Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial I Automàtica

Títol: DATALOGGER UNIVERSAL AMB SUPERVISIÓ WEB

Document: 1. Memòria

Alumnes: Lucas Ariel Castorina

Tutor: Antoni Martorano Gomis
Departament: Arquitectura i Tecnologia de Computadors
Àrea: ATC

Convocatòria (mes/any): Febrer/2017

ÍNDEX

1 INTRODUCCIÓ

Es vol dur a terme la realització d'un component electrònic que sigui capaç de monitoritzar diferents variacions d'un rang de voltatge normalitzat (0-5, 0-3.3), i posteriorment es voldrà guardar les variables, en un fitxer, els aparells que comunament fan aquestes funcions es diuen DATALOGGER, de la traducció de l'anglès logger com registrador.

A vegades es requereix mesurar certes variables físiques presents en un determinat ambient, per fer-ho existeixen eines electròniques capaces de censar aquestes variables físiques i emmagatzemar-les en una memòria extraïble perquè puguin ser processades com es desitgi, a aquestes eines se'ls denomina DATALOGGER.

Un DATALOGGER és un dispositiu electrònic que registra mesures ordenades en el temps, provinents de diferents sensors. Després cada mesura es emmagatzemada en una memòria, juntament amb la seva respectiva data i hora.

En general els DATALOGGERS són petits, i estan conformats per un microprocessador, una memòria per a l'emmagatzematge de les dades i diferents entrades. En aquest treball es vol portar a terme un disseny electrònic amb determinades característiques que faran que l'aparell tingui aplicació rellevant i a un cost relativament baix. Com a objectius constructius , les característiques bàsiques d'un DATALOGGER han de ser; portabilitat, baix cost, regulació d'intervals de mostreig programables, bona capacitat d'emmagatzematge de dades, i tenir una interfase amb l'usuari.

A més d'un DATALOGGER, s'ha implementat un sistema de monitoratge via web, en el qual s'utilitza un dispositiu que llegeix les variables a intervals de temps, amb l'objectiu d'enviar aquestes lectures a un servidor extern, anomenat Broker perquè siguin presentades a l'usuari mitjançant una pàgina web, en forma de gràfics, taules de dades, etc.

1.1 ANTECEDENTS

Durant les dues últimes dècades la necessitat de supervisar les dades obtingudes de diferents processos ens ha demostrat que quan més qualitat i control tenim de les dades major és el grau d'efectivitat al moment de prendre una decisió.

El naixement dels DATALOGGER va ser un gran avanç tecnològic a l'hora de registrar dades, i avui en dia la varietat és molt amplia, des de registradors de gran format amb múltiples funcions d'enregistrament i gran resistència mecànica, fins petits dispositius que es poden amagar amb una mà amb una funció especifica a l'hora de realitzar l'adquisició de dades.

Tot i que la funció és similar a la d'una targeta d'adquisició de dades, els DATALOGGERS realitzen una adquisició de dades més lenta, de l'ordre d'una mostra per segon en els casos més ràpids.

Els DATALOGGERS poden ser construïts específicament per monitoritzar una o vàries variables físiques en concret, com pot ser la temperatura, que és una de les més habituals. També existeixen DATALOGGERS universals, com és el cas d'aquest treball en concret, on l'aparell en lloc d'estar destinat a monitoritzar una variable física com per exemple la temperatura, té per objectiu monitoritzar una variable elèctrica (voltatge), obligant a l'usuari a entregar-li un senyal normalitzada (0-5V, 0-3.3V) independentment del tipus de sensor utilitzat.

En l'actualitat, al mercat existeix una gran varietat de DATALOGGERS, aquest projecte es diferenciarà amb la simplicitat d'us, flexibilitat funcional i un baix cost, entre un 60 i 80 per cent més barat que els que es troben en el mercat amb similars característiques.

Comparativament amb els productes que avui en dia existeixen en el mercat la idea és aportar una solució flexible que es puguin adaptar a les funcions que necessita el client, amb alguns canvis de programació.

La part de hardware es mantindrà sense canvis, però la possibilitat d'actualitzar el programa del producte permetrà més flexibilitat en l'ús per complir amb els requeriments de funcionament d'usuaris més exigents o amb funcionalitats més especifiques.

1.2 OBJECTE

L'objecte del treball és crear un DATALOGGER de baix cost, que guardi les lectures analògiques i a més de guardar-les es puguin consultar a través d'internet de manera remota.

També, com a funcions secundàries, es controlarà el temps d'enviament de dades, l'activació/Desactivació de gravació de dades a la SD i la modificació de paràmetres bàsics com pot ser data i hora mitjançant un menú de configuració.

L'alimentació serà a 230/110V AC amb un sistema de rectificació a corrent continua. En el cas que es vulgui donar més portabilitat amb una alimentació a bateries, en aquest document es realitzaran els càlculs teòrics, però no es durà a terme físicament.

1.3 ESPECIFICACIONS I ABAST

L'abast del projecte serà l'adquisició de dades de senyals analògiques de 0-5 Volts amb una resolució de 16 bits i 0-3,3 Volts amb una resolució de 10 bits, que es guardaran amb una targeta SD i es connectarà a una xarxa wifi per enviar les dades a un Broker online que permetrà consultar-les des de qualsevol PC.

També es podran supervisar les sortides/entrades digitals.

En el cas que no sigui viable la connexió a internet, tindrà l'opció de funcionament com a web server, en mode "Acces Point", donant així, la possibilitat de visualitzar les entrades que s'estan guardant dintre la SD i activar o desactivar la seva gravació.

Com a criteris de construcció del DATALOGGER, s'ha volgut emfatitzar en la flexibilitat d'ús, una bona capacitat d'emmagatzematge i un baix cost de producció.

2 DESENVOLUPAMENT DEL HARDWARE

Començarem definint els blocs que compondran el DATALOGGER, amb el seu material corresponent, les seves directrius de funcionament i com interactuen entre ells.

El diagrama de blocs ens dóna una visió més amplia a l'hora de definir el funcionament i separar la problemàtica afavorint la seva resolució de manera més clara i entenedora.

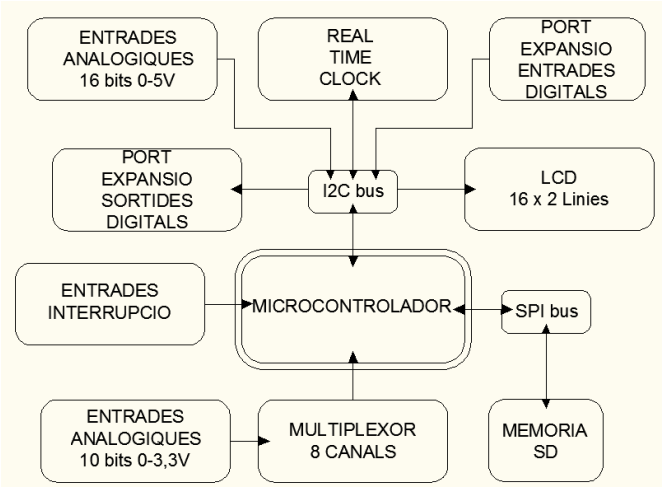

Figura 1. Diagrama de blocs

A la figura 1, podem veure que el cos central del disseny del hardware tenim el microcontrolador, que a més de rebre les lectures dels sensors decidirà quan i com s'enviaran i serà el cervell que tindrà els algoritmes de filtrat de la senyal i gestionarà els menús que l'usuari farà ús per configurar les funcions principals i el canvi de paràmetres, com per exemple la data i hora.

Les lectures analògiques es faran de dues maneres diferents, una de baixa resolució que és directa al ADC intern del microcontrolador, i l'altre que és de major resolució que serà amb un ADC extern que es comunicarà amb bus I2C.

El bus I2C, a més de portar les mesures del ADC de major resolució, portarà els ports d'expansió que són necessaris, ja que el microcontrolador que hem triat porta poques entrades/sortides que s'empraran per entrades d'interrupció i per controlar el multiplexor.

El Real Time Clock (RTC) es comunicarà per bus I2C. Serà necessari per tenir controlat el temps en què es fa la lectura del senyal i la data.

El LCD també es comunicarà amb el microcontrolador via bus I2C, aquest actuarà d'interfase entre l'usuari i el microcontrolador.

L'enviament de dades a la memòria SD es farà mitjançant un bus SPI que aporta més velocitat que el bus I2C i és el protocol que fan servir la majoria de memòries comercials del tipus SD.

Tant el bus I2C, com el bus SPI, són bidireccionals, i ens permetran modificar les registres de configuració dels circuits integrats al qual estan connectats fent que el component electrònic final sigui més flexible respecte les seves prestacions. Per exemple, gràcies a poder modificar les registres de manera bidireccional podrem variar el guany que porta integrat el convertidor analògic digital d'alta resolució.

Com el disseny porta components amb diferents valors de tensió, mes concretament, 5 Volts i 3.3 Volts, es necessari un adaptador de bus I2C perquè la comunicació sigui compatible. Aquest adaptador de bus es el PCA9515.

3 FUNCIONAMENT DELS PRINCIPALS COMPONENTS

El funcionament està centralitzat en un microcontrolador. La majoria d'entrades i sortides digitals del microcontrolador es faran servir per la comunicació amb els dos buses diferents, un I2C i l'altre SPI, i dues entrades més per interrupcions de polsadors.

El bus I2C connecta el microcontrolador amb els ports d'expansió, les entrades analògiques de 16 bits, el RTC, i el LCD.

S'han integrat els ports d'expansió per poder controlar més entrades/sortides digitals ja que les pròpies del microcontrolador es feien servir per interrupcions i comunicacions.

El RTC, que s'encarrega d'emmagatzemar la data i hora actual i porta una bateria perquè no s'hagi d'entrar la data i hora cada cop que es reinicià el sistema. El RTC és molt important a l'hora d'emmagatzemar les dades perquè cada mesura vindrà acompanyada de l'hora, minuts i segons en que s'ha fet el registre.

Tot i que el LCD és un display del tipus set segments i podria ser controlat per sortides digitals, porta incorporat un circuit integrat que fa de plataforma de comunicació a I2C, reduint el nombre d'entrades/sortides digitals necessàries del microcontrolador.

En el sistema hi ha molts paràmetres importants que són configurables per l'administrador mitjançant tecles i una la pantalla LCD, paràmetres com data/hora, temps de mostreig, sortides/entrades digitals, etc.

