Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

DESENVOLUPAMENT D'UNA INTERFÍCIE HARDWARE I SOFTWARE MITJANÇANT USB PER A PLATAFORMES ANDROID

Pablo Buenaposada Sanchez

Directors: José Bosch Estrada
Juan Daniel Prades García
Realitzat a: Departament d'Electrònica.
UB

Barcelona, 20 de juny de 2013

ÍNDEX

1. IDEA

¿Quina és la idea del projecte?

La idea del projecte és crear una interfície que es comuniqui amb un dispositiu Android mitjançant una connexió USB i Bluetooth.

La interfície serà un microcontrolador prèviament programat que serà capaç de comunicar-se amb una aplicació Android, de manera que podrem utilitzar els recursos del microcontrolador per poder d'alguna manera realitzar una interacció amb el món.

La gracia de tot això és que l'usuari només ha de programar el dispositiu Android per fer us del projecte, de manera que mitjançant una llibreria pugui escriure un codi molt senzill que interactuï amb el microcontrolador.

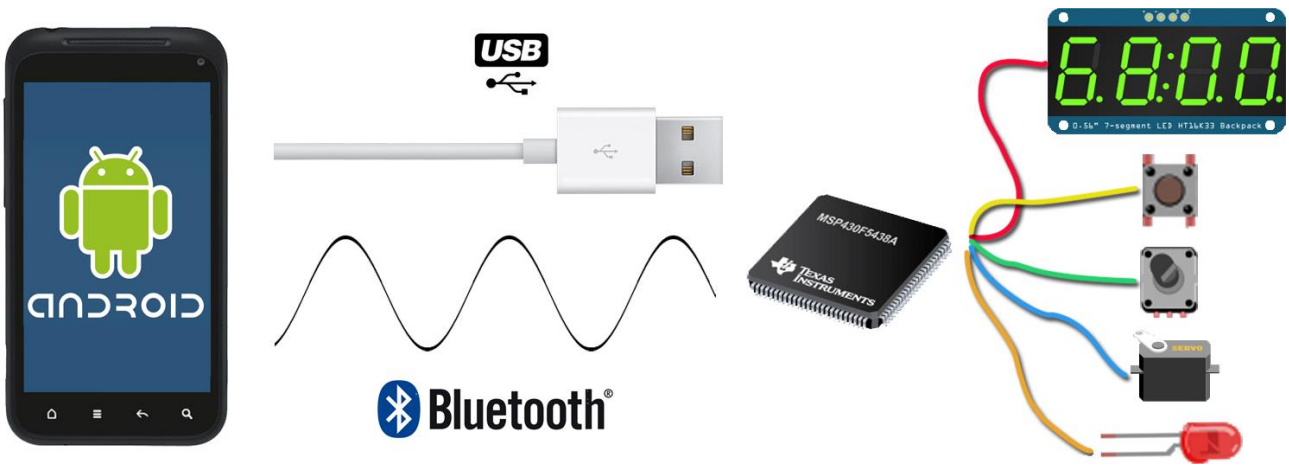


Figura 1: Representació de les diferents connexions entre el dispositiu Android i el microcontrolador

1.1 IDEA

What's the idea of the project?

The idea of the project is create a hardware and software interface to allow the acquisition of external parameters and control some of them from an Android device. The communication between this hardware and the Android device could be through USB connection and Bluetooth.

The hardware interface will be controlled by a microcontroller previously programmed to be able to communicate with an Android application, in order to use all the microcontroller resources for interact with the world.

The idea of this is that the user only needs to code in Android to use the system, so using a library will be able to write a very simple code that interacts with the microcontroller.

