Servei per controlar sistemes encastats

E.T. Informàtica de Sistemes

Estudiant

David Otal Esclarín

Consultor

Jordi Becares Ferres

Data Lliurament 2 de juliol de 2015

Dedicatòria i agraïments

Vull dedicar i donar les gràcies especialment a la meva dona i al meu fill, als que estimo, i no he pogut donar la dedicació desitjada durant la duració del projecte.

Gràcies també a la meva família en general, pel seu suport, confiança i paraules d'ànim.

Gràcies a tots els meus mestres, tutors i companys, amb què he pogut intercanviar i ampliar coneixements per fer possible aquest projecte.

I finalment gracies a tu per follejar aquest document, i interessar-te per aquest projecte.

Gràcies a tots.

Resum

Aquest projecte s'engloba en l'àmbit dels sistemes encastats. Un sistema encastat es defineix com un sistema informàtic, encapsulat dins d'un dispositiu, amb una funcionalitat especifica. Els avenços tecnològics han permès reduir la mida d'aquests dispositius i millorar les seves prestacions, a la vegada que s'han reduït els seus costos de fabricació. És per això que han tingut una gran expansió els darrers anys, i ara estan presents en quasi tots els aparells i dispositius moderns.

Aquest projecte neix amb el propòsit de, fent ús d'un PC i un microcontrolador, facilitar el desenvolupament d'eines per controlar o treballar amb alguns d'aquests dispositius. Alguns exemples pràctics per als quals es podria fer servir, són l'obtenció de dades de una estació meteorològica, el control d'una casa domòtica, control d'alarmes i seguretat, control d'un panell de llums, control de fonts amb llums de colors, control de reg al camp, robòtica, automatismes, control de maquinària i braços mecànics, etc.

Per fer això es desenvoluparà un servei de funcionalitats per a PC, que mitjançant el microcontrolador, permetrà obtenir dades o enviar instruccions als diversos dispositius connectats a la placa ja sigui físicament o per wifi, i de forma transparent a l'usuari final. Per fer possible aquest servei, el projecte es dividirà en 3 parts ben diferenciades:

1. Desenvolupament d'un nou protocol de comunicació entre un PC i un microcontrolador.
2. Implementació d'una API que ofereixi el servei al PC, fent ús del protocol.
3. Codificació del Firmware del microcontrolador, per tal de resoldre les peticions que li arribin via protocol.

Es desenvoluparà també una eina bàsica com a exemple d'us del servei, i per a fer proves del sistema, però sense oferir la funcionalitat final, ja que aquesta dependrà de l'ús o de les necessitats dels clients finals.

Per tant està orientat a totes aquelles persones que vulguin o requereixin desenvolupar un programari de PC per controlar aquesta mena dispositius, sense invertir massa temps ni recursos, ni requerir grans coneixements dels detalls tècnics. Els requisits que s'hauran d'acomplir serà la correcta connexió dels dispositius, la instal·lació del programari que ofereix aquest servei, i el desenvolupament o adaptació de la aplicació final que farà servir el servei.

Índex de continguts

Índex de figures

1. Introducció.

El projecte a desenvolupar consisteix en la implementació d'un protocol de comunicació entre un PC i una placa amb capacitat d'interacció amb el món físic mitjançant sensors i actuadors, en el nostre cas farem servir un LPC, per tal de poder recollir dades de sensors i fer servir actuadors. Per fer ús del protocol es desenvoluparà una API per la banda del PC, i un mòdul firmware per la part del LPC. La idea és que serveixi com a nivell d'abstracció del maquinari, per facilitar el desenvolupament d'aplicacions de PC que requereixin accedir a dades de sensors/actuadors per interactuar amb el món físic (per exemple estacions meteorològiques, sistemes de control de pas, domòtica, etc.).

El protocol proposat implementarà les capes de la pila OSI des dels nivells d'aplicació fins al de transport, per tant podrà treballar sobre diferents protocols de capa física com per exemple serie, wifi, usb,...

El desenvolupament inicial serà concretament per aplicacions que requereixin accedir les dades facilitades per un o més sensors o actuadors connectats a la placa LPC, mitjançant el port sèrie. El protocol serà flexible i escalable per permetre enviar diferents ordres, identificant el dispositiu i la comanda, i també serà versàtil per tal de què es pugui adaptar fàcilment a futures versions amb més funcionalitats. També es pretén que l'API sigui senzilla de fer servir, per facilitar el desenvolupament d'aquest tipus d'aplicacions.

Finalment, per tal de poder provar el sistema, es desenvoluparà una aplicació per a PC, que farà ús de l'API per comunicar-se, via port sèrie amb una placa LPC, per tal de reportar les dades obtingudes per un sensor i un actuador.

1.1. Justificació

En els darrers anys els sistemes encastats han experimentat grans avenços, i el seu ús s'estén cada vegada més, i en més àmbits. Aquest TFC esdevindrà una eina molt útil per facilitar i simplificar el desenvolupament d'aplicacions de PC que requereixin fer ús de sensors i actuadors, sense requerir grans coneixements tècnics, ja que farà de capa d'abstracció del hardware.
Gracies a aquesta eina tan versàtil, el client podrà concentrar-se més en el desenvolupament final del seu producte, i estalviar-se els costos i temps del desenvolupament del programari de comunicació, i de control del Hardware. Només requerira la instal·lació del meu servei, i la correcta connexió dels dispositius, i finalment implementar les crides al meu servei que li farà de pont de dades entre la seva aplicació final i els sensors i actuadors connectats.

Metafòricament el servei fa de pont d'abstracció entre una aplicació client i els sensors i actuadors, aconseguint que la aplicació client pugui fer servir els sensor i actuadors sense tindre que endinsar-se a les característiques i programació del hardware.

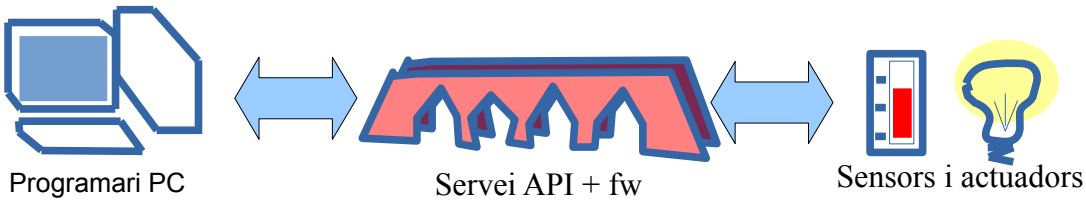


Figura 1.1 Il·lustració metafòrica del servei.

1.2 Descripció del projecte

El projecte constarà de 4 parts diferenciades:

1. Protocol de comunicació: Definició d'un nou protocol de comunicacions entre un programari de PC i el firmware d'una una placa LPC.
Un cop definit es desenvoluparà una modul al PC per implementar el protocol. Aquest modul complira 2 funcions: generar paquets del protocol, i interpretar paquets del protocol. Per que el protocol es faci efectiu, també es desenvoluparà un modul en el firmware amb les mateixes funcions.

2. Desenvolupament de l'API: L'API és qui ofereix pròpiament el servei, i es dividirà en 2 capes: la capa lògica del protocol, i la capa física d'enviament (per port sèrie, per wifi, etc). Inicialment es farà servir el protocol a través del port sèrie del PC. La seva missió serà enviar les comandes per controlar les diferents interfícies del LPC (I2C, ADC, SPI, UART, etc.), fent ús del protocol. El tipus de funció que contindrà, bàsicament seran de tipus inicialització i obtenció de dades.

L'API es desenvoluparà en Python perquè funcioni tant en Windows com en Linux, i es farà seguint un disseny modular per permetre posteriors ampliacions de comandes, i que es pugui reutilitzar en el futur per altres projectes.

3. Aplicació per a PC: Desenvolupament d'una aplicació que farà servir les funcions de l'API, per processar i reportar la informació rebuda. L'aplicació permetrà la recepció instantània de dades, i també podrà ser programada per rebre dades de la placa LPC periòdicament. Aquesta aplicació servirà per poder testejar que les funcions de l'API funcionin correctament i com a exemple pràctic d'ús de l'API. Al igual que l'API haurà de funcionar tant en Windows com Linux.

4. **FirmWare del LPC:** Aplicació que funcionarà sobre la placa encastada. Rebrà les comandes de l'API, les interpretarà fent servir el nou protocol, i les processarà.

1.2.1 Requisits necessaris.

Requisits necessaris de cada part del projecte :

Part del projecte	Requisits
Protocol	- Generació de paquets. - Interpret de paquets - Ampliable i genèric
API	- Lògica de control del protocol i enviament de comandes. - Interpretació de respostes, via protocol. - Disposar de comandes de lectures periòdiques. - Utilitzable per diferents S.O. (Linux i Windows) - Abstracció del hardware. - Funcions senzilles, i facilitat d'ús. - Oferir funcions d'obtenció de dades de sensors, i de control d'actuadors. - Versàtil
Aplicació	- Testejar l'API i servir com a exemple d'ús practic - Visualització de dades
Firmware	- Lògica de control del protocol i execució de les comandes del protocol. - Llegir dades de les diferents interfícies del LPC. - Comunicació amb els diferents sensor/actuador - Retorn de respostes i dades, via protocol.

Figura 1.2 Taula de requisits.

1.2.2 Esquema descriptiu del projecte

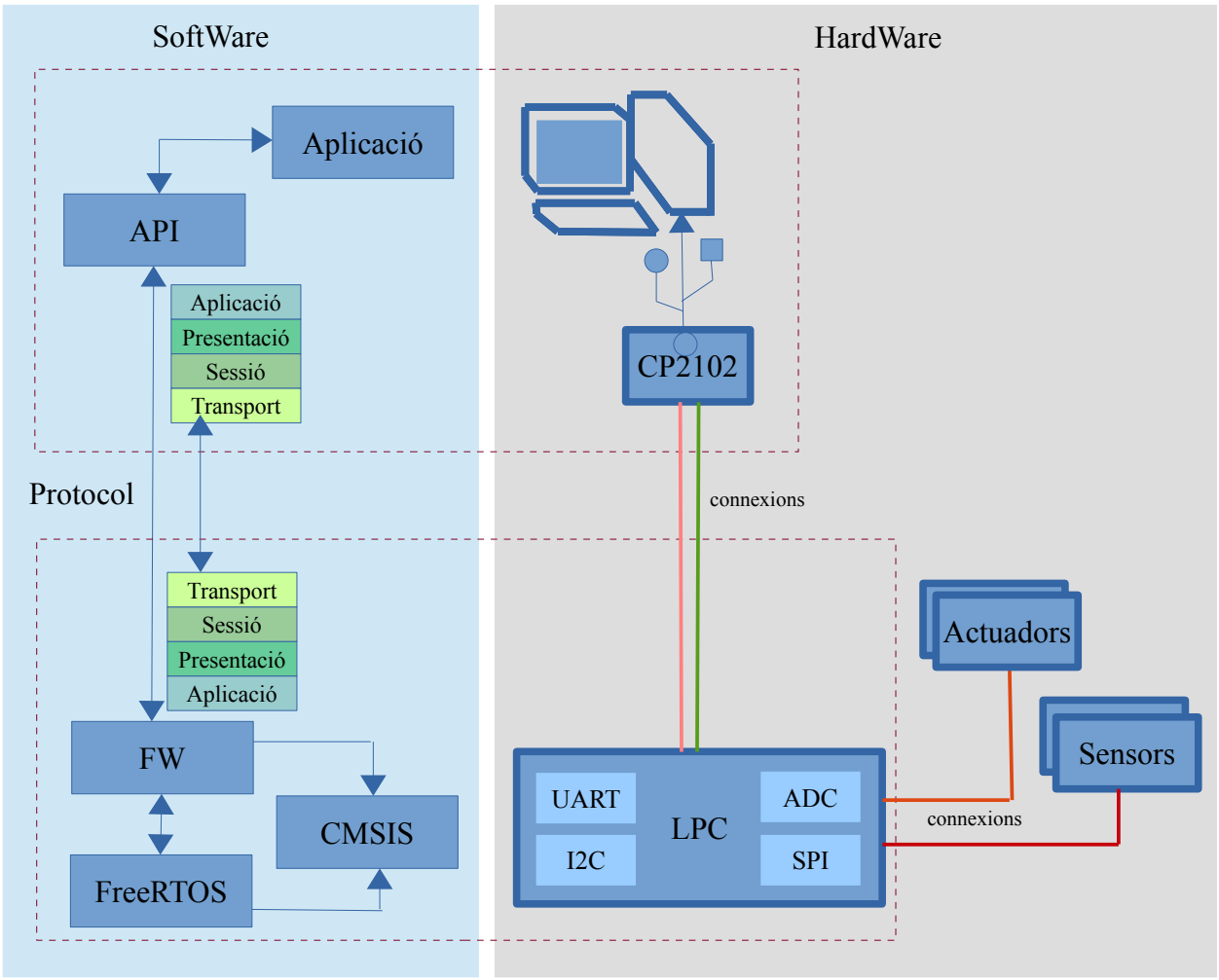


Figura 1.3 Esquema del projecte

1.3. Objectius del TFC.

L'objectiu principal del projecte es fer una capa d'abstracció del hardware tan amplia, que permeti poder connectar qualsevol sensor al LPC, i que simplement amb una comanda o dos comandes (per exemple getADC(0)), una aplicació de PC pugui obtenir el valor del sensor.

Per tant la llista d'objectius a assolir és:

- Abstracció del hardware.

El servei es podrà fer servir sense requerir gaires coneixements tècnics del hardware, ni requerir configuracions del hardware més enllà de les estrictament necessàries de cada dispositiu.

- Lògica de control independent del canal.

Per tal de poder adaptar el protocol a diferents medis de comunicacions (port serial, wifly, usb, etc.), la lògica de control a de ser independent i separada del diferents mòduls de control del medi. Per a cada medi l'API requerirà el seu propi modul de control.

- Protocol genèric, ampliable i independent del medi.

El protocol definit a de ser genèric i ampliable, per tal de què es pugui adaptar a altres sistemes encastats.

- Desenvolupament d'un sistema multi-plataforma.

Com a mínim el sistema a de servir per a Windows i Linux.

- Facilitat d'ús

Cada servei que dona l'API, s'ha de resoldre amb una o com a màxim dos crides a les seves funcions.

1.4. Enfocament i mètode seguit.

La part de programari par a PC s'ha desenvolupat fent ús de la orientació a objectes.
Per desenvolupar la part del firmware s'ha fet servir programació clàssica, amb tasques multi-fil.

L'enfocament empleat per a realitzar el desenvolupament del TFC, s'ha basat en un desenvolupament bàsic (Fase 1 i Fase 2) , i en revisions posteriors (Fase 3) per afegir funcionalitats. Detall de les fases seguides:

Fase 1. Preparació de l'entorn.

Instal·lació dels diferent IDE, llenguatges i llibreries en entorn Linux. Desenvolupament d'algunes funcionalitats bàsiques en el fw.

Fase 2: Desenvolupament bàsic. Simultàniament he:

1. Desenvolupat Aplicació per fer monitoratge, i crida l'API per encendre un led de la placa.
2. Desenvolupament comunicació per port sèrie (en l'API i el fw).
3. Implementació bàsica del protocol.
4. Desenvolupament bàsic per encendre el led de la placa.

Fase 3: Revisions posteriors per afegir funcionalitats.

Paral·lelament he anat redactant les memòries, agafant com a punt de partida la proposta de projecte, i la he aplicat sobre la plantilla de l'assignatura. A partir d'això he anat fent revisions i ampliacions, segons avança el projecte.