L'emmagatzematge de les dades es farà dintre d'una targeta SD, aquestes tenen una bona relació preu/espai i la descàrrega de dades és tan fàcil com introduir-la amb un adaptador USB a qualsevol ordinador per gestionar els arxius gravats per DATALOGGER.

El bus SPI es fa servir per comunicar amb la targeta SD, s'ha triat aquest tipus de comunicacions perquè el bus SPI és més ràpid que l'I2C, tot i que necessita més pins del microcontrolador i les targetes SD del mercat són amb SPI majoritàriament.

Els senyals analògiques entren al sistema de dues maneres diferents; directament al ADC del microcontrolador i indirectament per bus I2C. Més endavant es profunditzarà en aquests punts.

La monitorització de les dades s'ha fet amb un servidor web, que fa la funció de Broker. A la figura nº 2 es pot veure la pantalla web de supervisió de les dades que tindrem accés. Pel nostre exemple d'aplicació en concret s'ha fet servir el servidor Adafruit.io que té una bona documentació i llibreries. Aquest servidor ens mostrarà les dades analògiques i ens permetrà activar o desactivar entrades digitals o modificar el temps d'adquisició de dades o enviament.

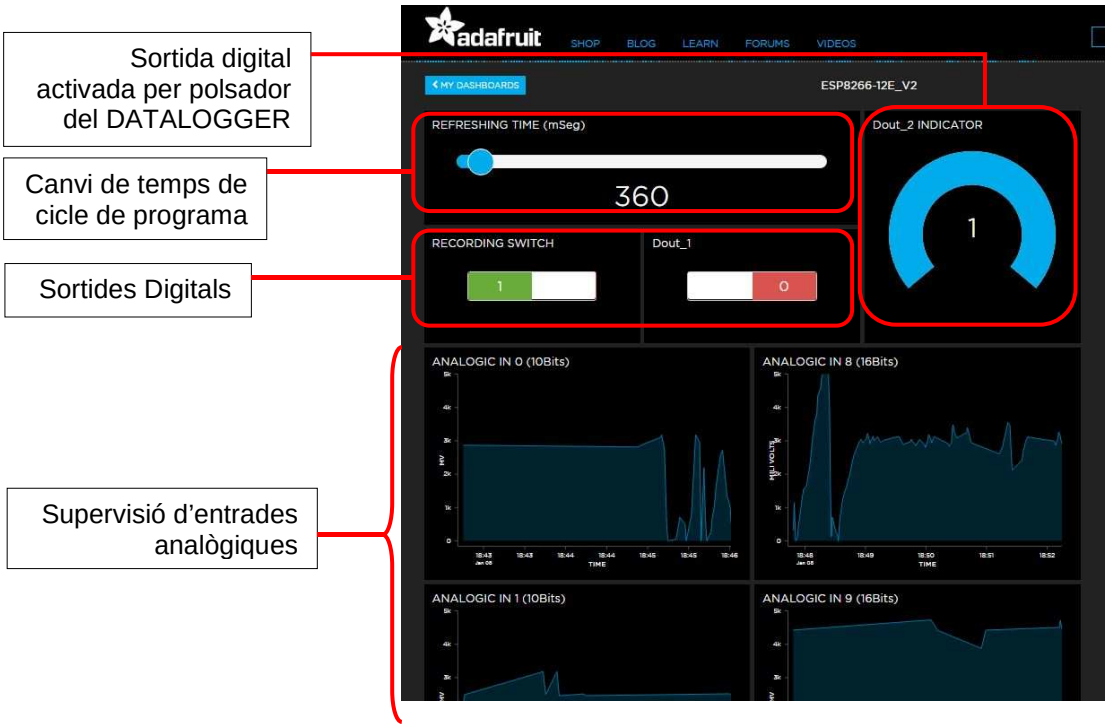

Figura 2. Pàgina web del servidor Broker ADAFRUIT

3.1 ENREGISTRAMENT DE DADES

El nostre sistema té per objectiu principal, enregistrar dades, per tant és molt important que les dades es puguin guardar de manera ordenada amb la data i hora en què s'ha fet.

A més de mantenir aquest ordre després ens interessarà poder tractar-les per fer gràfiques i càlculs per poder-les interpretar correctament. Per tant per poder-les tractar, s'han guardat

en un format txt, amb un espai del tabulador donant així la possibilitat d'obrir l'arxiu amb altres programes que no siguin de text com per exemple l'excel.

També s'ha triat aquest format perquè són arxius que ocupen poc espai en la targeta SD amb un gran volum de dades. Un arxiu txt ens guardarà uns cinquanta canvis per kilobyte. Si la sd és de 4 Gigabytes, podrem registrar més de docents milions de canvis.

Com el que es busca es l'enregistrament de canvis, era necessari decidir que es grava, descartant l'enviament de dades mentre la variable no canvia, evitant un mal ús de la memòria de la SD. El límit (Thereshold) del filtratge del senyal que permet l'enviament de dades a la SD és un paràmetre que es pot canviar i que per defecte s'ha deixat a 20 LSB (Bit menys significatiu) en el cas del ADC de 16 bits, tenint en compte que la mesura es pot veure afectada pel soroll extern de valors de 17 LSB.

És a dir, si la diferència que existeix entre la senyal actual i l'anterior és major a 20 LSBs s'enviarà a la SD i via wifi al servidor Broker.

Aquest algoritme s'ha implementat per software i en cada una de les entrades analògiques

Aquest DATALOGGER també té la possibilitat d'aturar l'enregistrament de dades de manera remota via web, i també es podrà configurar una sortida digital que ens indicarà que l'espai a la SD s'està esgotant.

3.2 MODES DE TREBALL: Broker o Webserver.

El nostre dispositiu, a més d'enregistrar les dades, que és el seu objectiu principal, també tindrà dos modes de funcionament; mode Broker, i mode Webserver. Aquests modes seran seleccionats amb un selector abans d'inicialitzar el microcontrolador.

Mentre estigui el mode Broker seleccionat, la visualització de les dades serà en una gràfica allotjada a una pàgina web que es podrà consultar des de qualsevol ordinador o dispositiu que tingui un navegador amb connexió a internet. Un cop les dades s'han filtrat i s'ha pres la decisió d'enviar-les, el microcontrolador les envia via wifi al router, i d'aquí al servidor Broker que s'encarrega de traduir-les perquè es puguin visualitzar en una gràfica entenedora. El

protocol d'enviament de dades del router al servidor Broker és MQTT (Message Queue Telemetry Transport), que avui en dia és un del més usats per comunicacions de petits aparells, també es coneix com el protocol d'Internet de les coses (IoT).

MQTT ofereix un protocol de publicació/subscripció altament escalable amb lliurament assegurat de missatges. Per enviar i rebre missatges MQTT, són necessàries llibreries que els mateixos servidors Brokers proporcionen, accelerant molt la posta en marxa del sistema.

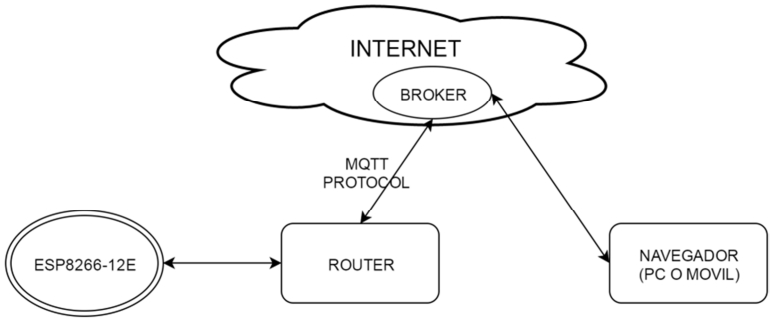

Figura 3. Adreçament de comunicació en mode Broker

Si s'ha triat amb el selector el mode WEBSERVER, les dades es visualitzaran en qualsevol ordinador o dispositiu que tingui connexió wifi, mostrant únicament l'últim canvi registrat, a diferència de la visualització per pantalla del mode Broker que ens ensenya un historial dels últims canvis.

El microcontrolador farà de punt d'accés, mentre el dispositiu o ordinador serà el client que sol·licita les dades. Aquest mode està pensat per llocs on la connexió a internet no existeix, i es vol tenir accés a les dades del DATALOGGER de manera remota, tenint en compte que les limitacions de la distancia venen donades per la potència del senyal WIFI, la potència de recepció de l'antena del nostre dispositiu client i els obstacles que es tinguin entre el microcontrolador i el client.

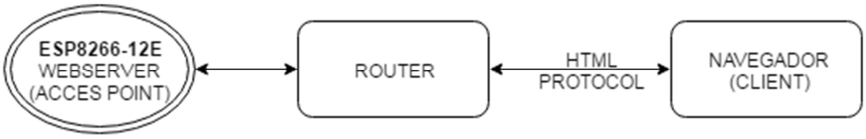

Figura 4. Adreçament de comunicació en mode Webserver

3.3 COMUNICACIÓ SERIE (UART)

El microcontrolador ESP8266 porta un port de comunicació UART, de les sigles en angles: Universal Asynchronous Receiver Transmitter. Aquest port permet comunicació serial asíncrona full-duplex amb altres dispositius, o components tals com a computadores, convertidors, mòduls sense fils, i d'altres dispositius. Les característiques principals del mòdul UART són; transmissió Full-duplex de 8 o 9 bits a través dels pines UxTX and UxRX., opcions de paritat Parell, Imparell o Sense paritat per a dades de 8 bits, un o dos bits de parada (Stop bits), detecció automàtica de la taxa de bauds, taxes de transferència des de 76 bps fins a 20 Mbps a 80 MHz, detecció d'errors de paritat i desbordament de buffer, interrupcions de transmissions i recepció, mode de circuit tancat (Loopback) per a diagnòstic i depuració.

Aquest port ens ha permès realitzar la depuració del programa, ja que la majoria dels compiladors porten un monitor del bus sèrie, mitjançant comandes simples en el nostre programa podem veure valors de variables d'interès, increment de comptadors, valors que s'han enviat des de la web, etc.

Antigament la majoria dels ordinadors portaven sortides UART amb protocols RS-232 amb connectors de format tipus DB9 amb els que es podien comunicar amb altres dispositius externs, avui en dia és més difícil trobar aquests connectors perquè s'han anat substituint per USBs.

La placa de desenvolupador ESP8266-12E, ja porta instal•lat un chip CH340, que s'encarrega de transformar la comunicació sèrie de la UART del microcontrolador a USB, així podem carregar el programa de l'ordinador a la memòria de la placa via USB. Aquesta comunicació també és molt útil per a realitzar la depuració del programa imprimint pel monitor serial informació de variables que proporciona el nostre programa que s'ha carregat dintre del microcontrolador.

