postion_y)
{
        u8 i,j,b;
        u8 *p;

        en_lcd();
        post_cmd(0x210,postion_x*8);        //x start point
        post_cmd(0x212,postion_y*16);       //y start point
```

```
        post_cmd(0x211,postion_x*8+7);      //x end point
        post_cmd(0x213,postion_y*16+15);    //y end point

        post_cmd(0x200,postion_x*8);
        post_cmd(0x201,postion_y*16);

        en_lcd_index();
        post_data(0x202);
        en_lcd_data();
        p=ascii;
        p+=casc*16;
        for(j=16;j>0;j--)
        {
                b=*(p+j-1);
                for(i=0;i<8;i++)
                {
                        if(b&0x01)
                        {
                                post_data(COLOR_BLACK);
                        }
                        else
                        {
                                post_data(COLOR_YELLOW);
                        }
                        b=b>>1;

                }
        }
        dis_lcd();
}

u8* swap(u8 *s,u8 sz)
{
   u8 i=0;
   static u8 b[10]={0};
   s+=sz-2;
   for(i=0;i<sz-1;i++)
   {
      b[i]=*s;
```

```
        s--;
    }
    s=b;
    return s;
}

void DisplayString(u8 *s,u8 x,u8 y,u8 Reverse)
{
        u8 a[10],i;
        if(Reverse)
        {
                i=0;
                while(*s){a[i]=*s;s++;i++;}
                s=swap(a,sizeof(a));
        }
        while (*s)
        {
                if(Reverse)
                        {DisplayChar_Reverse(*s,x,y);}
                else
                        {DisplayChar(*s,x,y);}
                if(++x>=30)
                {
                        x=0;
                        if(++y>=20)
                        {
                          y=0;
                        }
                }
                s++;
    }
}
/**/
u16 get_touch_data(u16 cmd)
{
        u8 tmp;
        SSPBUF = cmd;
        while(!SSPIF);
        SSPIF=0;
```

```
	SSPBUF = 0;
	while(!SSPIF);
	SSPIF=0;

	tmp =SSPBUF;

	SSPBUF = 0;
	while(!SSPIF);
	SSPIF=0;

	return ( ((u16)tmp)<<5 | ((u16)SSPBUF)>>3);
}

xy_t get_touch_xy(void)
{
	xy_t tmp_xy;
	if(!PENIRQ){
		tmp_xy.x = get_touch_data(TOUCH_CMD_X);
	   tmp_xy.y = get_touch_data(TOUCH_CMD_Y);
	} else{
		tmp_xy.x = 0xFFFF;
	   tmp_xy.y = 0xFFFF;
	}
	return tmp_xy;
}

u8 get_point_xy(void)
{
	u8 n,m,tmp;
	xy_t tmp_xy_buf[SAMP_COUNT], tmp_xy;
	u32 tmp_x = ((u32)tmp_xy_buf[SAMP_COUNT/2].x +
(u32)tmp_xy_buf[SAMP_COUNT/2-1].x)/2;
	u32 tmp_y = ((u32)tmp_xy_buf[SAMP_COUNT/2].y +
(u32)tmp_xy_buf[SAMP_COUNT/2-1].y)/2;

	if(touch_counter>=(TOUCH_MAX_CACHE-1)){
		return 0;
	}
```

```
	init_touch_spi();
	en_touch();

	for(n=0; n<SAMP_COUNT; n++){
		tmp_xy_buf[n] = get_touch_xy();
	}
	dis_touch();
	for(n=0; n<(SAMP_COUNT-1); n++){
		for(m=0; m<(SAMP_COUNT-n-1); m++){
			tmp = m+1;
			if((tmp_xy_buf[m].x + tmp_xy_buf[m].y) > (tmp_xy_buf[tmp].x + tmp_xy_buf[tmp].y)){
				tmp_xy = tmp_xy_buf[tmp];
				tmp_xy_buf[tmp] = tmp_xy_buf[m];
				tmp_xy_buf[m] = tmp_xy;
			}
		}
	}
	if((tmp_xy_buf[SAMP_COUNT/2].x - tmp_xy_buf[SAMP_COUNT/2-1].x > SAMP_THRESHOLD)
		|| (tmp_xy_buf[SAMP_COUNT/2].y - tmp_xy_buf[SAMP_COUNT/2-1].y > SAMP_THRESHOLD)){
		return 0;
	}

	if(tmp_x >= 0xFFF || tmp_y >= 0xFFF){
		return 0;
	}
	touch_xy_buffer[touch_wr_index].x = ((tmp_x * 240)>>12);
	touch_xy_buffer[touch_wr_index].y = ((tmp_y * 320)>>12);
	if(touch_wr_index < (TOUCH_MAX_CACHE-1)){
		touch_wr_index++;
	}else{
		touch_wr_index = 0;
	}
	touch_counter++;
	return 1;
}
```