1.5. Planificació del projecte.

1.5.1 Llista de tasques previstes inicialment

Tasca	Descripció de la tasca	Hores dedicades
Fase Inicial d'estudi		
1	Estudi de la tecnologia	84
2	Instal·lació i configuració dels diferents IDEs de desenvolupament del projecte.	24
3	Proposta de projecte	16
4	Definició del protocol	16
Fase de desenvolupament		
10	Programar i verificar el funcionament de l'API amb el port sèrie	2
11	Programació bàsica de la aplicació per cridar a l'API i veure resultats en mode Text	5
12	Programació i implementació del protocol en l'API	25
13	Programació i implementació del protocol en el FW	25
14	Desenvolupament de funcions en el FW per treballar ADC, UART, I2C i SPI	16
15	Desenvolupament de funcions de l'API, per treballar amb les funcions del hardware	8
16	Desenvolupament a la aplicació de la presentació bàsica de les dades del sensor.	4
17	Programació en el Firmware de tasques periòdiques	8
Fase de proves		
21	Proves de funcionament amb un sensor (un acceleròmetre).	5
22	Proves de funcionament amb un actuador (led).	5
23	Jocs de proves i depuració de codi.	12
Fase de documentació		
31	Memòries del projecte	En curs
32	Presentació del projecte	En curs

Figura 1.4 Taula de tasques.

Les hores dedicades a cadascuna de les tasques son aproximades.

1.5.2 Cronograma de la planificació inicial del projecte.

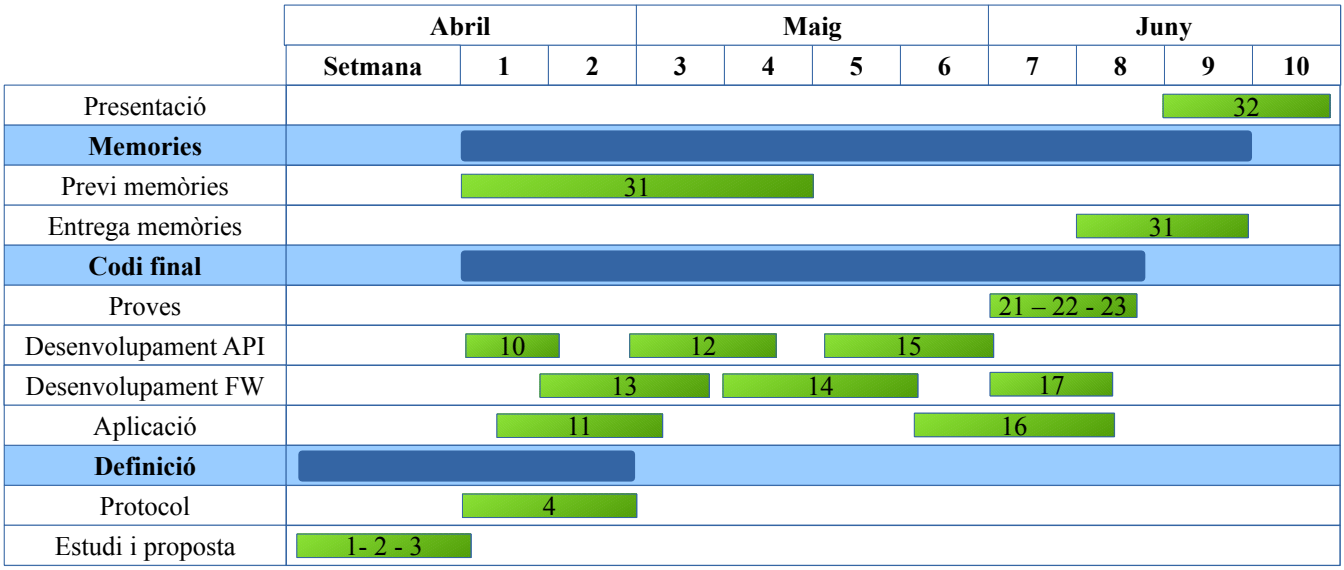


Figura 1.5 Cronograma inicial.

1.5.3 Desviaments de la planificació inicial.

La definició del protocol (tasca 4) de la fase inicial s'ha allargat una setmana , perquè al iniciar el desenvolupament s'ha vist necessari realitzar algunes modificacions sobre la definició inicial.

Al cronograma inicial estava previst iniciar la implementació del protocol al firmware de la placa (tasca 13) i una setmana més tard implementar-ho a l'API (Tasca 12) . Però al la pràctica s'ha invertit l'ordre, ja que ha sigut més convenient, implementar la generació de paquets de sol·licitud a l'API, i verificar que els paquets arribaven convenientment al firmware de la placa.

La implementació del protocol (tasques 12 i 13) son les que més s'han allargat, ja que es on més dificultats tècniques han sorgit, i s'han detectat algunes errades als mòduls que ja havia implementats prèviament durant l'estudi de la tecnologia. També s'ha allargat debut a que per la seva complexitat s'ha desenvolupat per fases, segons avançaven les altres tasques:

1ª Fase: Enviament de paquets al microcontrolador amb una sol·licitud bàsica.
2º Fase: Recepció de paquets, i execució bàsica.
3ª Fase: Enviament de paquets de dades al PC, com a resposta, i interpretació de les dades.
4ª Fase: S'afegeixen paràmetres als paquets, per poder parametritzar les sol·licituds.
5º Fase: Adaptació a paquets amb més d'una sol·licitud.
6º Fase: Depuració de codi

S'ha afegit dues funcions importants a la tasca de treball amb els dispositius (tasca 14), que mancaven a la definició inicial de tasques. Son el desenvolupament de funcions per treballar amb el Led de la placa, i amb el pins de la placa (GPIO). Aquest afegit ha fet que també es dilati la duració de la tasca.

Finalment degut a la dilatació de les tasques indicades s'ha hagut de reduir el temps disponible, per les de presentació de dades (16) i execucions periòdiques (17) , motiu per el qual s'ha fet una implementació molt bàsica.

L'inici de la fase de proves s'ha avançat, ja que s'ha vist necessari iniciar-la per anar provant les parts del projecte que s'anaven desenvolupant, sense esperar al final.

1.5.4 Cronograma final del projecte.

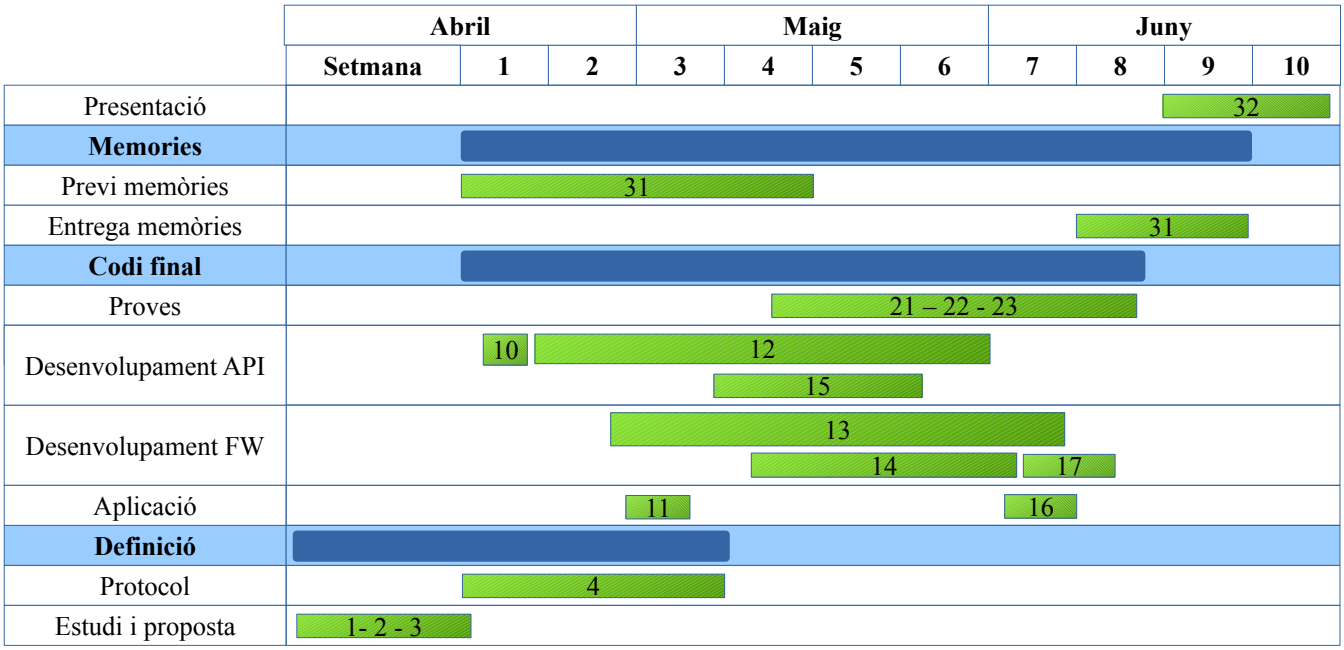


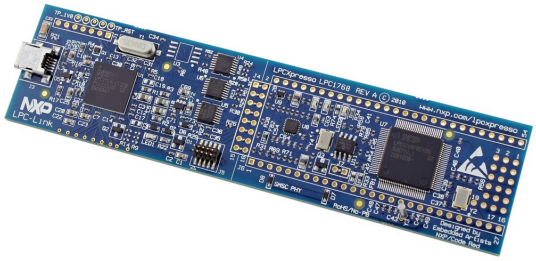
Figura 1.6 Cronograma final.

1.6. Recursos emprats

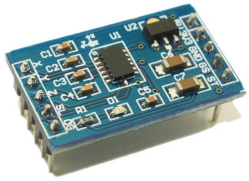
1.6.1 Recursos de software.

	**LPCXpresso v.7.6.2** Entorn de desenvolupament per Eclipse, amb compiladors de C/C++. Amb suport per a microcontroladors NXP amb arquitectura ARM i processadors Cortex-M. (c) Copyright NXP Semiconductors 2006-14. (c) Copyright Eclipse contributors and others 2000, 2014.
	**FreeRTOS** Sistema Operatiu en temps real per a sistemes encastats. Llicencia Open Source.
	**CMSISv2p00** Llibreria de funcions per a microcontroladors Cortex-M, per a sistemes Operatius RTOS,
	**Python 3.4** Interpret del llenguatge Python, llenguatge de programació orientat a objectes i multi-plataforma. Llicencia Open Source.
	**Eclipse Platform 3.8.** Entorn de desenvolupament per diferents llenguatges i plataformes. (c) Copyright Eclipse contributors and others 2000, 2012.
	**PyDev** Entorn de desenvolupament open Source adaptat a Python, per Eclipse. Marca registrada per Appcelerator. Llicencia EPL (Eclipse Public License).
	**PySerial** Modul d'accés al port sèrie per Python. Per Windows i Linux. Licencia Free Software. Copyright (C) 2001-2013 Chris Liechti .
 	**Sistemes Operatius Linux Ubuntu.** **Sistema Operatiu Windows Vista.**

1.6.2 Recursos de Hardware

Placa LPC1769 Placa amb microcontrolador ARM Cortex-M3 i CPU de 120 MHz de freqüència. Inclou una memòria flash de 512 kB, i una memòria de dades de 64 kB. Disposa de Ethernet MAC i de connector USB, de controlador DMA, quatre UARTs, dos canals CAN , dos controladors SSP, dispositius SPI i I2C, vuit canals ADC de 12 Bits, 10 bits de DAC, etc.	

CP2102 UART to USB converter. Transforma senyal de UART a USB i USB a UART. Per comunicar el PC amb el microcontrolador.	

MMA7361 Tre-Axis Analog Accelerometer Sensor Sensor acceleròmetre analògic de tres eixos (X, Y i Z) de baix consum. Amb auto-test, estat d'hibernació, i detector de gravetat 0. Pot treballar amb dos sensibilitats (800 i 206 mV/g).	

Cables de connexió de pins. Cables femella / femella, per connectar els pins dels diferents dispositius.	

Connector mini-USB USB. Cable per carregar el fimware desenvolupat des de l'IDE LPCXpresso a la placa LPC1769.	

1.7. Productes obtinguts

S'han obtingut els següents productes.

1. Una API que serveix per controlar sistemes encastats fent servir una placa LPC.
2. Una aplicació per a PC d'exemple d'ús de l'API.
3. La definició d'un nou protocol per a la comunicació entre un PC i un sistema enquestat.
4. Un sistema encastat format per una placa LPC1769, i un fw amb la implementació del protocol.

1.8. Breu descripció dels altres capítols de la memòria.

A continuació es presentarà un estudi dels següents aspectes:

En el capítol 2 es fa una aproximació a la situació dels sistemes encastats, i les conclusions d'un petit estudi del mercat actual.

En el capítol 3 es fa una descripció funcional del sistema desenvolupat. Primer es fa la descripció global del funcionament i disseny de tot el sistema, i a continuació s'amplia amb una descripció funcional més especifica, per cada una de les principals parts del sistema en el següent ordre, l'API, el firmware, i finalment la aplicació de PC.

En el capítol 4 es fa la descripció del sistema, però a nivell tècnic, detallant els aspectes més importants de cadascun dels elements del sistema desenvolupat.

En el capítol 6 es fa un estudi de la viabilitat tècnica del projecte.

En el capítol 7 es fa una petita valoració econòmica dels costos d'implementació del projecte.

En el capítol 8 es detallaràn les principals conclusions del projecte.

EN el capítol 9 hi ha un glossari de paraules.

En el capítol 10 es pot consultar la bibliografia, i les referencies a les fons d'informació consultades.

2. Antecedents

2.1. Estat de l'art

2.1.1 Plaques i microcontroladors.

En els darrers temps els sistemes encastats han sofert una gran evolució, i actualment es poden diferents plaques amb microcontroladors a un preu econòmic, amb gran varietat d'accessoris, sensors i actuadors. Per la seva diversitat no entrarem a valorar els accessoris, sensors, etc, però si fem a continuació una petita introducció a algunes de les plaques amb microcontrolador més populars del mercat:

Arduino.
Plataforma de Hardware lliure basat en una placa amb microcontrolador, i el seu propi entorn de desenvolupament. La placa es pot comprar feta, o comprar el components, i muntar-la un mateix. Es programa en llenguatge Arduino (basat en Wiring), i des del 2012 treballa també sobre microcontroladors CortexM3 . Molt estes en l'àmbit acadèmic i aficionat. Com a punt fort, hi ha disponible a Internet gran quantitat d'informació i d'exemples sobre aquesta plataforma. Amb la gran popularitat d'Arduino, han sortit diferents plaques al mercat, que imiten el seu model o son directament compatibles amb ell com per exemple Tessel, Maple, NetDuino o pcDuino.

Raspberry Pi.
Originàriament desenvolupat al Regne Unit com a eina de baix cost per a l'aprenentatge de la computació i de la programació. Disposa de tarja gràfica, entrada de targes SD, i dispositius USB.
S'hi poden instal·lar diversos sistemes operatius basat en Linux, i a les ultimes versions també Windows 10.
Hi han disponibles al mercat múltiples accessoris per aquest models de microcontroladors.
I disposa en la xarxa d'una gran comunitats de seguidors i suport.

Placa Gizmo d'AMD.
Com a tret diferencial amb les altres plaques esmentades, aquesta treballa amb processador de tecnologia x86 en comptes d'ARM. Te un bon rendiment, però també te un gran inconvenient, el seu cost es més elevat.

Família LPC de NXP
Família de plaques amb microcontrolador de tecnologia ARM, i processadors CORTEX, a un preu accessible del fabricant NXP (derivat de Philips). Aquest fabricant proporciona de forma gratuïta el seu propi IDE (el LPCXpresso) de programació en C. El seu ús també esta molt estes, encara que no es tan popular en la xarxa com Arduino o Raspberry Pi. La placa que s'ha fet servir en aquest tfc pertany a aquest

fabricant, i família de plaques.