El port serial envia i rep bytes bit a bit. Tot i que això és més lent que la comunicació en paral·lel, que permet la transmissió d'un byte complet per vegada, aquest mètode de comunicació és més senzill i pot aconseguir majors distàncies, però en el nostre cas no és prioritari la velocitat, per això ens funcionarà perfectament la UART que porta serial que porta el microcontrolador ESP-8266.

La velocitat de transmissió (baud rate), indica el nombre de bits per segon que es transfereixen, i es mesura en bauds (bauds). Per exemple, 300 bauds representa 300 bits per segon. Quan es fa referència als cicles de rellotge s'està parlant de la velocitat de transmissió. En el nostre cas s'ha configurat a 9600 bauds dintre dels paràmetres del microcontrolador.

En el cas de la versió Standard del ESP8266-12e, on l'interface UART-USB no està integrada amb el CH340 a la mateixa placa, s'ha fet servir per fer proves un altre placa per separat amb el chip FTDI232R, que compleix la mateixa funció però comercialment és més econòmica que l'anterior. Aquesta configuració és molt útil per reduir les mides del disseny de la PCB, ja que aquesta interfase només és útil per transferir el programa i debugar-lo quan esta en fase de prototip, en el nostre cas s'ha fet servir la placa de desenvolupador ESP8266-12E a les dos versions de prototips.

4 ELECCIÓ DEL MICROCONTROLADOR

El model que s'ha triat és el ESP8266, més concretament el ESP8266-12E en versió Development Board.

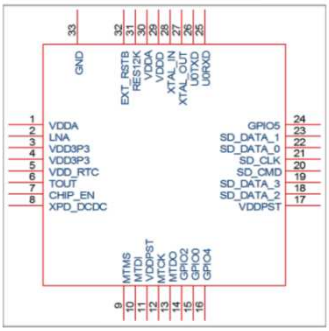

Figura 5. Pin OUT ESP8266

Els principals motius per triar aquest microcontrolador és que incorpora comunicació WIFI amb 2.4 GHz amb antena inclosa integrada a la placa en forma de pista de coure sobre pcb, una velocitat del rellotge de la cpu de 80Mhz i un preu per sota dels 3 euros. Un altre motiu de pes, a l'hora de triar aquest microcontrolador ha sigut que existeix bona documentació i llibreries en C i C++ que faciliten el funcionament i donen més agilitat a l'hora de realitzar la integració de moltes funcions i components.

Té un port de comunicació serial (UART) que ens serà de molta utilitat a l'hora de realitzar el debuggin del programa. Aquesta sèrie té l'opció de reduir el consum d'energia en mode Sleep, important a l'hora de prioritzar la portabilitat del DATALOGGER, si fos el cas en que es volgués fer una pressa de dades en zones remotes amb una alimentació autònoma amb bateries i panells solars.

Actualment, aquest microcontrolador és un dels més usats i populars dintre de les aplicacions que requereixen connexió wifi. El seu preu ha sigut una veritable revolució del mercat tecnològic, ja que la seva competència, el MKR1000, deu vegades més car i la seva programació era molt més complexa reduint molt els usuaris que podien contribuir al desenvolupament del software lliure per fer ús de totes les seves prestacions.

Cal esmentar que pels dos prototips (V1 i V2) s'ha fet servir el microcontrolador ESP8266-12E amb una versió per desenvolupadors (Development Board) que porta una placa on té integrat l'interconecció de l'uart per realitzar la descàrrega del programa a través del USB del PC (Personal Computer) i una millor disposició dels pins que ens permeten fer proves fent les interconnexións en qualsevol breadboard per testejar el seu funcionament.

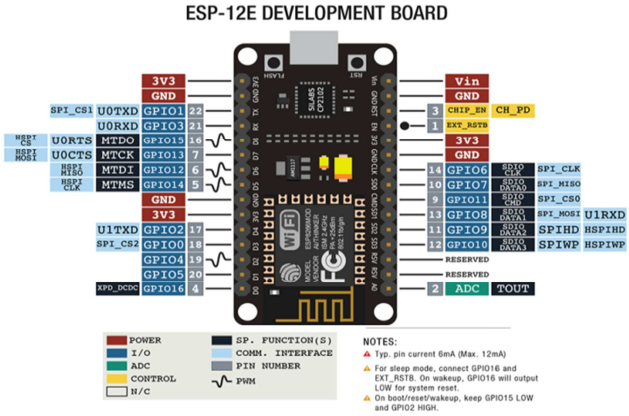

Figura 6. ESP8266-12E Development Board pin out.

L'encapsulat del microcontrolador ESP8266 es QFN de 32 pins, però comercialment es ven muntat amb una petita placa de 25x15 mm, amb la memòria rom integrada, una antena en forma de pista de coure sobre la PCB, els components mínims necessaris perquè funcioni correctament i una carcassa metàl·lica connectada a terra per reduir el soroll extern.

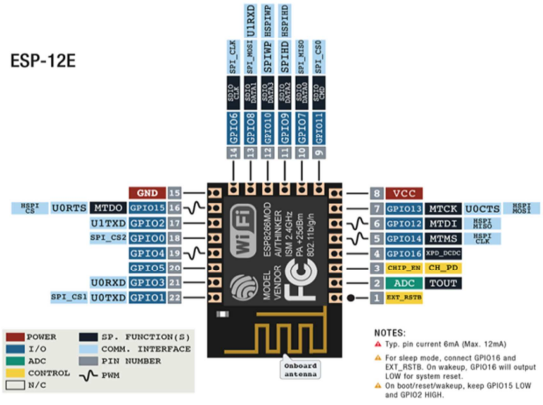

Figura 7. ESP8266-12E pin out versió comercial.

5 ENTRADES DE SENYALS ANALÒGIQUES

Les entrades analògiques són tractades pel microcontrolador de manera **directa** e **indirecta**.

De manera **directa**: El microcontrolador ESP8266 incorpora un únic pin d'entrada ADC de 10 bits a 3,3V, els senyals analògiques es fan passar per un multiplexor de 8 canals i cada canal és seleccionat pel microcontrolador amb sortides digitals, obtenint així, l'entrada de 8 analògiques per un sol pin del microcontrolador. El multiplexor és el MAX4617 i suporta alta velocitat de commutació de canals, 15nS per Ton i 10nS per Toff.

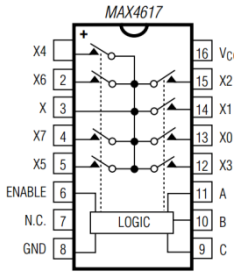

Figura 8. Pin Out del multiplexor MAX4617

De manera indirecta: Amb un ADC de quatre canals de 16 bits per canal, amb un rang d'entrada de 0-5 Volts i comunicació I2C. Concretament el ADS1115. Aquest integrat incorpora un "Voltage Reference" que li dóna major estabilitat a la mesura i també incorpora un guany programable (PGA) de 2/3, 1, 2, 4, 8, o 16, per ampliar la resolució acondicionant el senyal d'entrada. També és possible programar el tipus d'entrada, admet 2 entrades diferencials o 4 en mode de terminació simple amb el terminal negatiu a terra.

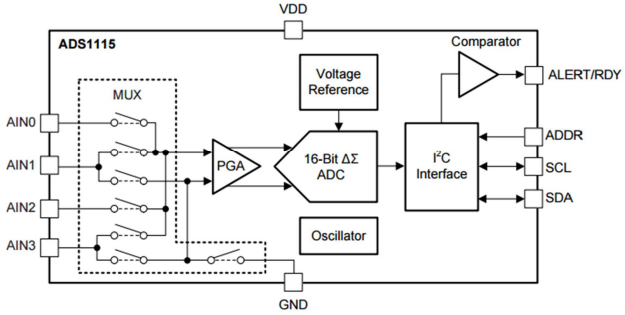

Figura 9. Diagrama de blocs del ADC ADS1115

Per calcular la resolució del sistema (LSB), sabem que:

$$\Delta = \frac{Vref(+) - Vref(-)}{2^n} \qquad \text{(Eq. 1)}$$

La resolució del ADC intern del microcontrolador, és de 10 bits, i les seves entrades són de 0-3,3 Volts, per tant la resolució màxima serà de 3,22 mV.

En el cas del ADC extern de 16 bits amb un ranc de 0-5 Volts, la resolució màxima serà de 0,076 mV.

Per tant el sistema veurà canvis a l'entrada de les analògiques de 10 bits de 3,22 mV i a les analògiques de 16 bits veurà canvis de 0,076 mV, el que ens permetrà obtenir molt bones mesures tenint en compte que el tipus de variables físics més comuns tenen rangs amb 200 valors d'interès, per exemple la temperatura que pot anar de -30 a 50 graus en el cas amb més amplitud tèrmica possible, suposa uns 160 canvis si es vol tenir una resolució de mig grau.

6 ENTRADES/SORTIDES DE SENYALS DIGITALS

Les entrades i sortides digitals ens permetran que el sistema sigui més versàtil ampliant les possibilitats de controlar dispositius que es puguin activar de manera remota via internet o si es volgués segons l'estat de les lectures analògiques, com per exemple una alarma en cas d'alta temperatura o excés d'humitat.

Com el nostre microcontrolador té comunicació I2C, s'ha optat per integrar un port d'expansió d'entrades/sortides per no ocupar les pròpies del microcontrolador que són necessàries per interrupcions per hardware, ja que les que són controlades a través del bus I2C són una mica més lents.

Quan el pin del port d'expansió MCP23008 està configurat com a sortida, pot arribar a donar un màxim de 25mA, donant tensió a una bobina de relé en el cas que es vulgui controlar un element de molta potència.

7 MENÚ D'USUARI

El menú d'usuari s'ha creat per donar la possibilitat d'introduir paràmetres, com l'hora, data, i l'activació de gravació.

Amb el polsador 1 ens anirem movent pels paràmetres que volem canviar i un cop seleccionat el paràmetre, mantenint pressionat el polsador 2 modificarem el valor del paràmetre, tant si és incrementar en nombre de valors, com si es vol saltar entre dos valors d'activació com pot ser on i off. Quan es mostra per pantalla el valor que ens interessa, es deixa de polsar el polsador 2 i es gravarà el valor, ensenyant per pantalla el valor gravat. Aquest menú s'ha implementat dintre del programa del DATALOGGER amb una màquina de 7 estats: 0.Funcionament Normal, 1.Canvi Hora, 2.Canvi minuts, 3.Canvi mes, 4.Canvi dia, 5.Canvi any, 6.on/off gravació. Totes les variables s'incrementen mantenint polsat el polsador 2 i tornen a començar al final del seu cicle com per exemple, el dia, que un cop arribat a 31, torna a zero. En el cas específicament de l'any, s'ha deixat com any màxim 2099, per després tornar a fer la volta començant per 2015.