```
u8 draw_lcd(void)
{
	u8 n;
	init_lcd_spi();
	en_lcd();
	if(touch_counter==0){
		return 0;
	}

	touch_counter--;

	post_cmd(0x210,touch_xy_buffer[touch_rd_index].x);
	post_cmd(0x212,touch_xy_buffer[touch_rd_index].y);
	post_cmd(0x211,touch_xy_buffer[touch_rd_index].x+(DOT_WIDTH-1));
	post_cmd(0x213,touch_xy_buffer[touch_rd_index].y+(DOT_WIDTH-1));
	if(touch_rd_index < (TOUCH_MAX_CACHE-1)){
		touch_rd_index++;
	}else{
		touch_rd_index = 0;
	}
	//post_cmd(0x0005,0x0010);

	en_lcd_index();
	post_data(0x202);
	en_lcd_data();
	for(n=0; n< (DOT_WIDTH*DOT_WIDTH); n++)
	{
		post_data(COLOR_BLACK);
	}
	dis_lcd();
	return 1;
}
```

13.23. Annex 23 ESPconfig

```
/*La funció configura l'ESP com a servidor*/

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
```

```
void sendstring(char*);
void recive (void);

void ESPconfig(void)
{
   extern char f[100];
   unsigned char pas=1;

   while (pas){

      if (pas==1)
      {
         _delay(500000);
         sendstring("AT\r\n\0");
         recive();
         _delay(500000);
         pas=2;

         if (f[7]=='O'&&f[8]=='K')
           {pas=2;}

         else
           {pas=1;}
      }

      if (pas==2)
      {
         _delay(500000);
         sendstring("AT+CWMODE=3\r\n\0");
         recive();
         _delay(500000);
         sendstring("AT+CIPMUX=1\r\n\0");
         recive();
         _delay(500000);
         sendstring("AT+CIPSERVER=1,80\r\n\0");
```

```
            recive();
            _delay(500000);
            pas=0;
        }
    }
}
```

13.24. Annex CheckIP.

```
/*La funció determina i retorna la direcció IP a la que s'ha de conectar el sensor */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

void sendstring(char*);
void recive (void);

void CheckIP(void)
{
    extern char f[100];
    extern char link[25];
    extern char AccesPoint[15];
    extern char Password[20];
    unsigned char j=24;
    unsigned char i=8;
    unsigned char k=22;
    unsigned char l=7;

    _delay(500);
    sendstring("AT+CIFSR\r\n\0");      //El comandament AT+CIFSR pregunta a l'ESP la direcció del servidor
    recive();
    _delay(500000);

    while (f[j]!="")
    {
        link[i]=f[j];
        i++;
```

```
        j++;
    }
    link[0]='L';
    link[1]='l';
    link[2]='N';
    link[3]='K';
    link[4]=' ';
    link[5]='l';
    link[6]='S';
    link[7]=':';
_delay(5000000);
    CREN=0;
    _delay(100);
    CREN=1;
    _delay(500000);
    _delay(500);
    sendstring("AT+CWSAP?\r\n\0");   //El comandament AT+CWSAP pregunta a l'ESP
                                     //el nom del punt d'acces i la contrasenya
    while(!RCIF);
    recive();
    _delay(500000);

    AccesPoint[0]='A';
    AccesPoint[1]='P';
    AccesPoint[2]=' ';
    AccesPoint[3]='l';
    AccesPoint[4]='S';
    AccesPoint[5]=' ';
    AccesPoint[6]='=';

    Password[0]='P';
    Password[1]='a';
    Password[2]='s';
    Password[3]='s';
    Password[4]=' ';
    Password[5]='i';
    Password[6]='s';
    Password[7]='=';
```

```
    while (f[k]!="")
    {
        AccesPoint[l]=f[k];
        k++;
        l++;
    }
    k=k+3;
    l=8;
    while (f[k]!="")
    {
        Password[l]=f[k];
        k++;
        l++;
    }
    _delay(500000);
}
```

13.25. Annex 25 wait_sensor_connection

```
/*
 * La funció s'espera a que el PIC amb el sensor es connecti
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

void sendstring(char*);
void recive (void);

void wait_sensor_conection(void) {

    extern char f[100];
    unsigned char IN;

    while(!RCIF);
    recive();

    while (IN){
        CREN=0;
```

```
        _delay(100);
        CREN=1;
        sendstring("AT+CWLIF\r\n\0"); //El comandament AT+CWLIF llegeix les direccions IP conectades al servidor
        recive();
        _delay(50000000);

    //En el cas de que la direcció connectada al servidor sigui "0.0.0.0"(no hi ha cap client conectat)
    //el PIC segueix esperant, en cas contrari, la rutina s'ha acabat.
        if(f[11]=='1'){IN=0;}