2.1.2 Sistemes Operatius en temps reals per a dispositius encastats.

Descartem els sistemes Operatius encastats, sense execució en temps real, per a sistemes encastats més simples, i ens centrarem en els sistemes operatius en temps real, com el que he fet servir en aquest TFC. Un sistema Operatiu en Temps real es aquell que te un control de temps, normalment amb un planificador de tasques i control d'interrupcions, i per tant permet que les seves funcions i activitats estiguin acotades en el temps. Alguns dels principals Sistemes Operatius en temps real existents són :

RTLinux
Es un sistema en temps real estricte. Es una ampliació de Linux per a oferir el temps real, per oferir creació de tasques en temps real, i control d'interrupcions, dissenyat inicialment per arquitectures x86.

VxWorks
Sistema operatiu en temps real per a sistemes encastats basat en Unix i desenvolupat per Wind River Systems. S'ha fet servir en infinitat de dispositius, i com a curiositat, s'ha fet servir a vehicles espacials com el Sojourner, una mena d'explorador enviat al planeta Mart. Actualment es pot fer servir en quasi tot tipus de processadors, i el seu desenvolupament es fa des d'un Host Unix o Windows.

FreerRTOS
Sistema operatiu desenvolupat majoritàriament en llenguatge C i assemblador. Àmpliament estes al món acadèmic. Permet el control de tasques, prioritats, semàfors i cues. Es pot integrar amb diversos IDE com eclipse o MPLAB per a la programació i depuració. Es el sistema operatiu fet servir per a aquest projecte.

2.2. Estudi de mercat.

2.2.1 Estudi de protocols.

Buscant per la xarxa no he trobat cap protocol estàndard similar al de la proposta d'aquest TFC, orientat a sistemes encastats, i que implementi només els quatre nivells superiors del model OSI. En canvi si que existeixen múltiples protocols estàndard que implement els nivells inferiors, com són:

X.10
Protocol estàndard molt senzill de comunicació entre dispositius, amb trames de pocs bits, bàsicament per encendre i apagar dispositius.

ZigBee
Protocol més elaborat, per a comunicacions sense fil amb dispositius encastats de baix consum.
Molt estès en domòtica, sistemes hospitalaris, etc. Es potser el més complert i elaborat.

2.2.2 Orientació

El sistema desenvolupat va dirigit a estudiants o professionals que requereixin treballa amb sistemes encastats, pensat inicialment en sensors i actuadors però sense descartar altres àmbits. Per tant esta orientat tant al àmbits acadèmic, domestic i empresarial.
Alguns del principals sector per als quals es preveu interessant són:
La domòtica, estacions meteorològiques, seguretat, sector del entreteniment, sector agrari i control de maquinària industrial.

3. Descripció funcional

La funció del sistema desenvolupat , es un servei (en forma d' API) per a controlar sensors i actuadors des d'un PC, senzill i fàcil de fer servir. El sistema requereix que el PC estigui connectat a una placa LPC, ja sigui físicament o amb wifi, des de la què es controlaràn els diferent sensors o actuadors connectats també a la placa LPC. S'ha implementat un protocol per a la comunicació entre el PC i la placa LPC.
El disseny del sistema esta format per tres blocs diferenciats: un API per a PC, el firmware encapsulat a la placa LPC, i finalment una aplicació PC per fer proves.

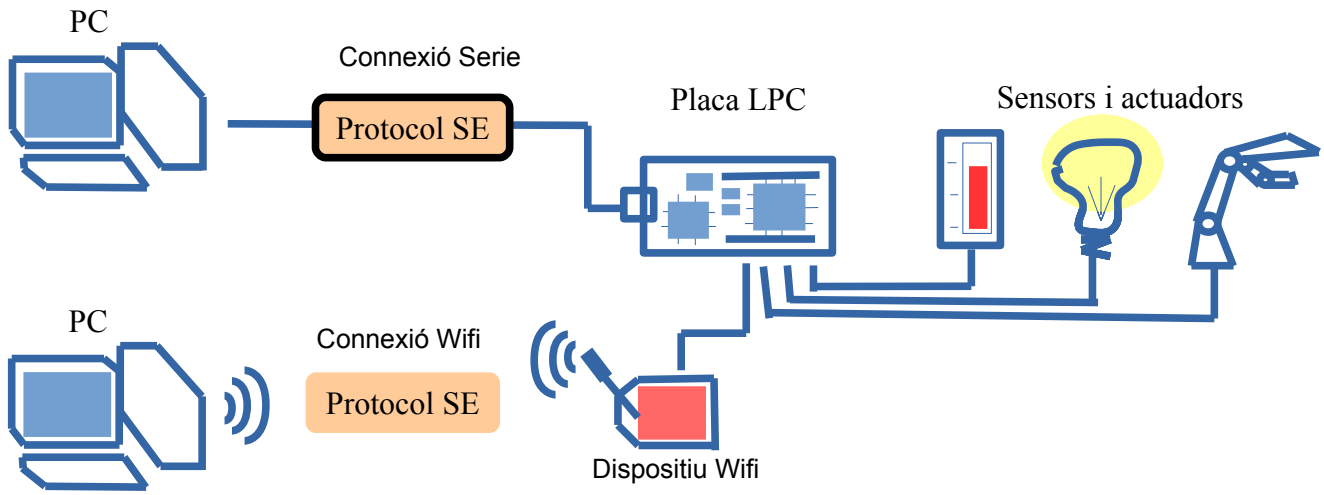


Figura 3.1 Il·lustració del sistema.

3.1. Servei per controlar sistemes encastats

3.1.1 API

La seva funció es proporcionar un servei, per a controlar els diferents sensors i actuadors des d'un PC.
Es un servei genèric, i la utilitat practica final dependrà de l'aplicació client i del ús que se li doni.
Constarà d'una col·lecció de funcions, senzilles i amb pocs paràmetres, per garantir la facilitat d'ús.
L'API esta formada per 2 nivells conceptuals diferenciats:

1. El nivell lògic. Aquest nivell estarà format per tres parts:
   a) La col·lecció de funcions o serveis que ofereix l'API. Es la porta d'entrada de l'API per les aplicacions de PC clients.
   b) La implementació del protocol. Es l'encarregat de què la transmissió de dades entre un PC i una placa LPC segueixi un format entenedor. Estarà format per un empaquetador de dades amb el format del protocol, i un interpret per analitzar les dades rebudes.
   C) La lògica de control. Fa d'intermediari entre les funcions d'entrada, i la implementació de protocol.

2. El nivell físic. En aquest nivell es controlaràn els canals físics de connexió. Inicialment es controlarà el port sèrie, però la previsió es permetre afegir en el futur la connexió wifi.

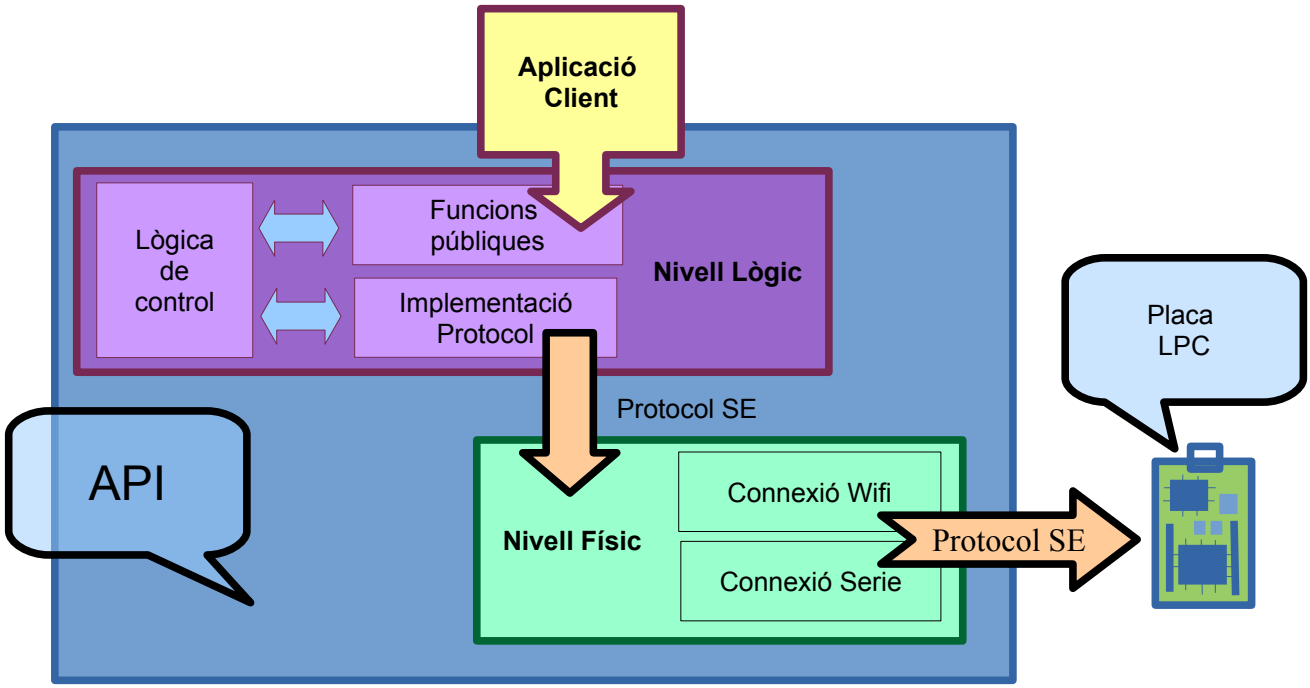


Figura 3.2 Diagrama de blocs de l'API.

Principalment el servei oferira els següents tipus de funcions:

a) Modificacions d'estat: Aquestes funcions serviran per modificar l'estat d'un dispositiu, per exemple encendre o pagar un dispositiu, o per exemple per modificar el voltatge d'un pin entre High o Low.

b) Lectura de dades: Funcions que serviran per recollir les dades dels sensor, ja sigui de forma puntual, o de forma periòdica.

c) Enviament de dades: Funcions per enviar instruccions o dades als diferents dispositius connectats a la placa LPC.

d) Funcions de connexió i configuració de dispositius.

3.1.2 Firmware o sistema encastat (Disseny de l'aplicació fet a la mota)

Es el software que s'instal·larà a la placa del microcontrolador., i farà de pont entre les sol·licituds del PC i els diferents sensors i actuadors connectats a les entrades i sortides de la placa del microcontrolador. Disposarà de diferents mòduls per controlar els diferents ports de comunicació com són els ports UART, el GPIO (pins de la placa), el ADC (per entrades analògiques), etc.
La comunicació amb el PC la farà via protocol.

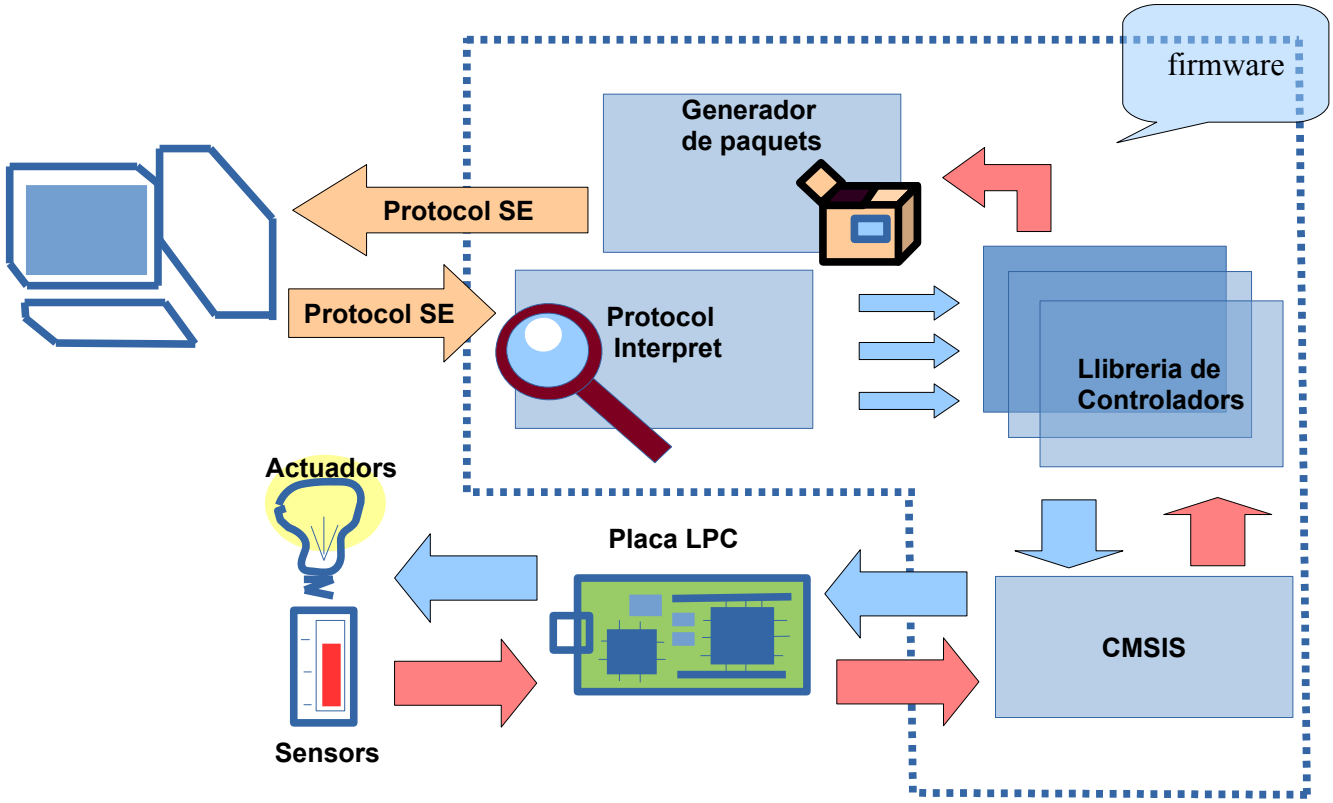


Figura 3.3 Diagrama de blocs del firmware

3.1.3 Interacció dels diferents objectes en el sistema

Las interaccions comencen quan la aplicació de PC desenvolupada d'exemple, o qualsevol aplicació client, fa ús del servei. Les interaccions segueixen els següents passos:

1. La aplicació client fa crides a les diferents funcions o serveis de l'API.
2. L'API tradueix al format del protocol, la sol·licitud feta per la aplicació client, i retransmet els paquets de la sol·licitud per el dispositiu de sortida, ja sigui el port sèrie, o per wifi.
3. El firmware rep els paquets del protocol, per un dels ports de la UART, que pot estar connectada al port sèrie o a un dispositiu wifi.
4. El firmware realitza les operacions d'entrada i sortida per els ports o canals de la placa LPC, requerits per la sol·licitud. Com que estan connectats als sensors i actuadors, realment esta intercanviant dades amb els dispositius.
5. El firmware transforma les dades en paquets del protocol, i envia els paquets de resposta, per la UART.
6. L'API rep els paquets per el dispositiu de comunicació, els desempaqueta i els retorna a la aplicació client.