Per entendre l'estructura del menú d'usuari es seguirà la filosofia de l'estructura que és gràfica a continuació, ja tenint com a previsió la futura ampliació de funcions.

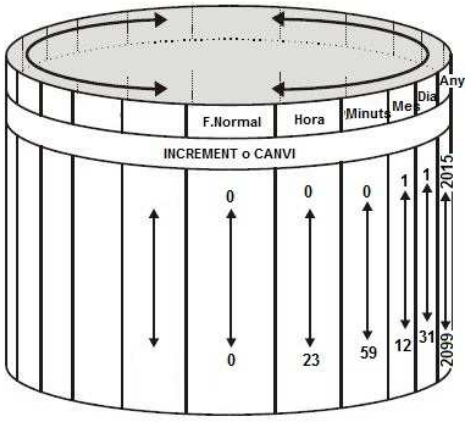

Figura 10. Estructura del menú d'usuari

8 ALIMENTACIÓ DEL SISTEMA

8.1 FONT D'ALIMENTACIÓ

El nostre DATALOGGER porta una font reduïda integrada amb un encapsulat del tipus Relay, de la casa Hi-Link HLK-PM01 de 3W, que ens subministrarà 5 Volts, amb una entrada de 230 Volts AC.

Figura 11. Conjunt regulador 230AC/5DC HLK-PM01

Tot i que posteriorment en aquest document s'ha fet el càlcul per una alimentació amb bateria i panells solars, de manera pràctica s'ha fet amb el Hi-Link de 230AC/5DC per una qüestió de costos i practicitat.

Aquest tipus de font és molt econòmica, de bona estabilitat i amb els seus 600 mA de potència estem per sobre del consum de la nostra placa.

El càlcul per aquesta font s'ha fet tenint en compte una simultaneïtat de "1" a les sortides directes, si cada sortida pot aportar un màxim 25mA, les 8 sortides necessiten 200mA en el pitjor dels casos, i el consum del ESP8266-12E en mode normal d'emissió és de 170mA a màxima potència d'emissió (Tx802.11b, CCK 11Mbps, P OUT=+17dBm), per tant amb els 600mA anem per sobre de la potència total que pot gastar la nostra placa.

En el interior del Hi-Link tenim un IC1/AP8012 que fa la rectificació, i que ens proporciona l'alta eficàcia de transformació de 230AC/5DC de casi un 70 per cent.

En el cas que el DATALOGGER sigui instal·lat en una zona on no es pot accedir a una presa de corrent de 230 AC, s'ha d'alimentar amb una bateria que sigui carregada per panells solars, i s'ha de modificar el programa perquè un cop s'han gravat i enviat les dades que han canviat, entri en mode Deep sleep el microcontrolador. En aquest mode el consum

de 170mA es durant períodes de 3 segons per enviament de dades, per passar a 10uA allargant considerablement la duració de les bateries. En la versió 1 del programa, l'enviament de dades està lligat als canvis, i per això s'ha implementat la font 230AC/5DC en lloc de bateries i panell solar. El càlcul de la bateria s'ha de fer en funció al tipus de variable física a mesurar, és a dir, s'ha de tenir en compte cada quan s'espera que canviï de valor. Si per exemple es vol mesurar temperatura i ens interessa gravar i enviar dades cada cop que canvia un grau, en una zona de clima mediterrani es podria dir que tindrà unes 40 variacions cada 24 hs, per tant el consum per dia serà de 20.4A, el que seria el mateix que 0.85A/hs.

Una vegada seleccionada la bateria, s'ha d'establir la mida de la placa solar. Per això, s'ha de tenir en compte la ubicació del sistema, la radiació solar mitja i la inclinació respecte els mesos de major radiació.

En el nostre cas calcularem la placa com si la instal·lém a Catalunya, al Pirineu a mil metres sobre nivell del mar, amb un angle de 35º, i 5 HPS (hores de llum solar equivalent per dia) segons el fabricant per la zona de Catalunya.

Si triem un panell d'un corrent màxim de 300 mA, llavors serà 5HPS · 300mA=1.5 Ah, el que significa que amb aquest panell ens sobrarà capacitat per alimentar el nostre circuit. El panell solar triat serà un de 4,5W/16V/300mA, encapsulat en una carcassa de policarbonat, hermètic a l'aigua, la humitat, la corrosió. El panell incorpora, internament, un díode de bloqueig per evitar la descàrrega de la bateria, quan la llum solar sigui insuficient, i inclou un suport per orientar-l'ho, que pot ser muntat, indistintament, en la paret, el sòl o el sostre.

9 PROGRAMACIÓ

La programació del microcontrolador del DATALOGGER és en pseudo C, tot i que existeixen diferents llenguatges amb bona informació i llibreries, s'ha optat per pseudo C ja que utilitza una plataforma similar a la de l'entorn Arduino, i no només existeix bona documentació i exemples si no que gràcies al important rerefons que Arduino ha portat al mercat global, s'ha pensat en profunditzar més en aquest tema.

Aquest entorn és de software lliure i codi obert, es troben multitud de llibreries que ens ajudaran molt a optimitzar el programa i crear aplicacions que funcionin més eficaçment.

Aquest tipus de microcontrolador ve de fàbrica amb el firmware per treballar amb comandes del tipus AT. Per facilitar la programació i deixar de banda les comandes AT, és necessari descarregar un plugIn dintre del programa de compilació de l'Arduino que s'encarregarà de fer la transformació de les comandes AT i ens permetra utilitzar llibreries Arduino i el seu compilador. Entre les llibreries que s'han fet servir, podem esmentar les més importants com poden ser les dels protòcols de comunicació SPI i I2C, les pròpies del servidor Broker, i la que ens ajuda a configurar el ESP8266.

Per mantenir un ordre, al programa s'ha intentat separar per funcions, que es poden veure al final del codi. Entre les més rellevants podem esmentar; "grabacion"; realitza la gestió de la SD i la gravació de les dades rebudes, "Lectura_RTC" ; guarda l'hora actual en cada cicle de programa en les variables hora/data, "MQTT_connect" ; realitza la connexió amb el servidor MQTT, "interrup1/2"; Gestionen les interrupcions dels polsadors, "leerMCPport"; s'encarrega de fer la lectura del port d'expansió d'entrades a través del bus I2C, "entrarRTCpulsador"; gestiona el menú d'usuari i la seva màquina d'estats per modificar els paràmetres de configuració, "lectura_ADS"; s'encarrega de capturar les entrades analògiques del ADS1115 guardant-les en variables, "configuraMCP1"; aquesta funció configura els registres interns dels ports d'expansió, "debounceInterrupt"; gestiona el antirrebote dels polsadors i amb les seves interrupcions.

Tot i que la primera versió del programa porta mes de mil línies, sempre es pot depurar mes i ampliar a mes funcionalitats el sistema. La avantatge serà que no caldrà modificar el hardware per millorar el sistema.

9.1 DIAGRAMA DE FLUXE DEL PROGRAMA

L'estructura de l'ordre del flux del nostre programa és; decisió del mode de treball en el setup, configuració del microcontrolador, lectura d'entrades analògiques, aplicació de filtratge de senyal i d'algoritme d'estabilització de mesura, gravació de dades en la SD, i finalment, enviament/recepció de dades via WIFI .

Es comença per carregar les llibreries que configuren; la memòria SD, l'enviament de dades via WIFI, les comunicacions del bus SPI, I2C, i la lectura del rellotge en temps real (RTC). Després es passa a la configuració dels ports d'expansió d'entrades/sortides, per poder prendre la decisió del mode de treball del microcontrolador, sigui com a Broker o Webserver. Posteriorment, en el cas que s'hagi triat mode Broker, es prepara la targeta SD, i ja es pot realitzar les lectures dels senyals analògiques. Amb les lectures analògiques és necessari aplicar un filtratge de soroll, i s'aplica un algoritme per estabilitzar la lectura i no realitzar una gravació de dades repetides. Després si la gravació està activada, s'envien les lectures a la SD per ser gravades i finalment s'envien les dades a la funció de recepció/enviament de dades que s'encarrega d'establir una comunicació amb el servidor Broker per publicar les dades que han canviat, després es realitza la subscripció per veure si s'ha canviat el temps de refresc de cicle o l'activació de la gravació de dades.

En el cas que durant el setup del micro es detecti que s'ha entrat en el mode WEBSERVER, el flux del programa es pràcticament el mateix a diferència de l'enviament de dades, que no es realitza amb publicacions i subscripcions com estableix el protocol MQTT, sinó que en aquest cas el microcontrolador ESP8266-12e actua com ACCESPOINT on es consulta els últims valors dels senyals analògiques mitjançant un CLIENT com pot ser per exemple un mòbil amb wifi.

A la figura nº 12 podem veure el diagrama de flux del nostre programa.