        }
    _delay(500);
}
```

13.26. Annex 26 Main.c (emissor al µC amb la pantalla LCD)

```
/*
 * File:   main.c
 * Author: aziz
 *
 * Created on 18 / de novembre / 2018, 17:44
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "ConfigBits.h"
#include "functinit.h"


#define S1 PORTDbits.RD0

unsigned char valor_temperatura[9];
char f[100];
```

```
char *user="ESP";
char *pass="password";
char *link="192.168.4.1";



void main(void) {

   OSCCON=0b11111100;   //configuració de la frecuencia d'oscilació interna a 8MHz
   T1CON=0b11101101;    //configuració del TIMER1

   TRISDbits.TRISD0=1;  //inicialització del botó S1 com a entrada


   TRISB=0x00;          //configuració dels registres que encenen els LED's com a sortida
   long time1=0;
   unsigned char primercicle=1;
   unsigned char p=0;

   WRITETIMER1(0);
   while(1)
   {
      if(primercicle)            //El primer cicle, es configura la uart
      {  LATB=0x00;              //i s'inicialitza i configura la conexió wifi
         startESP();
         primercicle=0;
         confignetwork(user,pass);
         LATB=0xFF;
         _delay(100000);
      }
      time1=READTIMER1();
      if(time1>30000)
      {WRITETIMER1(0);
      p++;
      time1=0;
      }
      if (!S1||p==200){
      p=0;                    //cada com que s'apreta el botó S1, es llegeix i envia
      LATB=0x00;                 // la temperatura al servidor
```

```
        readtemp();
        CIPSTART();
        _delay(1000000);
        CIPSEND();
        _delay(1000);
        LATB=0xFF;
        }
    }
}
```

Bibliografia

[1] [https://wiki.aprbrother.com/en/Firmware_For_ESP8266.html] [7/1/2019]

[2] [https://sourceforge.net/projects/realterm/files/] [7/1/2019]

[3] *[ESP8266 Datasheet]* [https://www.espressif.com/en/support/download/other-tools] [7/1/2019]

[4] [https://www.waveshare.com/wiki/File:2.2inch-320x240-Touch-LCD-A-Code.7z] [7/1/2019]

[5] *[DS18B20 Datasheet]* [https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf] [7/1/2019]

[6] [https://www.microchip.com/mplab/mplab-x-ide] [7/1/2019]

[7] [AT Instruction set] [https://www.espressif.com/sites/default/files/documentation/4aesp8266_at_instruction_set_en.pdf] [7/1/2019]

[8] [https://www.apachefriends.org/es/index.html] [7/1/2019]

[9] [https://kst-plot.kde.org/download/] [7/1/2019]

[10] *[MPLAB X IDE User's Guide]* [http://ww1.microchip.com/downloads/en/DeviceDoc/50002027D.pdf] [7/1/2019]

[11] *[Pic18F4520 Datasheet]* [https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf] [7/1/2019]

[12] [https://www.aprendiendoarduino.com/tag/wemos-d1-mini/] [12/1/2019]

[13] [https://aprendiendoarduino.wordpress.com/2017/09/13/uso-esp8266-con-arduino-puerto-serie/] [12/1/2019]

[14] [https://simba-os.readthedocs.io/en/latest/boards/esp12e.html] [12/1/2019]

[15] [https://www.amazon.es/SODIAL-Conversor-cepillo-alambre-descargar/dp/B00YGPU19U/ref=sr_1_21?ie=UTF8&qid=1547322800&sr=8-21&keywords=usb+ttl] [12/1/2019]

[16] https://es.aliexpress.com/item/PIC18F4520-I-P-PIC18F4520-PIC-8-bit-RISC-Development-Evaluation-Board-Open18F4520-Standard/700652633.html [12/1/2019]

[17] https://articulo.mercadolibre.com.ar/MLA-757973816-ds18b20-sensor-de-temperatura-18b20-arduino-original-x-20-u-_JM?quantity=1 [12/1/2019]

[18] https://uk.rs-online.com/web/p/chip-programmers/6601264/ [12/1/2019]

[19] https://commons.wikimedia.org/wiki/File:ASCII-Table.svg [12/1/2019]

[20] https://www.behind-the-scenes.co.za/using-the-uart-interface/ [12/1/2019]

[21] https://www.solitontech.com/uart-protocol-validation-service/ [12/1/2019]

[22] https://www.prometec.net/bus-spi/ [12/1/2019]

[23] https://www.apple.com/environment/pdf/products/iphone/iPhone_8_PER_sept2017.pdf [12/1/2019]

[24] https://www.malavida.com/es/analisis/cual-es-el-impacto-ecologico-de-un-movil-asi-contamina-un-telefono-005716 [12/1/2019]