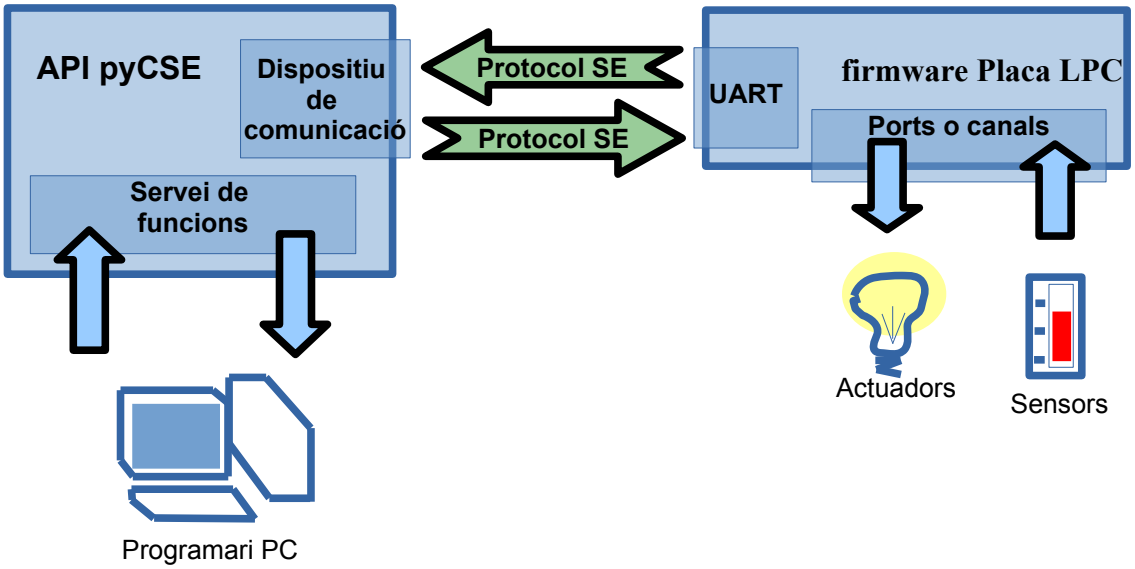
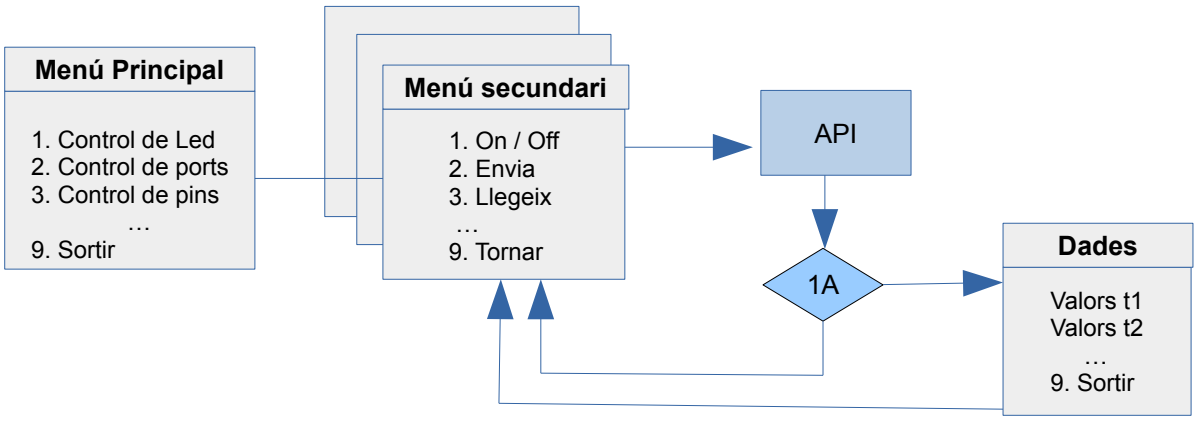


Figura 3.4 Il·lustració d'interacció.

3.2 Aplicació per a PC

La aplicació per a PC inicialment serà una aplicació bàsica, amb monitorització, per fer proves del sistema desenvolupat, i com exemple d'ús. A mida que avanci el projecte s'afegiran funcionalitats i un entorn gràfic amigable. La idea del projecte es que, donat l'exemple cada client final desenvolupi la seva pròpia aplicació, fent servir l'API, però adaptat a les seves necessitats, ja que el projecte es un sistema obert amb múltiples possibilitats. Per exemple es podria ser una aplicació controlada en remot, o amb processos automàtics planificats . La aplicació es farà en Python.

Constarà d'un menú principal, que mostrarà cada tipus de port o dispositiu de la placa, cadascun dels quals enllaçarà amb un segon menú, amb les possibles accions a realitzar sobre el element seleccionat.



1A – Si la API retorna dades mostrara la pantalla de dades. En cas contrari tornara al menú secundari d'opcions

Figura 3.5 Diagrama d'interacció

3.2.1 Diagrama de blocs.

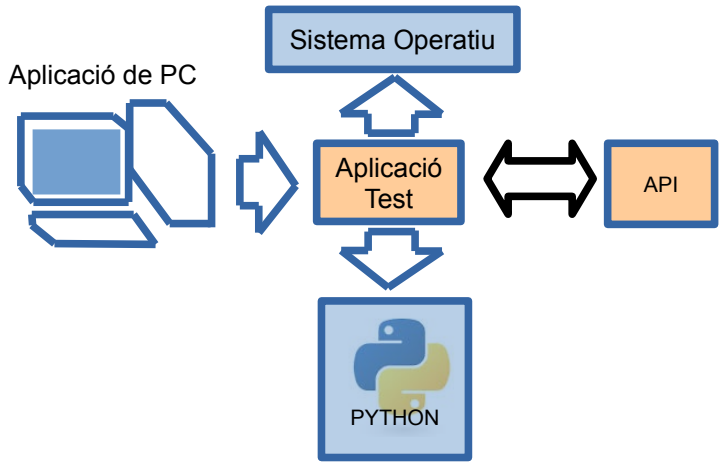


Figura 3.6 Diagrama de blocs de la aplicació PC.

4. Descripció detallada

A continuació es detalla la descripció tècnica de cada component del sistema començant per la part de connexions del maquinari i després la programació del firmware. De l'API, el protocol de comunicació, i l'aplicació d'exemple.

4.1 Connexions de la placa LPC.

4.1.1 Connexió de la placa LPC al PC.

La UART[1] es el port de comunicacions en sèrie, i La placa LPC1769 disposa de 4 ports UART. Per realitzar la comunicació en sèrie entre la placa LPC i el PC, es connecta els 2 pins UART de la placa amb el dispositiu CP2102. El dispositiu CP2102 es un adaptador USB d'entrada de dades en sèrie, i es connecta a l dispositiu USB del PC. Per al projecte es fa servir el port 3 de la UART, igual que es pot fer servir el ports 0 o 1.

Correspondència de les connexions dels port UART amb el dispositiu CP2102:

CP2102		GND	TX	RX
LPC1769	UART3	GND Pin J.1	RX3 Pin J.10 Port P0.1	TX3 Pin J.9 Port P0.0
	UART1		RX1 Pin J.14 Port P0.16	TX1 Pin J.13 Port P0.15
	UART0		RX0 Pin J.22 Port P0.3	TX0 Pin J.21 Port P0.2

Figura 4.1 Taula de connexions UART.

4.1.2 Connexió de la placa LPC al sensor.

Per a les proves amb un sensor d'aquest projecte es fa servir un acceleròmetre de 3 eixos. Aquest sensor envia un senyal analògic de cada eix, cadascun dels quals es connectarà a un canal del ADC[2] de la placa. La placa LPC1769 disposa de 8 canals ADC, que es el dispositiu encarregat de transformar la senyal analògica en un senyal digital. Per a aquest projecte s'han fet servir els canals 0, 1, 2, però també es poden fer servir els altres. El sensor disposa també de quatre pins funcionals (ST, GS, OG i SL) que es poden connectar als port lliures de la placa LPC.

1 Universal Asynchronous Receiver-Transmitter
2 Analog to Digital Converter

Correspondència de les connexions del sensor i la placa.

acceleròmetre	X	Y	Z	3V3	GND	ST, GS, OG, SL
LPC1769	AD0.0 Pin J.15 Port P0.23	AD0.1 Pin J.16 Port P0.24	AD0.2 Pin J.17 Port P0.25	VOUT Pin J.28	GND Pin J.54	Als diferents ports lliures de la placa

Figura 4.2 Taula de connexions sensor

4.1.3 Esquema de connexió del PC amb la placa LPC i l'accelerómetre.

Esquema d'exemple de connexió al PC per la UART 3, i al sensor per els port 0, 1 i 2 del ADC.

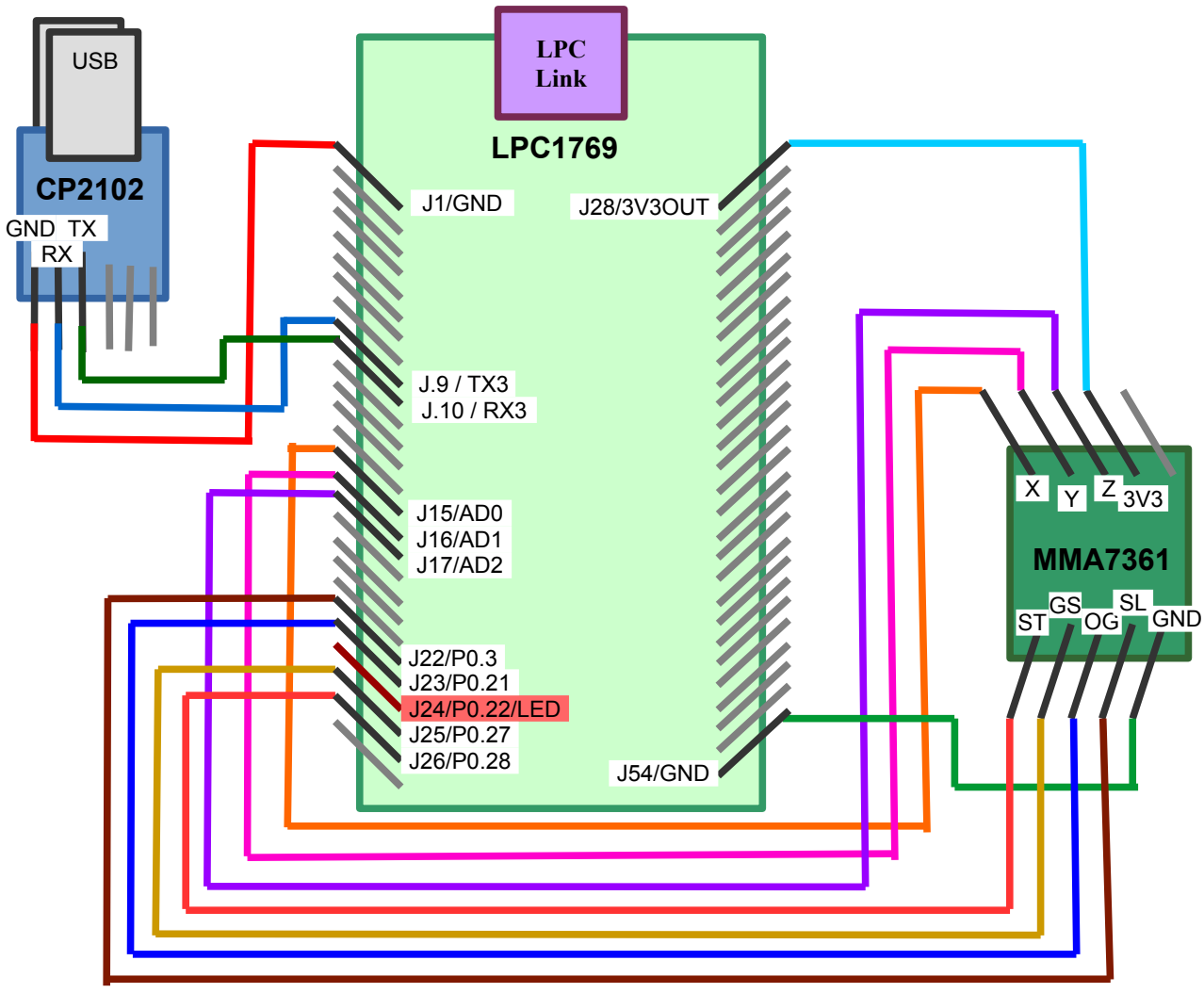


Figura 4.3 Esquema de connexions del sistema.

4.1.4 Esquemes de connexió per UART0 i UART1.

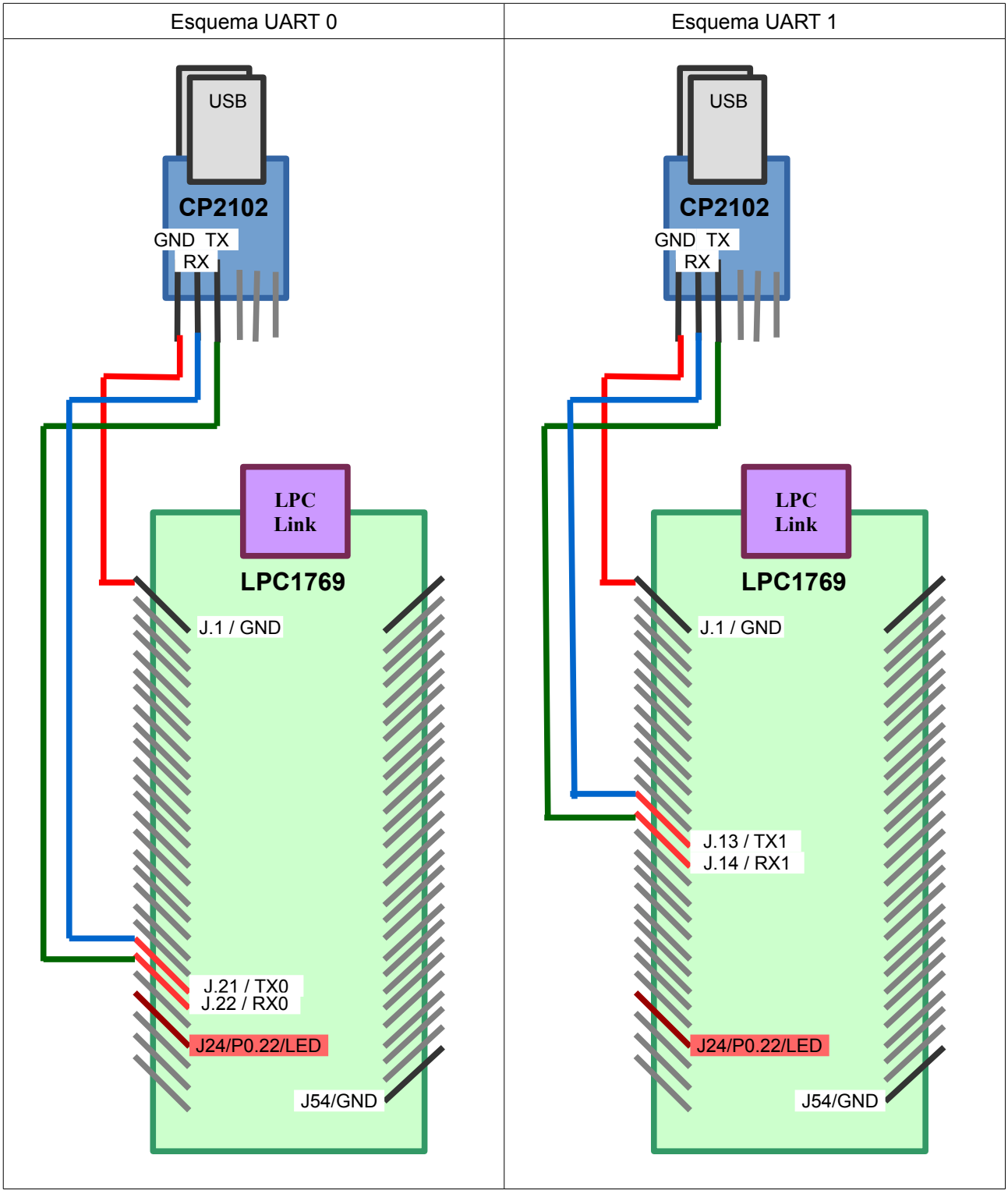


Figura 4.4 Esquema de connexions UART.