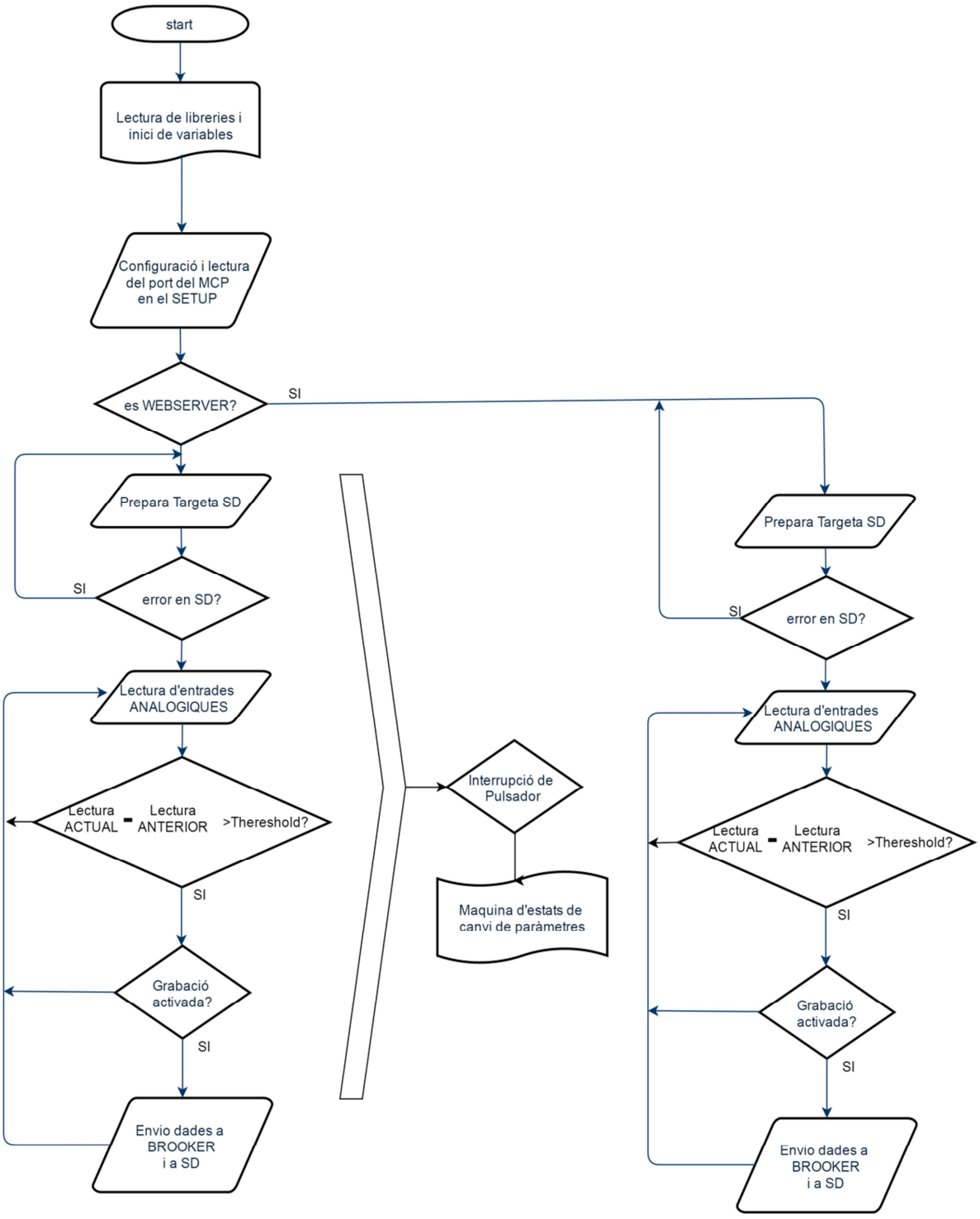

Figura 12. Diagrama de flux de programa

10 CONDICIONS DE FUNCIONAMENT

Les següents consideracions s'apliquen a l'elecció de l'emplaçament d'instal·lació i als requisits sobre l'exposició dels instruments electrònics propis de la placa del DATALOGER a les condiciones mediambientals per prendre dades correctament.

La temperatura normal de funcionament és del rang de -10 a 60 graus, gràcies a les borns de connexió per entrar els senyals analògiques ens dóna la possibilitat de col·locar el sensor lluny de la placa per poder registrar temperatures diferents d'aquest rang.

La instal·lació en entorns del tipus industrial amb sorolls causats per motors o altres aparells de gran potència, pot malmetre la mesura, per això és necessari sortir amb cable de parells trenats i mallat. Tot i que el microcontrolador està tancat amb una caixa metàl·lica connectada a terra per protegir les entrades analògiques com s'ha comentat anteriorment.

El microcontrolador té una potència d'emissió del senyal WIFI teòric segons la taula 1. Traduït a distancia és un abast teòric d'entre 50 i 100 metres, sempre que sigui en línia recta i sense obstacles, en el nostre cas en concret ens ha donat 40 metres amb el DATALOGGER instal·lat dintre d'un bloc de pisos i un mòbil com a client fora de l'edifici i sense altres edificacions pel mig.

Tx Potencia	802.11 b: +20dBm
	802.11 g: +17dBm
	802.11 n: +14dBm
Tx Sensibilitat	802.11 b: -91dBm (11 Mbps)
	802.11 g: -75dBm (54 Mbps)
	802.11 n: -72dBm (MCS7)

Taula 1. Nivell de potència d'emissió/recepció

11 DISSENY DE LA PLACA PCB

El disseny de la versió 1 de la placa PCB es pot veure a la figura nº 14. Aquest s'ha fet amb la filosofia d'integrar tots el components o diferents subplaques que s'havien provat anteriorment per separat sobre Protoboards. Un altre objectiu d'aquesta primera placa, no ha sigut només la integració del conjunt, sinó també reduir el possible soroll amb plans de massa i condensadors extres de desacoblament.

La disposició de les entrades de senyal i alimentacions s'han fet amb borns del tipus KF301 amb cargol, que s'han disposat al voltant de la placa de 10x10 cm per la cara inferior. Les subplaques es connecten mitjançant terminals simples de pas 2.54, i els polsadors també estan integrats en la placa.

El disseny de la placa V2, ha sigut amb la intenció de descartar les subplaques de: RTC, conversió 5/3.3 i ADC, per poder integrar tos els components per separat, millorar el pla de massa, i consolidar una bona disposició dels elements que són intercanviables com pot ser la placa del microcontrolador, la pila del rtc o la SD. També s'han afegit borns per connectar més elements al bus I2C, més punts de controls i una serigrafia més entenedora a l'hora de realitzar les connexions.

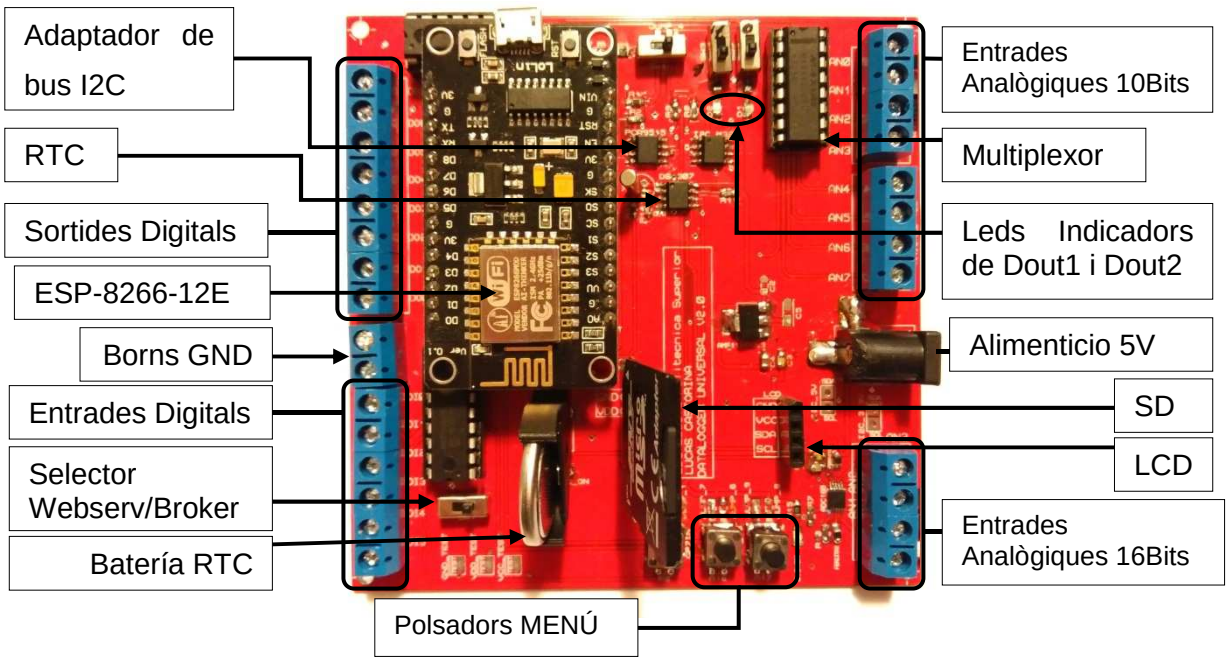

Figura 13. Distribució de components de PCB Versió 2

En el disseny V2 també s'han previst selectors extres i leds d'indicació de funcionament.

El disseny de la placa PCB s'ha fet amb el programa Proteus. Els arxius Gerbers s'han fet amb l'estàndard RS-274X, i l'envio pel seu processament ha sigut amb un proveïdor xines que tot i que proporciona uns terminis d'entrega de fins a un mes i mig, proporciona uns preus molt assequibles baixant els costos de fabricació considerablement.

Tant a la versió 1 com a la versió 2 de les plaques, podem veure que el microcontrolador i la SD són extraïbles, la qual cosa serà molt útil a l'hora de llegir les dades guardades en el cas de la SD o reemplaçar el microcontrolador en cas que estigui defectuós.

Figura 14. Disseny de la PCB versió 1

12 RESUM DEL PRESSUPOST

Per a la realització d'aquest projecte s'ha tingut en compte la utilització de components accessibles d'una manera fàcil i barata. Els preus s'han ajustat al màxim possible per poder dur a terme un projecte econòmicament viable, hem de tenir en compte que aquests preus són per un únic prototip i no s'ha tingut en compte les hores de programació que s'haurien de repercutir de forma distribuïda en cada aparell venut, en funció de la previsió de vendes. També cal esmentar que fent una producció en massa del producte baixaria considerablement el seu cost.

El cost de la realització pràctica d'aquest projecte és de cent cinquantè-set amb norantè set cèntims, sense IVA.

13 CONCLUSIONS

Un cop realitzat el projecte i veient el resultat obtingut, es pot afirmar que s'han aconseguit els objectius plantejats a l'inici d'aquest. S'ha realitzat una correcta adquisició dels senyals analògiques, un emmagatzematge precís d'informació recollida després de passar pel filtrat i tant la recepció com l'enviament de les dades s'ha pogut veure online correctament.

S'ha complert amb els objectius de comunicació, enregistrament de dades, simplicitat de funcionament i baix cost de fabricació. Hem vist que per aplicacions on l'enregistrament de dades es fa amb un fitxer del tipus txt, no és necessari la creació de softwares instal·lats en el PC que comuniquin i facin d'interpretació de les dades, reduint més els costos de I+D.

Totes les solucions aquí recollides es basen en l'intent de realitzar un sistema el més modern, versàtil i simple possible, per satisfer les necessitats tecnològiques que aquí s'han presentat, sense incórrer en un cost excessiu. Aquest no és un sistema massa generalitzat al mercat, ja que majoritàriament els DATALOGGERs tenen funcions més especifiques, la qual cosa pot obrir noves vies tecnològiques, essent possible que es vegi sotmès a possibles millores de software i de supressió d'EMIs (Interferències electromagnètiques) per poder treballar en entorns industrials més exigents.

El nostre programa final ha tingut més de mil línies i sempre es podrà millorar i depurar. El hardware no serà necessari canviar-lo. Aquesta és un avantatge molt gran a l'hora de comercialitzar el producte, ja que permet resoldre futurs problemes o defectes detectats per l'ús de milers de clients i també permet actualitzar el software amb possibles millores que havien quedat pendents en el moment del llançament del producte.

Lucas Ariel Castorina

Graduat en Enginyeria Electrònica i Automàtica Industrial.

Girona, 6 de gener de 2017

14 RELACIÓ DE DOCUMENTS

Els documents del projecte són: memòria, plànols, plec de condicions, estat d'amidaments i pressupost.

15 BIBLIOGRAFIA

Acrobotic (https://acrobotic.com/featured/acr-00018, 1 de novembre de 2016)

Adafruit Inc. (https://learn.adafruit.com/category/adafruit-io, 12 de novembre de 2016)

Analog Devices (http://www.analog.com/en/index.html, 29 de novembre de 2016)

Arduino (https://www.arduino.cc/en/Reference/HomePage, 1 de novembre de 2016)

Bonifacio Martín Del Brío, Sistemas electrónicos basados en microprocesadores y Microcontroladores, Prensas Universitarias de Zaragoza 1994.

Malvino Albert Paul, Principios de Electrònica, sexta edició, Mc Graw Hill, 2000

Microchip Technology Inc. (http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC18F2550, 15 de setembre de 2016)

Wikipedia (https://en.wikipedia.org/wiki/Data_logger, 23 de setembre de 2016)

16 GLOSARI

EMI: Electro Magnetic Interferences, Interferències electromagnètiques.

I+D: Investigació i desenvolupament

LSB: Less significant Bits, Bit menys significatiu.

PC: Personal Computer, ordinador.

A PROGRAMA

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>
Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */
#include <RTClib.h>
RTC_DS1307 RTC;
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <SPI.h>
#include <SD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR    0x20 // <<-Dirección del esclavo A1/2/3 a masa del LCD
#define BACKLIGHT_PIN     3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

/************************ WiFi Access Point *********************************/
const char* WLAN_SSID = "JAZZTEL-QMGDLN";
const char* WLAN_PASS = "3QEJCR4S";////
//const char WLAN_SSID = "JAZZTEL-QMGDLN";
//const char WLAN_PASS = "3QEJCR4S";
const char* WLAN_SSID_2 = "lucad";
const char* WLAN_PASS_2 = "qwertyuiop";

/************************ Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "lucascastorina"
#define AIO_KEY         "c066edc9cc8b4a7f92c7e4c361fd5c20"

/************ Global State SIN CAMBIOS ******************/

WiFiClient client;// Create an ESP8266 WiFiClient class to connect to the MQTT
server.


// Store the MQTT server, username, and password in flash memory.
// This is required for using the Adafruit MQTT library.
const char MQTT_SERVER[] PROGMEM    = AIO_SERVER;
const char MQTT_USERNAME[] PROGMEM  = AIO_USERNAME;
const char MQTT_PASSWORD[] PROGMEM  = AIO_KEY;
```

```
// Setup the MQTT client class by passing in the WiFi client and MQTT server and
login details.
Adafruit_MQTT_Client   mqtt(&client,   MQTT_SERVER,   AIO_SERVERPORT,MQTT_USERNAME,
MQTT_PASSWORD);

/*****************************                    CREACION                    DE
Feeds************************************/
/*****************************************************************************
********/
////////////////////////////////            Setup            feeds            for
PUBLISHING/////////////////////////////
//////////     Recordar     que     MQTT     paths     for     AIO     follow     the
form:<username>/feeds/<feedname>
/*****************************************************************************
*********/
//////////////////////////////ESP-->INTERNET
(PUBLISHING)/////////////////////////////
/*****************************PUBLICACION
ANALOGICA*************************************/
const char sensor0_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor0";
Adafruit_MQTT_Publish sensor0 = Adafruit_MQTT_Publish(&mqtt, sensor0_FEED);
const char sensor1_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor1";
Adafruit_MQTT_Publish sensor1 = Adafruit_MQTT_Publish(&mqtt, sensor1_FEED);
const char sensor2_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor2";
Adafruit_MQTT_Publish sensor2 = Adafruit_MQTT_Publish(&mqtt, sensor2_FEED);
const char sensor3_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor3";
Adafruit_MQTT_Publish sensor3 = Adafruit_MQTT_Publish(&mqtt, sensor3_FEED);
const char sensor4_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor4";
Adafruit_MQTT_Publish sensor4 = Adafruit_MQTT_Publish(&mqtt, sensor4_FEED);
const char sensor5_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor5";
Adafruit_MQTT_Publish sensor5 = Adafruit_MQTT_Publish(&mqtt, sensor5_FEED);
const char sensor6_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor6";
Adafruit_MQTT_Publish sensor6 = Adafruit_MQTT_Publish(&mqtt, sensor6_FEED);
const char sensor7_FEED[] PROGMEM = AIO_USERNAME "/feeds/sensor7";
Adafruit_MQTT_Publish sensor7 = Adafruit_MQTT_Publish(&mqtt, sensor7_FEED);

/*****************************PUBLICACION
DIGITAL*************************************/
const char PUSH_FEED[] PROGMEM = AIO_USERNAME "/feeds/push"; //OJO igual nombre que
en adafruit.iofeed
Adafruit_MQTT_Publish push1 = Adafruit_MQTT_Publish(&mqtt,PUSH_FEED);//push a secas
sera variable
//de entrada al micro cuando se apreta

/*****************************************************************************
*********/
//////////////////////////////////INTERNET-->ESP
(SUBCRIBING)/////////////////////////////
/*****************************************************************************
*********/
```

```
////////////////////////////////DIGITALES                                (INTERNET--
>ESP)//////////////////////////////
const char ONOFF_FEED[] PROGMEM = AIO_USERNAME "/feeds/onoff";
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt,ONOFF_FEED);
const char lamp_FEED[] PROGMEM = AIO_USERNAME "/feeds/lamp";
Adafruit_MQTT_Subscribe lampbutton = Adafruit_MQTT_Subscribe(&mqtt, lamp_FEED);

////////////////////////////////ANALOGICAS(INTERNET--
>ESP)//////////////////////////////
// Setup a feed called 'analogica1' for subscribing to changes.
const char analogica1_FEED[] PROGMEM = AIO_USERNAME "/feeds/analogica1";
Adafruit_MQTT_Subscribe                    analogica1                    =
Adafruit_MQTT_Subscribe(&mqtt,analogica1_FEED);
void MQTT_connect(); // Function to connect and reconnect as necessary to the MQTT
server.
////////////////////////////////INICIO
VARIABLES//////////////////////////////////////
volatile int m=1;//volatile para que se copie el flanco de interrupcion sin mirar
registros
volatile int mant=0;
int tiempo = 1000;
int p[8];//Array de diferencias
int sensorValueANT[8]={0,0,0,0,0,0,0,0};//Array de lecturas anteriores
int sensorValue[8]={0,0,0,0,0,0,0,0};//Array de lecturas
int largoparam=0;
//String parametros[];
char parametros[80];
int s=0 ;//para maquina de estados, cambia de estado con interrupcion de pulsador
bool GrabaStatus=1;//Si GrabaStatus=1 me graba
File myFile;
byte MCPportIN= 0;
int hora;
int minuto;
int segundo=0;
int any;
int mes;
int dia;
bool GRABon=1;
int ADS_in0, ADS_in1, ADS_in2, ADS_in3;
const uint8_t MCP1address   = 0x21; //segun se configura A0 A1 A2
//const uint8_t MCP2address   = 0x22; //segun se configura A0 A1 A2
bool gp0,gp1,gp2,gp3,gp4,gp5,gp6,gp7;
unsigned long previousMillis = 0;        // will store last time LED was updated
const long interval = 100;
WiFiServer server(80);

//////////////////////////////////////////////              VOID              SETUP
//////////////////////////////////////////////
/*********************************************************************************
*****************/
void setup() {
static const uint8_t D1   = 5;//Asignacion de pines de ESP8266 DEV BOARD
```

```
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;//PUSH BUTTON INTERRUP
static const uint8_t D4   = 2;//PUSH BUTTON INTERRUP
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;//chp select del SPI

Serial.begin(115200);
lcd.begin (16,2); //  <<--- pantalla LCD 16x2
lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE); // Enciende el backlight
lcd.setBacklight(HIGH);
RTC.begin(); // Inicia la comunicación con el RTC
Wire.begin(D2,D1); // Inicia el puerto I2C, SDA scl
delay(100);
 ////////////////////////////////SETUP        DEL        ADC        de        16bits
//////////////////////////////////////
 //                                                                  ADS1115
 ads.setGain(GAIN_TWOTHIRDS);   // 2/3x gain +/- 6.144V   1 bit =          0.1875mV
(default)
//con gain=2/3 se pierden bit desde 26683 al 32767 pero se cubren los 0-5 volts
//ads.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit =            0.125mV
//ads.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit =       0.0625mV
//ads.setGain(GAIN_FOUR);       // 4x gain   +/- 1.024V  1 bit =       0.03125mV
//ads.setGain(GAIN_EIGHT);      // 8x gain   +/- 0.512V  1 bit =       0.015625mV
//ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit =       0.0078125mV
  ads.begin();

////////////////////////////////LECTURA          SD          PARAMETROS          de
para.txt//////////////////////
if (!SD.begin(D8)) { //Comienzo con chip select
   Serial.println( ); Serial.println("ERROR en inicio de SD ");
   }else{
  Serial.println("Inicio de SD OK.");
  Serial.println("Lectura de parametros de inicio (para.txt)");
    }
myFile = SD.open("para.txt");
  if (myFile) {
    while (myFile.available()) {
      // parametros+=myFile.read();
       parametros[largoparam]=myFile.read();
       largoparam=largoparam+1;
    }
    myFile.close();
    Serial.print("Parametros Inicio:");
       Serial.println(parametros);
       Serial.print("Control parametros:");
       Serial.println(largoparam);
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
```