4.2 FW.

El firmware de la placa LPC s'ha desenvolupat en llenguatge C amb l'eina LPCXpresso, ja que es el entorn de desenvolupament que proporciona la placa LPC1769 amb la que s'implementa aquest projecte.

4.2.1 Instal·lació del firmware.

Per fer servir aquest projecte, es requisit imprescindible, a més de disposar dels elements del hardware, instal·lar o tenir prèviament instal·lat a la placa del microcontrolador el firmware de control i suport del protocol implementat. Per fer la instal·lació:

a) Ha de connectar-se amb un cable mini-usb la sortida de la placa LPC-Link am la entrada USB del PC.

b) Ha d'instal·lar-se o tenir instal·lat l'IDE LPCXpresso.

c) Ha d'importar-se (opció File → Import) el programari C d'aquest projecte.

d) Fer un Build del projecte, amb el botó . 

e) Polsar sobre el botó Program Flash. 

4.2.2 Llibreries i components del firmware.

Per fer el firmware s'han aprofitat els següents elements de programari:

1. Es fa servir la llibreria CMSIS, que es la llibreria que proporciona el fabricant del processador Cortex-M3 com a abstracció de la programació del processador abaix nivell.

2. Es fa servir el Sistema Operatiu FreeRTOS, que es un sistema operatiu en temps real per a Sistemes encastats. Com a utilitats destacables, aquest sistema operatiu proporciona les eines per treballar amb múltiples tasques d'execució en paral·lel i en temps real, i controlar l'accés recurrent dels recursos amb semàfors.

3. Es fa servir la llibreria LibUOC, desenvolupada durant el curs, i que conté els mòduls necessaris per enviar o rebre dades per la UART, controlar el Led de placa, modificar l'estat dels pins, accedir al ADC, etc.

Finalment s'ha desenvolupat la lògica del firmware per contribuir al servei de control del sistema encastat.

4.2.3 Diagrama de flux del firmware.

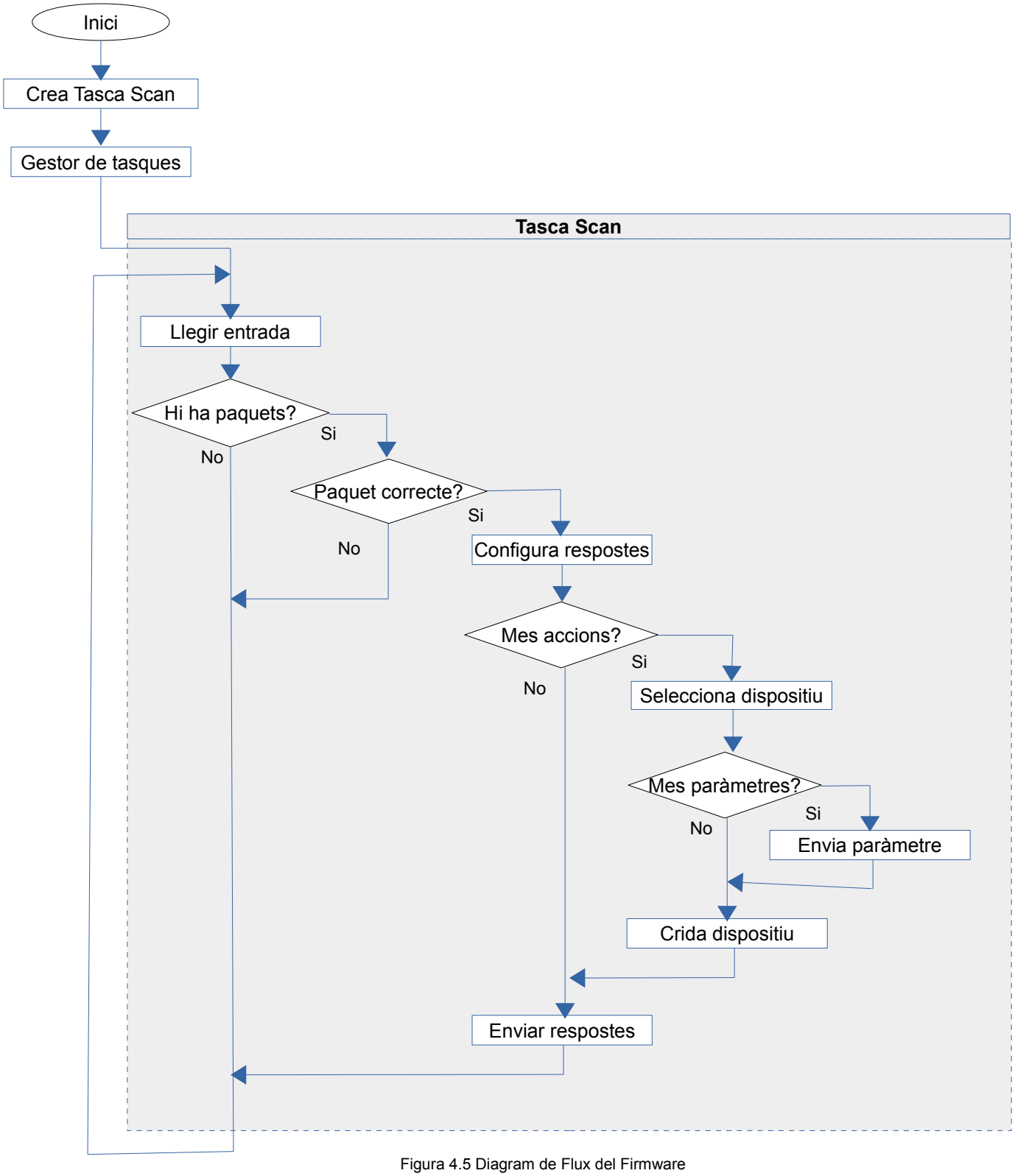


Figura 4.5 Diagram de Flux del Firmware

4.3 API.

S'ha decidit desenvolupar l'API en python per els següents motius:

1. Permet la orientació a objectes.
2. Es multi-plataforma (funciona amb Windows i Linux) .
3. Permet desenvolupaments més ràpids que el Java, i el temps es un factor important en el desenvolupament de projectes.

S'ha cregut convenient dividir l'API en 2 parts ben diferenciades, la part física, encarregada del medi de comunicació, i la part lògica, que ofereix el servei. El motiu es que es pugui oferir el servei de control dels sistemes encastats, sigui quina sigui al via de comunicació, e inclús si es canvia.

El nom de la API desenvolupada es pyCSE.

4.3.1 Diagrama de classes.

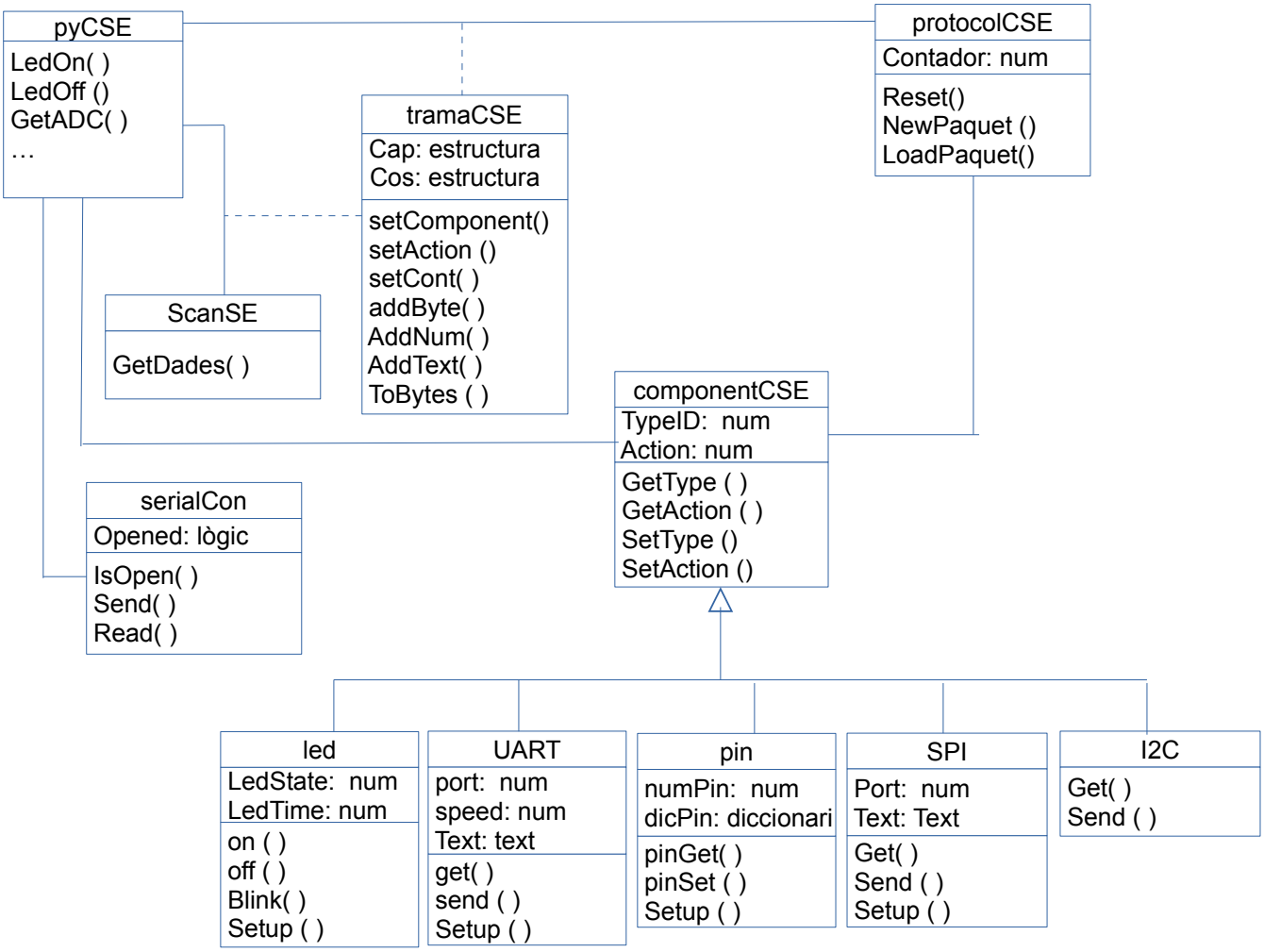


Figura 4.6 Diagram de Classes de l'API

4.3.1 Nivell físic.

Inicialment s'ha desenvolupat la comunicació per port sèrie, però amb la idea d'incorporar en el futur la opció d'oferir comunicació wifi, i també es deixa la porta oberta per permetre afegir en el futur noves vies de comunicació. Abans de fer servir el nivell lògic es requisit imprescindible estblir el tipus de connexió física.
Ja que l'API s'ha desenvolupat en llenguatge Python, per oferir la comunicació per el port sèrie, s'ha decidit fer servir la llibreria pySerial, desenvolupada amb aquesta finalitat.

4.3.2 Nivell lògic.

Tota la col·lecció de funcions lògiques que ofereix l'API s'han encapsulat dins de la classe pyCSE que serveix com a interfície del servei.
El nivell lògic treballarà sobre una classe connexió, per enviar i rebre dades, independentment del tipus de connexió que s'estableix al nivell físic, abans de fer ús del nivell lògic.
Cada funció, invocara a la classe dispositiu que correspongui, i la enviara al generador de paquets, el qual li retornarà una seqüencia de bits, per retransmetre a la classe connexió. Si la funció espera rebre algun tipus de resposta, sol·licitarà a la classe connexió, la seqüencia de dades rebudes, i agafara els paquets que trobi en ella. Per discernir si les respostes corresponen a la sol·licitud, es fa servir el número de seqüencia de la capçalera, ja que els paquets de resposta mantenen el número de seqüencia de la sol·licitud.
Per mitja de la classe scan, analitzara els paquets, per cercar les dades esperades, segons el dispositiu.

4.4 Especificacions i disseny del Protocol de Comunicació

Es defineix un nou protocol de comunicacions entre l'API i el fiwmare, que estarà format per paquets amb una capçalera de 12 bytes i un cos de mida variable. A la capçalera s'identificarà el paquet amb un número seqüencial. L'Api enviarà paquets de peticions, i el firmware enviarà paquets de resposta.
Per poder implementar el protocol es desenvoluparà un generador de paquets, i un interpret de paquets. El protocol serà independent del canal de transmissió.

4.4.1 Identificador / Nom del Protocol.

He decidit donar-li com a nom descriptiu **CSE (Control de Sistemes Encastats).**

4.4.2 Comparativa de la implementació del protocol SE amb el Model OSI

Nivells del Model OSI	Nivells del Protocol CSE
Nivell de Transport Es l'encarregat de realitzar el transport de dades.	**Nivell transport.** Es l'encarregat d'agafar les seqüències de dades de mida variable, i enviar-les al canal de sortida, independentment de quin sigui, com per exemple pot ser el controlador del port sèrie, o el controlador wifi.
Nivell de Sessió Es l'encarregat de mantenir i controlar l'enllaç establert entre dos dispositius que transmeten dades.	**Nivell se seqüencia.** Aquí simplement es l'encarregat de sincronitzar les sol·licituds amb les respostes, amb el número se seqüencia de les capçaleres. Però es deixa la porta oberta per a ampliar els controls de sessió.
Nivell de presentació. Es l'encarregat de representar la informació de manera que els dos dispositius la puguin reconèixer.	**Nivell de paquet.** Es el nivell encarregat d'interpretar, i generar els paquets, segons les especificacions del protocol.
Nivell d'Aplicació Ofereix a les aplicacions els serveis per accedir a la resta de nivells, i defineix els protocols que fan servir les aplicacions per intercanviar dades.	**Nivell se Servei.** Proporciona les diferents funcionalitat que ofereix l'API.
Nivell de Xarxa. Fa el adreçament lògic, i determina la ruta de transmissió.	**Nivell sense implementació.** Aquest nivell vindran implementats per els controladors dels dispositius de comunicació, com el port sèrie o el wifi.
Nivell d'Enllaç Fa el adreçament físic.	
Nivell Físic S'encarrega del senyal de transmissió.	

Figura 4.7 Comparativa amb el model OSI

4.4.3 Disseny del protocol de comunicació.

El protocol treballarà amb paquets, cadascun dels quals constarà d'una capçalera de 10 bytes, i d'un cos de dades de mida variable, excepte els paquets de resposta que podran no tenir cos. El disseny del paquet serà el mateix en els dos sentis de la comunicació.

Disseny de la capçalera del paquet:

Capçalera del paquet (mida total 10 bytes)			
Camp	**Mida**	**Valor**	**Descripció**
Protocol	2 Bytes	'SE'	Identificador del protocol. El identificador serà SE
Versió	1 Byte	1	Versió del protocol
Contador	2 Bytes	n	Número seqüencial del paquet
Mida	1 Bytes	m	Mida en bytes del paquet (incloent capçalera i cos).
Resp.	1 Byte	0	Indicador de resposta. S'activarà a 1 per respostes OK, i a 2 per respostes KO. La resta de valor es podran fer servir en futures ampliacions.
Reserva	3 Bytes	0	Reserva per a futures ampliacions del protocol

Figura 4.8 Disseny de capçalera dels paquets

Pet tal de donar flexibilitat i versatilitat en les comunicacions el disseny del cos del paquet tindrà les següents característiques:

1. Mida variable amb un limit màxim 245 bytes (255 - 10 capçalera). En cas de voler enviar gran volum de dades, es tindria que dividir en paquets.
2. Permet concatenar varies comandes en un sol paquet (formades per **dispositiu + acció + dades**) fins assolir la mida en bytes del paquet, indicada a la capçalera. La finalitat es estalviar temps de comunicacions, i enviaments de capçaleres.
3. Permet enviar un número de variable de dades. Aixo permet la enviament de comandes complexes als dispositius, i per altra banda, permet concatenar seqüencies de valors, especialment útil per a els paquets de resposta de dades dels sensors.