```
configuraMCP1(MCP1address);//CONFIGURACION DEL MCP1 como entrada
//configuraMCP2(MCP2address);//CONFIGURACION DEL MCP1 como salidas
//////////////////////Asignacion de entradas
INTERRUPCIONES////////////////////////////////
pinMode(D4, INPUT_PULLUP); // PUSH button de interrupcion
attachInterrupt(digitalPinToInterrupt(D4),interrup1,RISING);//subrutina
interrupcion iniciada por push D4
pinMode(D3, INPUT_PULLUP); // PUSH button de interrupcion
attachInterrupt(digitalPinToInterrupt(D3),interrup2,RISING);
  Serial.println(); // Connect to WiFi access point.
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  lcd.setCursor (0,0);
  lcd.print("CONECTANDO....");
  lcd.setCursor (0,1);
  lcd.print(WLAN_SSID);
 delay(500);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
      }

  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  lcd.setCursor (0,0);
  lcd.print("WIFI CONECTADO");
  lcd.setCursor (0,1);
  lcd.print(" ");
  delay(2500);
  lcd.setCursor (0,0);
  lcd.print("IP address:    ");
  lcd.setCursor (0,1);
  lcd.print("               ");
  lcd.setCursor (0,1);
  lcd.print(WiFi.localIP());
  delay(2000);

  /**********************************  Setup  MQTT  subscription  for  feeds.
**********************/
 ////////////////////////////////////AGREGAR LAS SUSCRIPCIONES
CREADAS//////////////////////////
  mqtt.subscribe(&onoffbutton);
  mqtt.subscribe(&analogica1);
  //mqtt.subscribe(&lampbutton);// no se leen mas cosas del broker


}//ACABA VOIP SETUP
```

```
/*////////////////////////////////////////                                    VOIP
LOOP//////////////////////////////////////////*/
/*****************************************************************************
***************/
void loop() {
MCPportIN = leerMCPport(MCP1address);//MCPport es el estado del puerto del MCP,de
izquierda a derecha-->7..0
//Serial.print("Todo el puerto LOOP= ");
//Serial.println(MCPportIN,BIN);
gp6=bitRead(MCPportIN,6);
Serial.print("WebServer en LOOP= ");
Serial.println(gp6,BIN);

while (gp6==1){
Serial.println("WHILE DE gp6==1");
lectura_RTC();//Llamada a funcion de lectura RTC
entrarRTCpulsador(MCP1address);
MQTT_connect();//Llamada funcion de reconexion al servidor
lectura_ADS();
Serial.print("AIN0: "); Serial.println(ADS_in0); //de 0-26600

//***************************          BUCLE          de          envio          de
DATOS**********************************/
Adafruit_MQTT_Subscribe       *subscription;//LECTURA       Y       ENVIO       DE
TIEMPOS////////////////////
while ((subscription = mqtt.readSubscription(tiempo))) {//1 seg esta BIEN pero se
puede probar menos
 ////////////////////////////////SUSCRIPCION           DIGITALES(INTERNET-->ESP)
//////////////////////////////////////
 if (subscription == &onoffbutton) {//Se LEE variable del broker y se desac/activa
la  GRABACION
      Serial.print(F("GRABACION: "));
      Serial.println((char *)onoffbutton.lastread);
      char *value = (char *)onoffbutton.lastread;
      int current = atoi(value);
          if (current == 1 ){
               //Escribir gp7=1 del puerto del MCPout para indicar LED qie se graba
o no
              GrabaStatus=1;
               }else{
                //Escribir gp7=0 del puerto del MCPout
              GrabaStatus=0;
                    }
                     }
////////////////////////////SUSCRIPCION           ANALOGICA           (INTERNET--
>ESP)//////////////////////////////////////
       if (subscription == &analogica1) { //
      Serial.print(F("analogica1: "));
      Serial.println((char *)analogica1.lastread);
      char *valueT = (char *)analogica1.lastread;
      tiempo = atoi(valueT);
     Serial.println(tiempo);
```

```
    }
  }//FIN WHILE de readSubscription
//////////////////////////////////////////////LECTURA A0
MULTIPLEXADA//////////////////////////////
int leerMUX[8];
Serial.end();////PARA PODER USAR SALIDAS TX RX
delay(10);
static const uint8_t RX   = 3;   //MUX_B
static const uint8_t TX  = 1;   //MUX_A
static const uint8_t D0 = 16;//MUX--C
 pinMode(TX, OUTPUT); pinMode(D0, OUTPUT); pinMode(RX, OUTPUT);
for (int abc=0; abc <= 7; abc++){
  int r=50;
if (abc ==0){
  digitalWrite(D0, 0);
  digitalWrite(RX, 0);
  digitalWrite(TX, 0);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
  }
if (abc==1){
  digitalWrite(D0, 0);
  digitalWrite(RX, 0);
  digitalWrite(TX, 1);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
if (abc==2){
  digitalWrite(D0, 0);
  digitalWrite(RX, 1);
  digitalWrite(TX, 0);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
if (abc==3){
  digitalWrite(D0, 0);
  digitalWrite(RX, 1);
  digitalWrite(TX, 1);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
if (abc==4){
  digitalWrite(D0, 1);
  digitalWrite(RX, 0);
  digitalWrite(TX, 0);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}

if (abc==5){
  digitalWrite(D0, 1);
  digitalWrite(RX, 0);
```

```
  digitalWrite(TX, 1);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
if (abc==6){
  digitalWrite(D0, 1);
  digitalWrite(RX, 1);
  digitalWrite(TX, 0);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
if (abc==7){
  digitalWrite(D0, 1);
  digitalWrite(RX, 1);
  digitalWrite(TX, 1);
  delay(r);
  leerMUX[abc] = analogRead(A0);//Lectura de entrada analogica
}
}//FIN FOR
Serial.begin(115200); ////PARA PODER USAR SALIDAS TX RX
delay(200);///////////PROBAR TIEMPOS DE LECTURA diferentes

////////////////////////////////////FIN                LECTURA                A0
MULTIPLEXADA//////////////////////////////
for (int i=0; i <= 7; i++){
  Serial.print("leerMUX_");
  Serial.print(i);
  Serial.print(":");
  Serial.println(leerMUX[i]);
}

 ////////////////////////////////////////////ESP8266--
>INTERNET//////////////////////////

/*****************************************************************************
*********/
  ////////////////////////////////////           ANALOGICA(ESP8266-->INTERNET)
//////////////////
for(int i=0; i <= 7; i++){////BUCLE ENVIO DE DATOS y grabacion de analogicas
     sensorValue[i]=leerMUX[i];
    p[i]=sensorValueANT[i]-sensorValue[i];
    p[i]=abs(p[i]);
    Serial.print("DIFERENCIA: ");Serial.println(p[i]);
              if (p[i]>20){
    //ENVIARA si la diferencia es MAYOR a algo para filtrar error de lectura
         switch (i){
         case 0:
         if (! sensor0.publish(sensorValue[0])) {//Se publica el valor y devuelve
true/false
                    Serial.println(F("Failed"));
  lcd.setCursor  (0,0);lcd.print("ERROR  EN  EL  ENVIO         ");  lcd.setCursor
(0,1);lcd.print("DE DATOS     ");
```

```
  } else {
   Serial.print(F("\nSending                          ANALOG                          IN
"));Serial.print(i);Serial.print(":");Serial.println(sensorValue[i]);

  lcd.setCursor (0,0);lcd.print("Enviando entrada.....");
  lcd.setCursor                                        (0,1);lcd.print("ANALOGin_");
lcd.print(i);lcd.print(":");lcd.print(sensorValue[i]);
  lcd.print("   ");
                          if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                         grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                               }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;    //Para que salga del bicle for directo
        }
      case 1:
        if (! sensor1.publish(sensorValue[1])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                  Serial.print(F("\nSending   ANALOG   IN   "));   Serial.print(i);
Serial.print(":");Serial.println(sensorValue[i]);
                         if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                         grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                               }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
         case 2:
        if (! sensor2.publish(sensorValue[2])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                   Serial.print(F("\nSending    ANALOG    IN    "));Serial.print(i);
Serial.print(":"); Serial.println(sensorValue[i]);
                         if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                         grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                               }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
       case 3:
       if (! sensor3.publish(sensorValue[3])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                   Serial.print(F("\nSending         ANALOG         IN         "));
Serial.print(i);Serial.print(":");Serial.println(sensorValue[i]);
```

```
                        if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                        grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                              }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
        case 4:
        if (! sensor4.publish(sensorValue[4])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                   Serial.print(F("\nSending ANALOG IN "));
                   Serial.print(i);
                   Serial.print(":");
                   Serial.println(sensorValue[i]);
                        if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                        grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                              }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
       case 5:
       if (! sensor5.publish(sensorValue[5])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                   Serial.print(F("\nSending ANALOG IN "));
                   Serial.print(i);
                   Serial.print(":");
                   Serial.println(sensorValue[i]);
                        if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                        grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                              }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
       case 6:
       if (! sensor6.publish(sensorValue[6])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                  Serial.print(F("\nSending ANALOG IN "));
                   Serial.print(i);
                   Serial.print(":");
                   Serial.println(sensorValue[i]);
                        if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                        grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                              }
```

```
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
       case 7:
       if (! sensor7.publish(sensorValue[7])) {//Se publica el valor y devuelve
true/false
                   Serial.println(F("Failed"));
                   } else {
                   Serial.print(F("\nSending ANALOG IN "));
                   Serial.print(i);
                   Serial.print(":");
                   Serial.println(sensorValue[i]);
                         if (GrabaStatus==1){ //grabara datos mientras no haya
pulsado  LED D3=ON
                         grabacion (sensorValue[i],i); //FUNCION GRABAR DATOS
                               }
                  sensorValueANT[i]=sensorValue[i];
                  goto sigue;
        }
        }//FIN CASE

}//FIN PRIMER IF

}//FIN BUCLE FOR ENVIO DATOS y grabacion de analogicas


 ////////////////////////////////////             DIGITAL             (ESP8266--
>INTERNET)//////////////////////////////
sigue:
     if (m!=mant){ //if para activar indicador en web cuando se pulsa en D3
        if (! push1.publish(m)) {//Se publica el valor y devuelve true/false
        Serial.println(F("Failed"));
        } else {
        Serial.println(F("envio correcto OK!"));
        }
  mant=m;
  }
MCPportIN = leerMCPport(MCP1address);//MCPport es el estado del puerto del MCP,de
izquierda a derecha-->7..0
gp6=bitRead(MCPportIN,6);
}//FIN WHILE gp6==1
int r=1;//Para hacer el setup del webserver
while (gp6==0){
    if (r=1){
 Serial.print("Conectando webserver con ");
  Serial.println(WLAN_SSID_2);
  WiFi.begin(WLAN_SSID_2, WLAN_PASS_2);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
```

```
  Serial.println("WiFi conectado");
r=2;
  server.begin();
  Serial.println("Inicio de Server");
  //Print the IP address
  Serial.print("La URL de conexion es: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}//FIN if de conexion
  MCPportIN = leerMCPport(MCP1address);//MCPport es el estado del puerto del MCP,de
izquierda a derecha-->7..0
gp6=bitRead(MCPportIN,6);
 Serial.println("WHILE DE gp6==0");

Serial.print("Conectando webserver con ");
  Serial.println(WLAN_SSID_2);
  WiFi.begin(WLAN_SSID_2, WLAN_PASS_2);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  server.begin();
  Serial.println("Inicio de Server");
  //Print the IP address
  Serial.print("La URL de conexion es: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");
 }//FIN while gp6==0
}//FIN VOID LOOP GRAL

/*****************************************FUNCIONES*****************************
**************/
////////////////////////////////////funcion de configuracion del MCP como
ENTRADA
void configuraMCP1 (const uint8_t MCP1address){
Wire.beginTransmission(MCP1address); //begins talking to the slave device
Wire.write(0x00); //selects the IODIRA register
Wire.write(0B11111111); //1=inputs
Wire.endTransmission(); //stops talking to device
Wire.beginTransmission(MCP1address); //begins talking to the slave device
Wire.write(0x01); //Regitro IPOL inversion de entrada
Wire.write(0B10000000); //Si tengo 0=activo al quitar GND si tengo 1=activo si
coloco GND
Wire.endTransmission(); //stops talking to device
Wire.beginTransmission(MCP1address); //begins talking to the slave device
Wire.write(0x06); //Regitro GPPU activa Rs pullup entrada
Wire.write(0B11111111); //Todas entradas con Pullup
```

```
Wire.endTransmission(); //stops talking to device
}

//////////////////////////////////////funcion de lectura de entradas analogicas
de mayor precicion
void lectura_ADS(){

  ADS_in0 = ads.readADC_SingleEnded(0);
  ADS_in0= (float)ADS_in0/0.536600;//ESCALADO 5volts=50000 esta sin la COMA
  ADS_in1 = ads.readADC_SingleEnded(1);
  ADS_in1= (float)ADS_in1/0.536600;
  ADS_in2 = ads.readADC_SingleEnded(2);
  ADS_in2= (float)ADS_in2/0.536600;
  ADS_in3 = ads.readADC_SingleEnded(3);
  ADS_in3= (float)ADS_in3/0.536600;
  //Serial.print("ADS_AIN0: "); Serial.println(ADS_in0); //de 0-26600
  //Serial.print("ADS_AIN1: "); Serial.println(ADS_in1);
  //Serial.print("ADS_AIN2: "); Serial.println(ADS_in2);
  //Serial.print("ADS_AIN3: "); Serial.println(ADS_in3);
  //Serial.println(" ");
}

//////////////////////////////////////////////Segun  se  pulsa  se  entran  los
parametros del RTC
void entrarRTCpulsador(const uint8_t MCP1address){
if  (s==1){
  Serial.println("PULSE PARA INCREMENTAR HORA");
    while (bitRead (MCPportIN,7)==0 and s==1   ){//and hora<26al poner a gnd el
gpio7 va incrementando
      Serial.print("DEJE DE PULSAR PARA GUARDAR HORA=  ");
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera0;//Si no esta el goto suma un al salir del while
    }
    hora=hora+1;
    Serial.println(hora,DEC);
    delay(700);
     if (hora==24){
      hora=0;
    }
  }
fuera0:
RTC.adjust(DateTime(any, mes, dia, hora, minuto, 11));
    Serial.print("hora introducida:  ");
    Serial.println(hora,DEC);
    delay(2000);
  }

if  (s==2){
  Serial.println("PULSE PARA INCREMENTAR MINUTOS");
    while (bitRead (MCPportIN,7)==0 and s==2 ){
      Serial.print("DEJE DE PULSAR PARA GUARDAR MINUTOS=  ");
```

```
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera1;//Si no esta el goto suma un al salir del while
    }
    minuto=minuto+1;
    Serial.println(minuto,DEC);
    delay(400);
     if (minuto==61){
      minuto=0;
    }
  }
   fuera1:
   RTC.adjust(DateTime(any, mes, dia, hora, minuto, 0));
      Serial.print("MINUTOS introducidos:  ");
    Serial.println(minuto,DEC);
    delay(2000);
  }
if  (s==3){
  Serial.println("PULSE PARA INCREMENTAR DIA");
    while (bitRead (MCPportIN,7)==0 and s==3  ){
      Serial.print("DEJE DE PULSAR PARA GUARDAR DIA=  ");
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera3;//Si no esta el goto suma un al salir del while
    }
    dia=dia+1;
    Serial.println(dia,DEC);
    delay(400);
     if (dia==32){
      dia=0;
    }
  }
    fuera3:
    RTC.adjust(DateTime(any, mes, dia, hora, minuto, 0));
    Serial.print("dia introducido:  ");
    Serial.println(dia,DEC);
    delay(2000);
  }
  if  (s==4){
  Serial.println("PULSE PARA INCREMENTAR MES");
    while (bitRead (MCPportIN,7)==0 and s==4 ){
      Serial.print("DEJE DE PULSAR PARA GUARDAR MES=  ");
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera4;//Si no esta el goto suma un al salir del while
    }
    mes=mes+1;
    Serial.println(mes,DEC);
    delay(400);
    if (mes==13){
      mes=0;
    }
```

```
  }
   fuera4:
   RTC.adjust(DateTime(any, mes, dia, hora, minuto, 11));
   Serial.print("mes introducido:  ");
    Serial.println(mes,DEC);
    delay(2000);
  }

 if  (s==5){
  Serial.println("PULSE PARA INCREMENTAR ANY");
    while (bitRead (MCPportIN,7)==0 and s==5 ){
      Serial.print("DEJE DE PULSAR PARA GUARDAR ANY=  ");
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera5;//Si no esta el goto suma un al salir del while
    }
    any=any+1;
    Serial.println(any,DEC);
    delay(400);
    if (any>2050){
      any=2000;
    }
  }
   fuera5:
   RTC.adjust(DateTime(any, mes, dia, hora, minuto, 11));
   Serial.print("any introducido:  ");
    Serial.println(any,DEC);
    delay(2000);
  }
if  (s==6){
  Serial.println("PULSE PARA DESACTIVAR GRABACION");
    while (bitRead (MCPportIN,7)==0 and s==6 ){
     // Serial.print("DEJE DE PULSAR PARA GUARDAR ANY=  ");
    MCPportIN = leerMCPport(MCP1address);
    if (bitRead (MCPportIN,7)==1 ){
      goto fuera6;//Si no esta el goto cambia al salir del while
    }
    GRABon=!GRABon;
    delay(1500);
 fuera6:
    if (GRABon==1){
      Serial.println("GRABACION ACTIVADA");
      delay(1500);
    }else{
     Serial.println("GRABACION DESACTIVADA");
     delay(1500);
      }
  }
     }
}//FIN FUNCION entrarRTCpulsador

//////////////////////////////////////////FUNCION LECTURA PUERTO MCP por I2C
```

```
byte leerMCPport (const uint8_t MCPaddress){
Wire.beginTransmission(MCPaddress); //starts talking to slave device
Wire.write(0x09); //selects the GPIO pins
Wire.requestFrom(MCPaddress,1);//Se lee un BYTE del MCP1
byte MCPportx = Wire.read(); //MCPport me guarda el puerto del MCP
Wire.endTransmission(); //ends communication with the device
return MCPportx;
}

//////////////////////////////////////////////////Funcion Rutina de interrupcion
pulsador

void interrup1(){
  s=s+1;
  Serial.print("s interrupt = ");
  Serial.println(s);
if (s>=7){//8 ESTADOS maximo
  s=-1;
}
}//fin interrup

void interrup2(){
m=!m;
  Serial.print("m interrupt = ");
  Serial.println(m);
}//fin interrup

//////////////////////////////////////////////////////// Funcion de reconexion
a MQTTserver
void MQTT_connect() {
  int8_t ret;  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }
  Serial.print("Connecting to MQTT... ");
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}

//////////////////////////////////////////////////////////////////Funcion    lectura
RTC
```

```
void lectura_RTC()
{
DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
Serial.print(now.year(), DEC); // Año
any=now.year();
//Serial.print(any);////para aislar la variable
Serial.print(' ');
Serial.print('/');
Serial.print(' ');
Serial.print(now.month(), DEC); // Mes
mes=now.month();
Serial.print(' ');
Serial.print('/');
Serial.print(' ');
Serial.print(now.day(), DEC); // Dia
dia=now.day();
Serial.print(' ');
Serial.print(' ');
Serial.print(' ');
Serial.print(now.hour(), DEC); // Horas
hora=now.hour();
Serial.print(':');
Serial.print(now.minute(), DEC); // Minutos
minuto=now.minute();
Serial.print(':');
Serial.print(now.second(), DEC); // Segundos
segundo=now.second();
Serial.println();
//delay(500); // La información se actualiza cada 1 seg.
}

////////////////////////////////ESCRIBIENDO               DATOS               en
SD//////////////////////////
void grabacion (int DATO, int i)//DATO es el valor del sensor
{
     // File myFile;
      String nomarx ; //variable para darle nombre al archivo que se guarda en la
sd
      nomarx +=any;//se concatenan variables para crear nombre de archivo
      nomarx +=mes;//año/mes/dia
      nomarx +=dia;
      nomarx+=".txt";
      myFile = SD.open(nomarx, FILE_WRITE);
        if (myFile) {  //si se abre el archivo me escribe los datos
        myFile.print(hora);
        myFile.print(":");
        myFile.print(minuto);
        myFile.print(":");
        myFile.print(segundo);
        myFile.print(" Sens");
        myFile.print(" ");
        myFile.print(i);
```

```
        myFile.print(" ");
        myFile.print(":");
        myFile.print(" ");
        myFile.println(DATO);
        myFile.close();
        Serial.println("...CAMBIOS GRABADOS OK");
            } else {
        // if the file didn't open, print an error:
        Serial.println("ERROR de GRABACION");
        Serial.println(nomarx);
  }

}
```