Disseny del cos del paquet:

Cos del paquet (mida variable, mínim 5 bytes – màxim 245 bytes)		
Camp	**Mida**	**Descripció**
Dispositiu	2 Bytes	Interfície o dispositiu sobre el que es vol actuar (UART, ADC, Led, etc)
Acció	2 Bytes	Operació o acció a realitzar (Start, Reset, Get, Set, etc.)
Num.Dades	1 Bytes	Núm. de paràmetres o dades associats a la acció sol·licitada
Mida.1	1 Byte	Mida en bytes del operador o dada 1
Dada 1	X Bytes	Dada o operador 1
Mida X	1 Byte	Lóngitud del operador o dada X
Dada X	X Bytes	Dada o operador X

Figura 4.9 Disseny de cos dels paquets

Alguns valors de referència

La següent taula, només conte els primers valors definits, ja que vaig afegint, segons vaig necessitant.

Camp	Mida	Valors	Descripció
Resposta	1 Byte	x00	Sol·licitud
		x01	Confirmació de sol·licitud
		x02	Error en la sol·licitud
Interfície	2 bytes	x00x01	**ADC.**
		x00x11	**UART.** Com a dada s'enviarà el número de la UART (0, 1 o 3)
		x00x21	**SPI.**
		x00x31	**I2C**
		x01x01	**Led**. Com a dada s'enviarà el identificador del Led.
Operació	2 bytes	x00x01	**On.** Encendre, per exemple per a Leds.
		x00x02	**Off**. Apagar, per exemple per a Leds.
		x00x02	**Blink.** Fer intermitències.
		x01x01	**Get.** Sol·licitar un valor o una dada.
		x01x02	**Set.** Informar un valor o una dada.

Figura 4.10 Valors de referència

Exemple pràctic de comunicació:

En un dispositiu amb diversos leds, es podria interactuar amb varis leds a l'hora, indican com a dada el número de LED.

Exemple de paquet de sol·licitud, per **apagar el Led 1** i **encendre el led 2**:

Capçalera										Cos													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21.	22	23
S	**E**	**1**	**0**	**1**	**24**	**0**	**0**	**0**	**0**	**01**	**01**	**00**	**02**	**1**	**1**	**1**	**01**	**01**	**00**	**01**	**1**	**1**	**2**
Prot.		Ver	Cont		Mid	Res	Reserva			Disp.		Acció		Nro	Mid	Val	Disp.		Op		Nro	Mid	Val

Figura 4.11 Exemple de sol·licitud 1

Exemple de paquet resposta a la sol·licitud, amb confirmació simple:

Capçalera									
0	1	2	3	4	5	6	7	8	9
S	**E**	**1**	**0**	**1**	**10**	**1**	**0**	**0**	**0**
Prot.		Ver	Cont		Mid	Res	Reserva		

Figura 4.12 Exemple de resposta 1

Exemple de paquet de sol·licitud, per **obtenir un valor del ADC, del canal 0** :

Capçalera										**Cos**						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	**E**	**1**	**0**	**2**	**17**	**0**	**0**	**0**	**0**	**00**	**01**	**01**	**01**	**1**	**1**	**0**
Prot.		Ver	Cont		Mid	Res	Reserva			Disp.		Acció		Nro	Mid	Val

Figura 4.13 Exemple de sol·licitud 2

Exemple de resposta a la sol·licitud (on les ? Son qualsevol valor binari):

Capçalera										**Cos**									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S	**E**	**1**	**0**	**2**	**20**	**1**	**0**	**0**	**0**	**00**	**01**	**01**	**01**	**1**	**4**	**?**	**?**	**?**	**?**
Prot.		Ver	Cont		Mid	Res	Reserva			Disp.		Acció		Nro	Mid	Val			

Figura 4.14 Exemple de resposta 2

4.5 Aplicació d'exemple.

Es decideix desenvolupar-la en Python per concordança amb l'API.
Es decideix fer una aplicació senzilla i sense entorn gràfic, per manca de temps, i perquè després de estudiar els diferent entorns gràfics existents per a Pyton, s'ha conclòs que no garanteixen facilitat de instal·lació, i a la vegada siguin compatibles amb totes les plataformes.
Nomes es una aplicació bàsica de exemple i de test per fer proves, i mostrar resultats.

6. Viabilitat tècnica

El projecte és viable, ja que a priori l'únic requeriment físic es disposar d'un PC, una placa LPC1769, un adaptador CP2102 i d'un o més sensors/actuadors.

Les funcions FirmWare es desenvoluparàn en C, amb l'IDE LPCXpresso sota entorn Linux, per evitar incompatibilitats del LPCXpresso amb altres sistemes operatius. La part per desenvolupar l'API es farà en Python, per ser un llenguatge multi-plataforma, ja que funciona amb codi intermedi, i que permet desenvolupaments més ràpids per Java, pese a que Java es més complert. Es farà servir l'IDE Eclipse amb el paquet pydev. La part de desenvolupar la aplicació es aprofitarà el mateix llenguatge i entorn que per desenvolupar l'API (Python amb l'IDE Eclipse + pydev).

El projecte es provarà amb els sistemes operatius Windows Vista i Linux Ubuntu, per obtenir dades d'un acceleròmetre, i actuar sobre els leds del LPC.

La principal dificultats tècnica, inicialment serà aconseguir la comunicació d'un modul multi-plataforma, via port sèrie, amb la placa LPC. Un altre dificultat serà la correcta recepció/emissió del protocol tant des del PC com des del LPC. La resta de possibles entrebancs, seran les dificultats tècniques que puguin sortir de la comunicació amb cada una de les interfícies de la placa LPC.

Punts Forts

El material necessari per implementar aquest sistema es molt econòmic, i no requereix gran inversió.
El programari es molt genèric, modulable i escalable per adaptar-lo a altres sistemes.
El programari es molt obert i modulable per permetre afegir noves funcionalitats.

Punts Dèbils

D'entrada no es pot saber amb quin medi es connectarà la placa LPC amb el PC per comunicar-se (si per UART0, UART1, UART3, si per wifi o per qualsevol altre mitjà). Per resoldre aquest punt hi ha dos opcions:

- Opció 1: Verificar constantment la comunicació a totes les possibles entrades, i assumir conseqüent malbaratament de recursos de la placa.
- Opció 2: Que el client final es configuri les entrades desitjades al firmware, amb la qual cosa es perd un punt en el objectiu de facilitat d'ús del sistema.

Altre punt dèbil es la dificultat d'aconseguir l'estandardització d'un protocol.

7. Valoració econòmica

Pressupost material

Es el pressupost del material fet servit en aquest treball. En cas de què es volgués adaptar per a altres projectes, el pressupost variaria en funció dels tipus de sensor i actuadors que es volguessin fer servir.
Els preus descrits son orientatius i poden variar en funció del proveïdor.

referència del producte	Preu Unitari orientatiu	Unitats	Total
LPCXpresso LPC1769 Board. Fabricant NXP. MCU amb arquitectura ARM Cortex-M3	21 €	1	21 €
CP2102 UART to USB converter	4 €	1	4 €
Cable USB 2.0	3 €	1	3 €
Joc de cables multicolor (x 10)	1 €	1	1 €
Sensor acceleròmetre model MMA7361 Tri-Axis Analog	3 €	1	3 €
Despeses de tramesa del material	4 €	1	4 €
		Preu Total	**36 €**

Figura 7.1 Pressupost material

Pressupost en llicencies de software

referència del producte	Preu Unitari orientatiu	Unitats	Total
Linux Ubuntu 14	gratuït	1	0 €
Eclipse	gratuït	1	0 €
Python, Pydev i PySerial	gratuït	1	0 €
		Preu Total	**0 €**

Figura 7.2 Pressupost de programari

Costos d'Instal·lació i manteniment

Tot i que el projecte neix amb la idea de facilitar que s'ho pugui instal·lar i adaptar el propi client, es detalla a continuació el cost que suposaria encarregar a un tècnic la adequació el projecte a les necessitats del client.

referència del producte	**Preu orientatiu**
Estudi de viabilitat i estimació d'hores	100 €
Programació i adaptació del sistema	15€ la hora
Instal·lació i manteniment del sistema	15€ la hora

Figura 7.3 Costos d'instal·lació i manteniment

Material opcional no inclòs al pressupost

Descripció	Preu orientatiu	Unitats
PC amb entrades USB. En cas de no disposar, s'hauria d'afegir al pressupost. El seu cost seria variable, en funció de les prestacions	Molt variable	1
Windows Vista. Microsoft ja no el te a la venta. Preu orientatiu de Windows 8.1 o posteriors.	120 €	1

Figura 7.4 Costos de material opcional

8. Conclusions

8.1. Conclusions

Els objectius principals del projecte s'han complert, però no es pot afirmar que s'hagin complert totalment, ja que hi han alguns matisos que s'indiquen al apartat 8.1.2 objectius no complerts totalment i que cal tenir en compte.

8.1.1 Objectius complerts

Abstracció del hardware.
A partir de funcions simples el projecte desenvolupat permet controlar un led, l'estat dels diferents pins de la placa, i enviar i rebre dades per les diferents entrades i sortides de la placa.

Lògica de control independent del canal.
El control de comunicació de l'API pel port sèrie s'ha fet independent del control lògic de l'API. Per tant canviant el medi de comunicació, el servei continua oferint el mateixos serveis i funcionament.

Protocol genèric, ampliable i independent del medi.
El protocol es genèric, sense cap dependència del maquinari i amb indicador de versions per a futures ampliacions.

Desenvolupament d'un sistema multi-plataforma.
Tots els recursos de programari empleat funcionen tant als sistemes operatius Windows com a Linux.

Facilitat d'ús
Les funcions de l'Api desenvolupada són fàcils de fer servir.

8.1.1 Objectius no complerts totalment

Lògica de control independent del canal.
Tot i que la lògica de control és independent del canal, això no es pot verificar, ja que per manca de temps només s'ha desenvolupat la comunicació pel port sèrie. Com a verificació d'aquest punt era desitjable haver desenvolupat també la comunicació via wifi.

Desenvolupament d'un sistema multi-plataforma.
Tot i que funciona en més d'una plataforma, no es pot garantir que l'API desenvolupada funcioni sota altres

sistemes operatius com els de la casa Apple o Android.

8.2. Proposta de millores

Desenvolupament de la comunicació wifi

Es una limitació important que només s'hagi implementat la comunicació entre el PC i la placa LPC per el port sèrie, i seria convenient implementar la comunicació entre el PC i la placa LPC per via wifi, que com ja s'ha comentat no s'ha implementat per falta de temps.

Adaptació de l'API per a APPs de mòbil

Actualment hi ha un ús extensiu de la tecnologia mòbil, i seria molt interessant poder adaptar l'API per a poder fer servides des d'APPs de mòbil, perquè es pogués comunicar fent ús de la wifi amb una placa LPC. L'objectiu seria fer servir el mòbil com a comandament a distancia de sistemes encastats, com poden ser una casa domòtica, controlar dispositius d'un cotxe, o controlar una estació meteorològica.

Tot i que amb el sistema actual es factible fer-ho comunicant el mòbil per wifi a un PC que a la vegada es comuniqui amb la placa LPC, per a alguns sistemes encastats com els indicats el PC podria resultar un intermediari innecessari.

Adaptació del firmware a altres models de plaques

Seria interessant adaptar el firmware a altres models de plaques amb microcontroladors del mercat, perquè es puguin comunicar per el protocol implementat, i així facilitar les possibilitats d'ús del sistema desenvolupat.

8.3. Autoavaluació

8.3.1 Aprenentatge.

Gracies a aquest TFC he pogut introduir-me al món dels sistemes encastats, del qual no tenia gaires coneixements, i estic satisfet de tot el que he pogut aprendre, com es:

Rellevància

Aquest projecte m'ha fet conscient de la gran rellevància que tenen avui dia el microcontroladors, i que estan presents en la majoria de dispositius que ens envolten.

Conceptes.

M'ha servit per aclarir molts conceptes importants en aquest àmbit, com per exemple que es un sistema encastat, un FirmWare, un RTOS, o un microcontrolador.

Ampliació de coneixements

He apres que són i com funcionen alguns del components de la placa LPC, com que es una UART, un ADC, un I2C, un SPI etc, i com poden interactuar amb alguns dispositius com un sensor o un adaptador wifi.

Experiència amb els IDEs

He adquirit experiència i he apres a fer servir els IDEs LPCxpresso i PyDev, a crear projectes, compilar, etc.

Familiarització amb els RTOS

M'ha permès familiaritzar-me amb els sistemes operatius en temps real, i en concret amb el FreeRTOS, i les seves possibilitats (creació de tasques, controls de prioritat, semàfors i cues).

Diagrames de blocs

He adquirit experiència en la realització de diagrames de blocs. Mai no havia tingut oportunitat de fer-los servir en el desenvolupament d'un projecte.

8.3.2 Desenvolupament.

En general estic content amb el projecte desenvolupat donat el temps disponible. Tot i que m'han quedat alguns aspectes que hagués volgut millorar com són:

Entorn gràfic

Volia donar a la aplicació d'exemple un entorn gràfic agradable, però després d'estudiar els possible entorns gràfics disponibles per Python (com per exemple Tkinter, wxPython, pyGT, pyQT, etc) he descartat aquesta idea, ja que dels entorns estudiats uns tenien la pega que la instal·lació era complexa, i uns altres que no garantien el funcionament multi-plataforma.

SPI e I2C

Per falta de previsió i coneixement, ja que pensava que amb un sensor tindria prou per provar els diferents dispositius de la placa LPC, no vaig adquirir cap dispositius senzill que hem permetes fer proves amb aquests ports de comunicacions, i per tant no he pogut garantir el correcte funcionament d'aquests ports.

Depuració i proves

Finalment m'hagués agradat disposar de més temps per acabar de depurar el codi, fer algunes millores tècniques, fer més proves i corregir algunes errades detectades.

9. Glossari

A

ARM: De Advanced Risc Machine. Processadors simples amb arquitectura RISC de 32 o 64 bits ideals per a aplicacions de baixa potencia com els microcontroladors.

ADC: Analog to Digital Converter. Dispositiu que transforma una senyal analògica, en una senyal digital.

C

Cortex: Model de Processador d'arquitectura ARM.

Cortex-M3: Model de Processador d'arquitectura ARMv7-M amb control de temps i arquitectura de memòria Harvard, desenvolupat per microcontroladors.

CPU: Unitat central de processament. Maquinari que interpreta les instruccions d'un programari.

E

Eclipse: Entorn de desenvolupament gràfic, originàriament per a Java, però amb suport per a múltiples llenguatges.

F

Firmware: Programari encapsulat dins d'un dispositiu.

FreeRTOS : Es el nom d'un Sistema Operatiu en Temps Real, per a Sistemes Encastats.

FW: Sigles de FirmWare. Programari encapsulat dins d'un dispositiu.

I

I2C:v Bus de comunicació en sèrie unidireccional, del tipus mestre-esclau, amb dos línies de dades, i una línia de rellotge

M

MCU: abreviatura de microcontrolador.

L

LPC: Model de plaques amb arquitectura ARM i microcontrolador CORTEX,del fabricant NXP.

LPCxpreso: Entorn de desenvolupament sota l'IDE eclipse proporcionat per NXP, per desenvolupar aplicacions per a la seva gamma de plaques LPC.

P

Protocol: Sistema de regles que permet a dos o més entitats d'un sistema es comuniquin per transmetre informació.

R

RISC: De Reduced Instruction Set Computer. Es un disseny de CPUs que fa servir conjunts d'instruccions petites i simples. Es fa servir tant en microprocessador com en microcontroladors.

RTS: Sigles de sistema en temps real. Es un sistema amb control del temps.

RTOS: Sigles de Sistemes Operatius en Temps Real. Com el seu nom indica es un sistema operatiu amb control de temps.

S

Sistema Encastat: Programari encapsulat dins d'un maquinari amb unes funcions concretes.

SPI: Serial Peripheral Interface Bus. Bus de comunicació síncrona i en sèrie, tipus full-duplex i comunicació mestre-esclau, per a curtes distancies indicat per a sistemes encastats.

T

TFC: Sigles de Treball de Fi de Carrera.

U

UART: Sigles d'Universal Asynchronous Receiver-Transmitter. És un controlador de ports de comunicació en sèrie.

UC: Abreviatura de microcontrolador.

USB: Universal Serial Bus. Es un bus estàndard per connectar, comunicar i proporcionar alimentació elèctrica entre computadors o dispositius electrònics.

10. Bibliografia

«Sistemes encastats » Editat per la Universitat Oberta de Catalunya
Autors: Jose Maria Gómez Cama, Francisco Hernández Ramírez, Jose López Vicario, Antoni Morell Pérez, Juan Daniel Prades García, Ignasi Villajosana Guillén i Xavier Vilajosana Guillén

«Manual de referència C» Quarta edició (2001 – ed. McGraw-Hill)
Autor Herbert Schildt, traductor Luis Hernández Yáñez

«Python para todos» Edició en PDF amb llicencia Creative Commons Reconocimiento
Autor Raúl González Duque

«Wiki de l'assignatura» Pàgina amb guies i exemples per a iniciar-se:
http://cv.uoc.edu/webapps/xwiki/wiki/matembeddedsystemslabhome/view/Material/IniciCortexM3?

Informació general en:
www.ca.wikipedia.org/
https://www.**wikipedia**.org/

Informació sobre plaques i microcontroladors en:
http://www.nxp.com/
http://comohacer.eu/
http://hacedores.com/que-tarjeta-de-desarrollo-elegir-parte-2/
http://blog.goshield.es/2012/01/comenzando-por-lo-basico-2-disenos.html
http://www.forosdeelectronica.com/f24/empezar-microcontroladores-arm-55856/

Informació sobre RTOS en:
http://www.ecured.cu/index.php/FreeRTOS
http://www.taringa.net/posts/info/16112802/Los-Sistemas-Operativos-De-Los-Cajeros-Automoviles.html

Informació sobre protocols de comunicació en:
http://www.edukanda.es/mediatecaweb/data/zip/1159/page_21_2.htm

Documentació, tutorials, exemples i guies sobre Python3 en:
https://docs.python.org/3/
http://www.dotnetperls.com/built-in

Documentació i exemples de PySerial en:

http://pyserial.sourceforge.net/index.html

11. Annexos

11.1 Manual d'instal·lació de l'IDE de desenvolupament.

Sistemes Encastats

Manual de instal·lació del IDE
Python + Eclipse + Pydev + PySerial

Instal·lació de Python 3.4.3

Descarregar el programari de https://www.python.org/downloads/release/python-343/

Instal·lació de Python 3.4.3 en Linux Ubuntu 14

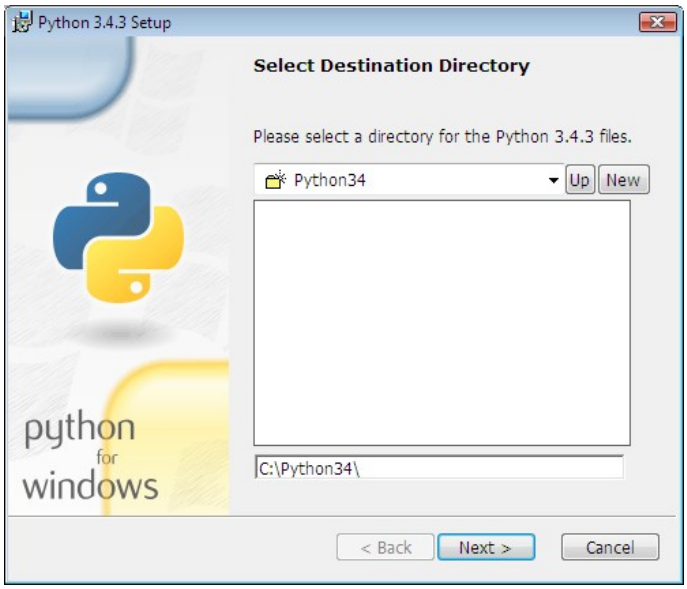
1. Moure el fitxer descarregat a la carpeta on es volguí instal·lar.
2. Descomprimir el fitxer (clicant al botó dret, i seleccionant extreure aquí) «Python-3.4.3.tgz»
3. Obrir una sessió del terminal (**tecles Ctrl + Alt + t**).
4. Amb la comanda CD situar-se a la carpeta d'instal·lació.
5. Introduir les següents comandes:

```
./configure
make
make test
sudo make install
```

Notes:
- El programari porta el fitxer README que conté les instruccions d'instal·lació mes detallades.
- El nom del fitxer, i les instruccions poden variar d'un versió a altre.

Instal·lació de Python 3.4.3 en Windows Vista

1. Fer botó dret al fitxer, i seleccionar instal·lar. Clicar al botó «next»:
2. Seleccionar el directori on es vol instal·lar:



3. Prémer el botó «Next».

5. Prémer el botó «Finish».

Instal·lació de Eclipse 3.8

Instal·lació en Linux Ubuntu 14 i en Windows Vista

1. Descarregar el programari de:

   https://www.eclipse.org/downloads/

2. Moure el fitxer descarregat a la carpeta on es volguí instal·lar.
3. Descomprimir el fitxer (clicant al botó dret, i seleccionant extreure aquí):

   Fitxer en Linux: «eclipse-java-luna-SR2-linux-gtk-x86_64.tar.gz»
   Fitxer en Windows 32 bits: «eclipse-java-luna-SR2-win32.zip»

4. Obrir, fent doble click amb el ratolí a la icona del eclipse.

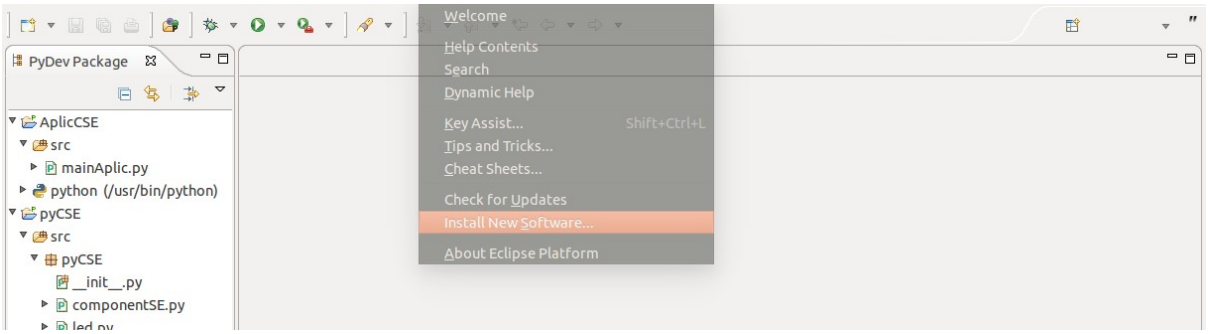
Notes:

- El nom del fitxer pot variar d'un versió a altre.

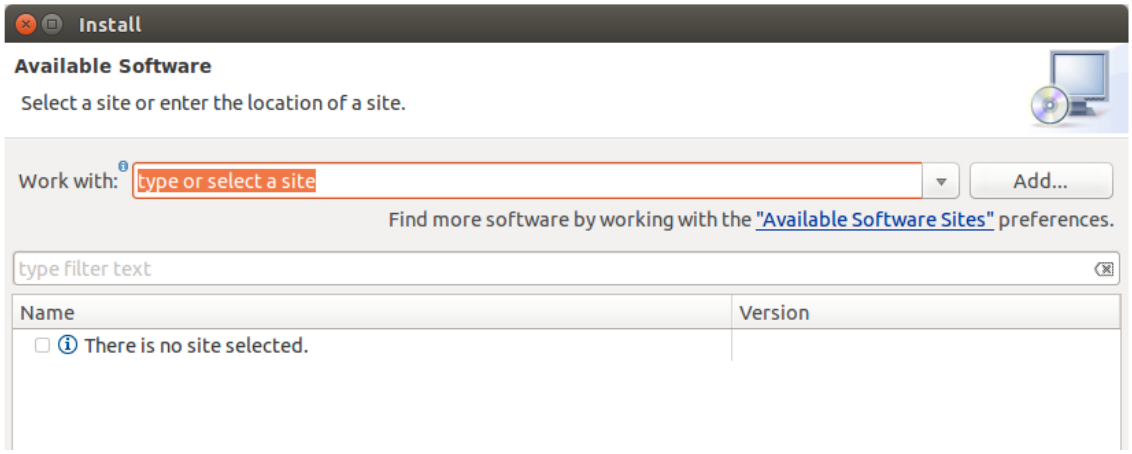
Instal·lació de PyDev

Instal·lació en Linux Ubuntu 14 i en Windows Vista

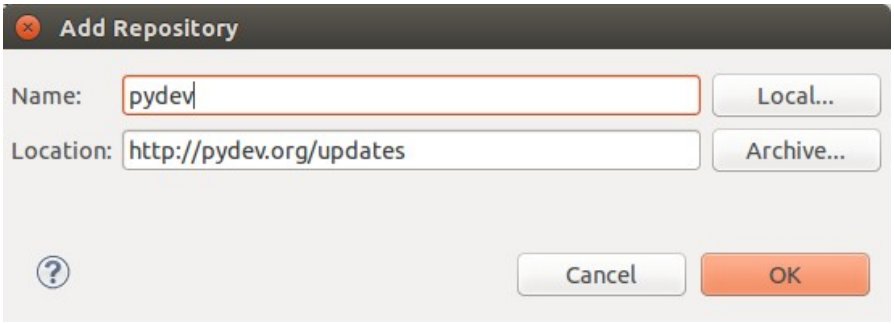
1. Es necessari haver instal·lat prèviament el Eclipse (veure apartat anterior).
2. Es necessari tenir instal·lat Java 7 o posterior.
3. Obrir l'aplicació eclipse.
4. En el menú superior, i dins del menú «**Help**», seleccionar «**Install New Software**».



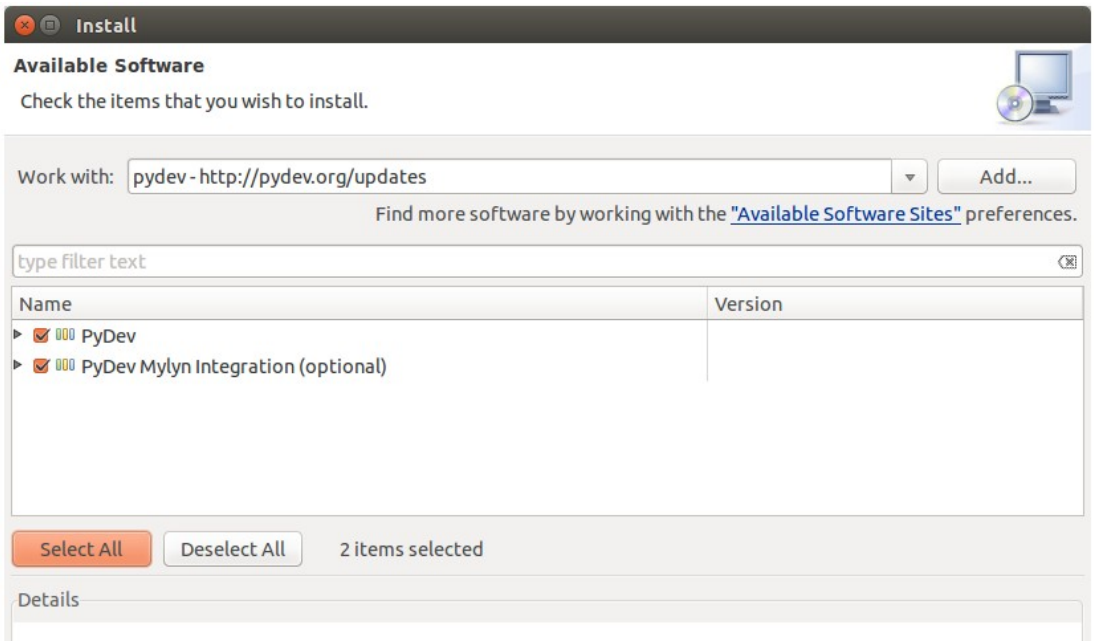
5. Polsar el botó **«Add»** de la part superior dreta.



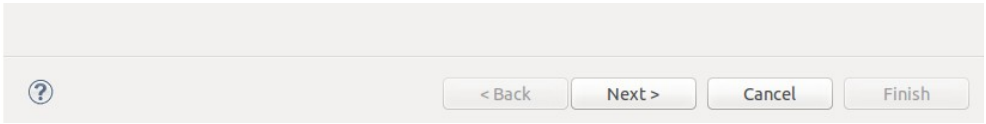
6. Afegir la URL http://pydev.org/updates



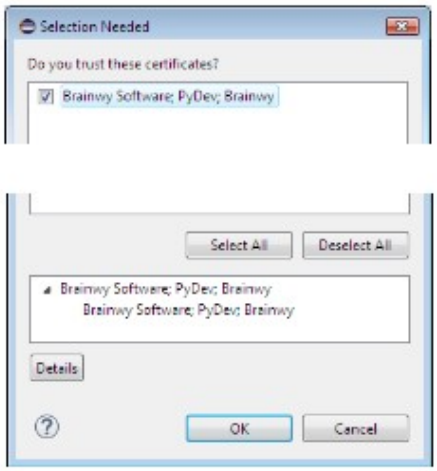
7. Polsar **«OK»**, i esperar que es carregui el modul.
8. Seleccionar el paquets que es desitgin instal·lar (com a mínim PyDev i PyDev for Eclipse).



9. Prémer el botó inferior **«Next».**

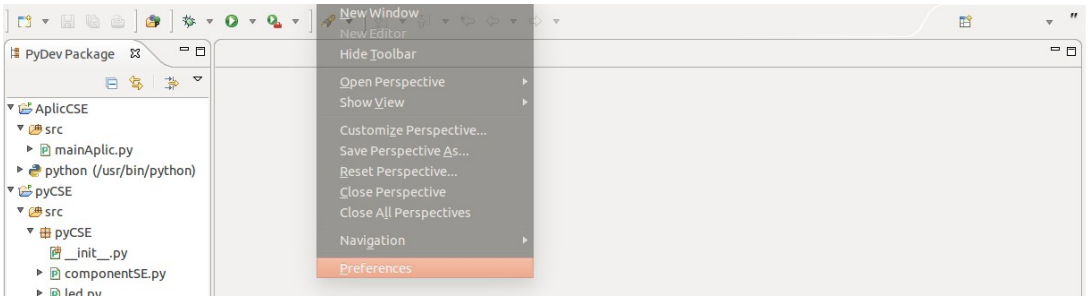


10. Tornar a prémer el botó **«Next».**
11. Acceptar el termes legals i prémer **«Finish».**
12. La instal·lació demanara acceptar el certificats de Aptana PyDev (en Linux) o Brainway (en Windows). Seleccionar i prémer **«OK»**.



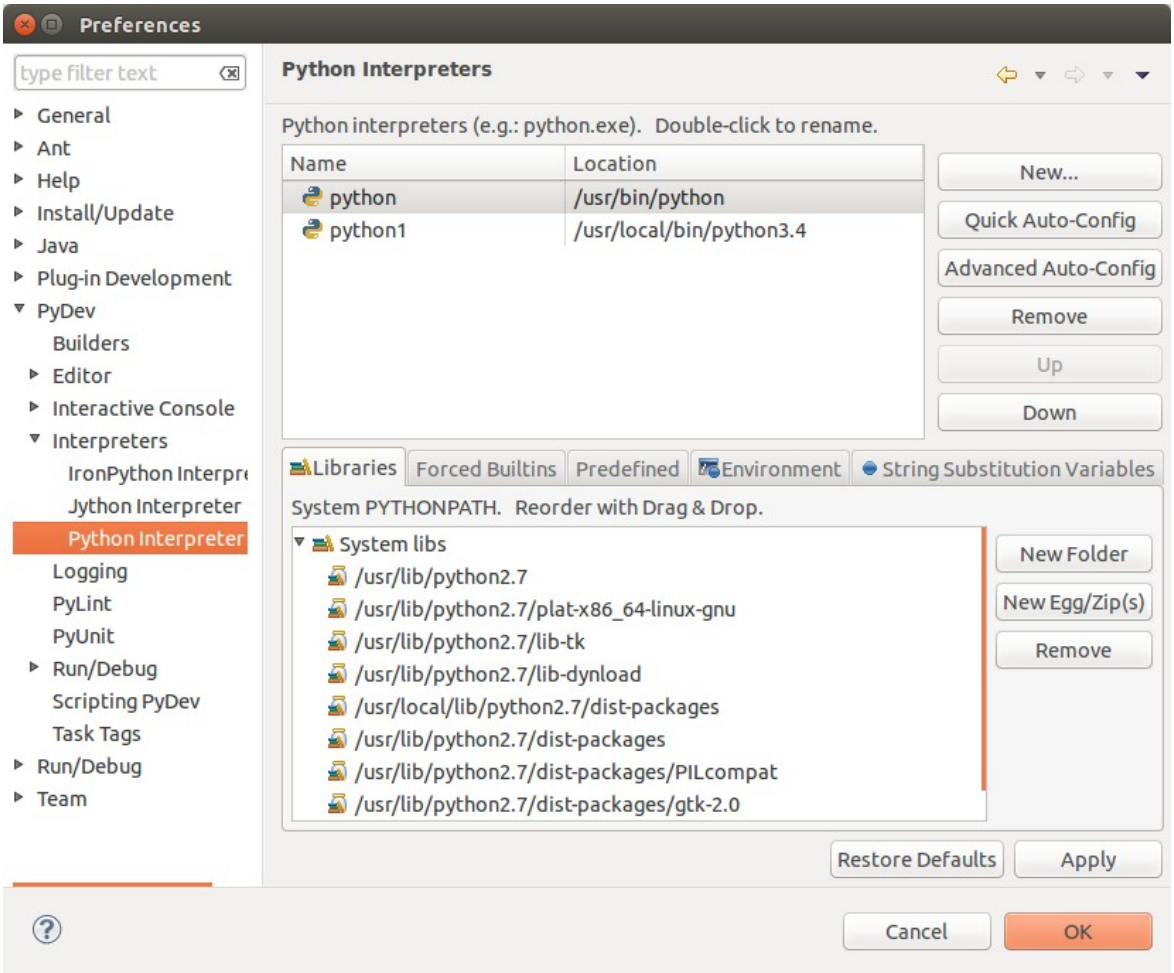
13. Reiniciar Eclipse

14. Anar al menú superior, i dins del menú «**Windows»**, seleccionar «**Preferences».**



14. Desplegar PyDev, i dins desplegar **«Interpreters»**.

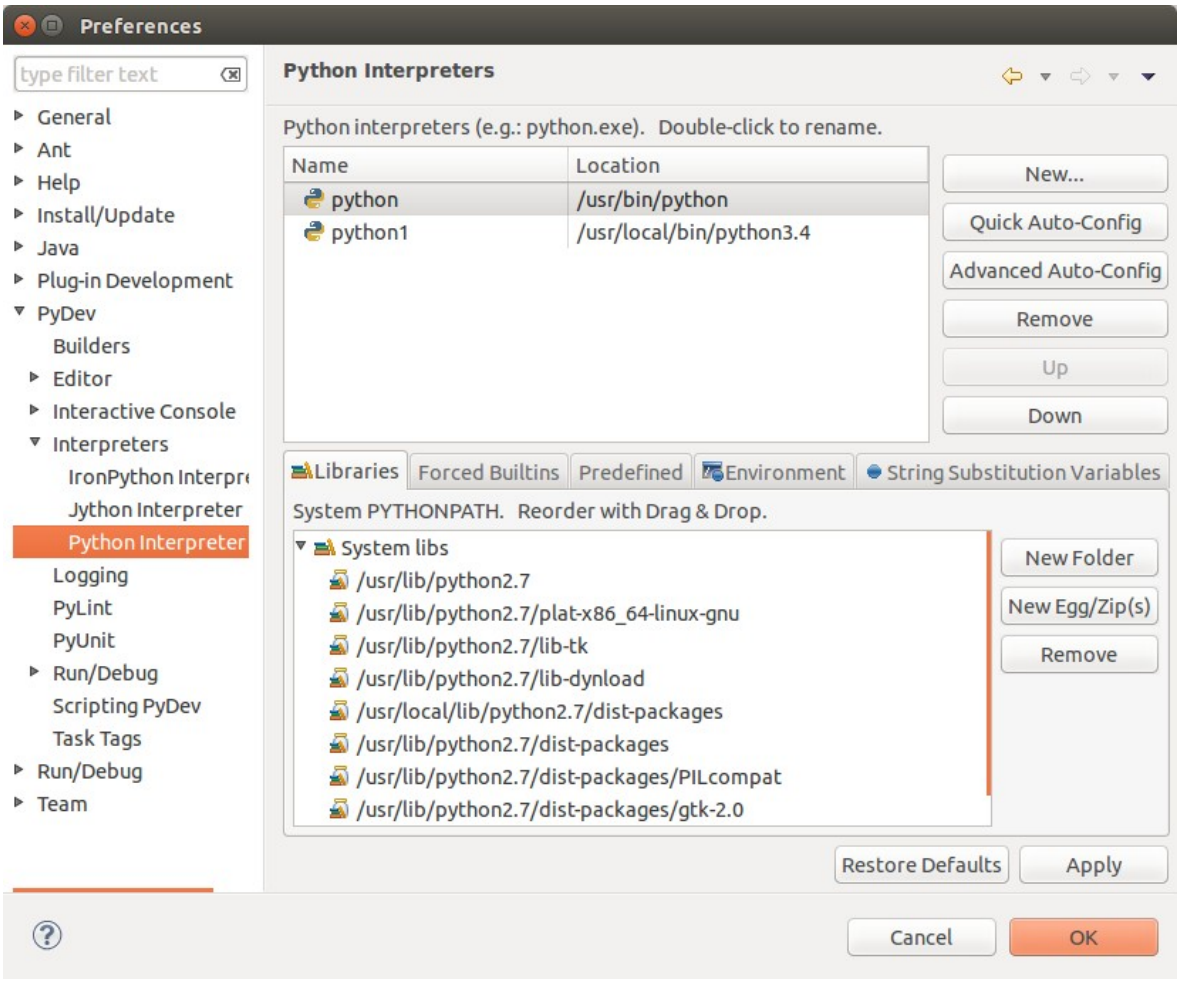
15. Seleccionar **«Python Interpretet»** . Premer el botó **«New»**.



16. Donar-li un nom, i seleccionar la ruta on esta instal·lat Python, per defecte:

    /usr/local/bin/python3.4

17. Prémer **«OK»**, **«Apply»** i **«Ok»**.:



Informació extreta de http://www.pythondiario.com/2013/06/eclipse-y-pydev-configuracion-del-ide.html

Instal·lació de PySerial 2.7

Instal·lació en Linux Ubuntu 14 i en Windows Vista

1. Descarregar el programari de:

   https://pypi.python.org/pypi/pyserial

2. Descomprimir el fitxer.
   En Linux: pyserial-2.7.tar.gz
   En Windows: pyserial-2.7.win32_py3k.exe

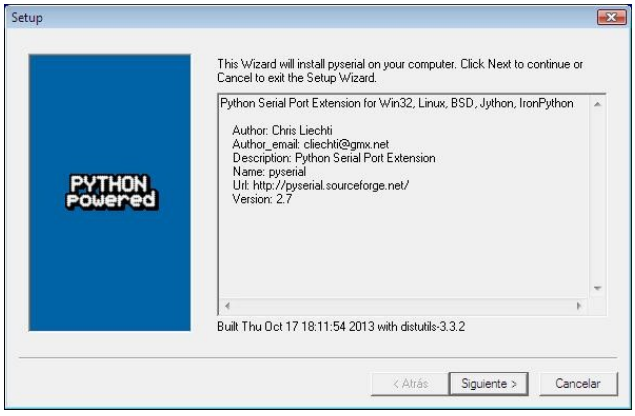
Instal·lació en Linux

3. Obrir una sessió del terminal (**tecles Ctrl + Alt + t**).
4. Des de el terminal entrar a la carpeta pyserial-2.7.
5. Escriure la següent comanda:

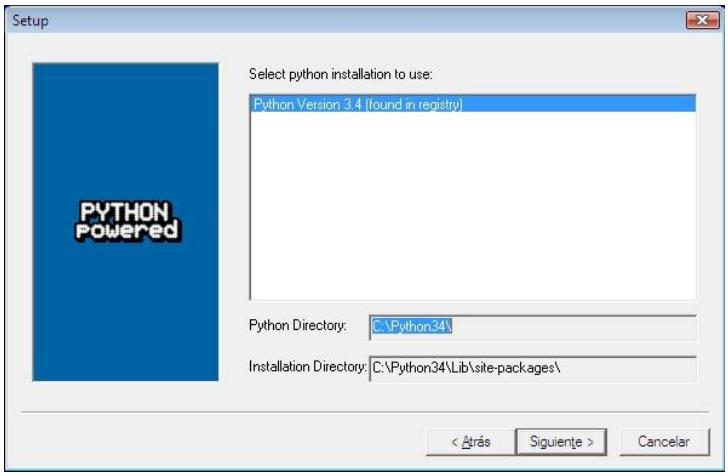
```
python3 setup.py install
```

Instal·lació en Windows Vista

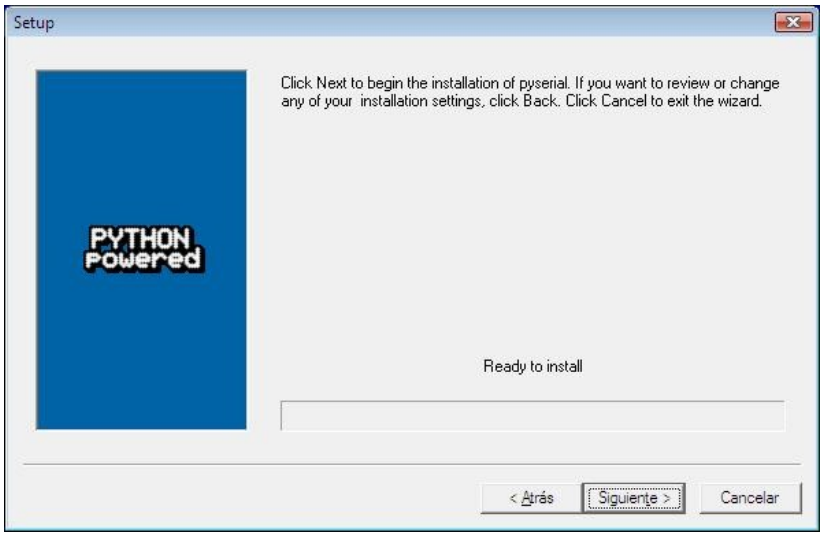
3. Fer doble click al fitxer de instal·lació, i prémer **«siguiente»**.



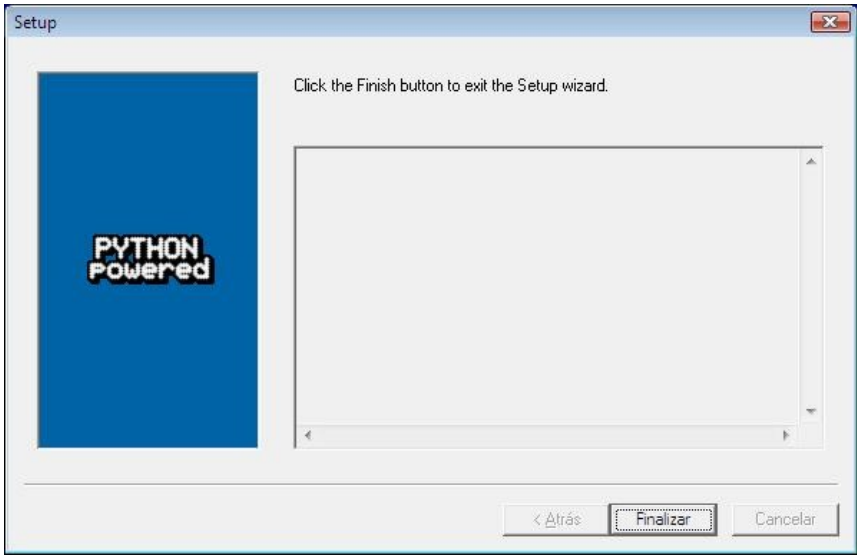
4. Seleccionar el directori on esta instal·lat Python..



5. Prémer **«siguiente»**.



6. Prémer **«Finalizar»**.



Configuració del port Serie a Linux

En entorns Linux es possible que el sistema operatiu no deixi accedir al port serie, si l'usuari no te assignats permisos d'accés. Per sol-ventar això:

Activació de permisos del port serie ttyUSB0

1. Obrir una sessió del terminal (**tecles Ctrl + Alt + t**).
2. Introduir la següent comanda.

```
sudo chmod 666 /dev/ttyUSB0
```

3. Introduir la contrasenya del administrador.

Activació de permisos del port serie ttyUSB0

Accès al Port Serie amb pySerial

Tota la documentació de pySerial, i de com accedir al port serie USB, es troba a la pàgina:

http://pyserial.sourceforge.net/

PySerial permet accedir al port serie, des de Python, igual que si fos un fitxer.
Les principals funcions definides a la classe serial son:

Serial(): Per obrir la comunicació, Rep 3 paràmetres (port, velocitat i temps limit)
write(): Per enviar dades. Rep per paràmetres les dades, en format String o Bytearray.
read(): Per llegir dades del port serie. Rep com a paràmetre la mida, i retorna les dades.

Exemple de iniciació de la comunicació:

```
try:
    self.ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=10)
except(OSError):
    print("Port Serie no disponible")
    self.ser = 0
```

Exemple de enviament de dades:

```
if self.ser != 0:
    self.ser.write(bytestream)
```

Exemple de lectura de dades:

```
if self.ser != 0:
    if self.ser.readable():
        dades = self.ser.read(128)
    else:
        print('no disponible')
```