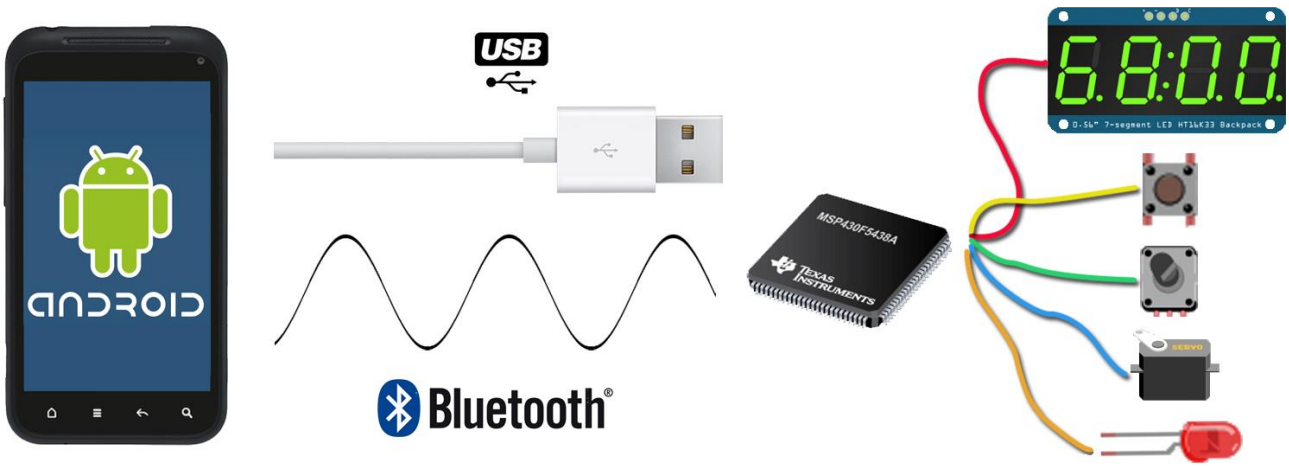


Figure 1: Representation of the different connections between the Android device and the microcontroller

2. OBJECTIUS

-L'objectiu bàsic és desenvolupar el hardware i software per poder gestionar l'adquisició de dades i control de processos des de dispositius Android.

-Enllaçar el microcontrolador a un dispositiu Android per USB i Bluetooth

-Crear un protocol de comunicació

-Crear una llibreria Android (Java) per controlar el microcontrolador

-Programar el microcontrolador per a que entengui la llibreria anterior (*Firmware*[*])

-Dissenyar una placa que contingui el microcontrolador i tots els dispositius necessaris per poder treballar amb el projecte

-Mapejar el màxim número de recursos del microcontrolador. És a dir, fer accessible el màxim número de recursos del microcontrolador des de les aplicacions desenvolupades al dispositiu Android.

-Programació de tasques de recollida de dades sense necessitat d'estar sempre connectats, el que denominarem treballar *off-line*

[*] El **firmware** és el software que té directa interacció amb el hardware, es troba gravat en una memòria (ROM, EEPROM, flash, etc), que estableix la lògica de més baix nivell. Controla els circuits electrònics d'un dispositiu de qualsevol tipus.

2.1 GOALS

-The main goal is develop the hardware and software to manage data acquisition and process control from Android devices

-Link the microcontroller to the Android device through USB and Bluetooth

-Define a communication protocol

-Create an Android Library (Java) to control the microcontroller

-Program the microcontroller to understand the previous library (*Firmware*[*])

-Design a PCB** containing the microcontroller and the rest of devices needed, to be able to work with the project

-Map the maximum number of resources of the microcontroller from the Android Applications developed using the libraries created in this project

-The possibility that the microcontroller hardware could work off-line, executing a sequence programed previously from the Android device.

---

[*] The **firmware** is the software wich has direct interaction with the hardware, is found burned in a memory (ROM, EEPROM, flash, etc) and establishes the low-level logic. It controls the electronic circuits of any kind of device.

[**] **PCB** stands for Printed Circuit Board

3. ANTECEDENTS

Actualment existeixen al mercat diferents tipus de hardware que estableixen una comunicació bidireccional entre un dispositiu Android i un microcontrolador, tot i que encara és un tema novedós, cada dia surten més opcions comercials.

Parlarem bàsicament de les dues opcions que més repercussió han aconseguit, que són d'Arduino i IOIO.

3.1 Arduino

Neix al 2005 a l'institut de disseny interactiu d'Ivrea a Itàlia com a projecte per a controlar petits dispositius construïts pels propis estudiants sobre una plataforma més moderna i més barata que les opcions que hi havia en aquell moment.

Arduino proporciona un hardware de propòsit general i un llenguatge de programació molt simple basat en C y Processing.

El microcontrolador que van fer servir és un Atmel AVR[1], de baix cost i de baix consum, encara que, en l'actualitat les noves versions d'Arduino ja utilitzen microcontroladors més nous i potents.

Tant el software com el hardware es troben sota llicencies de disseny i distribució lliures, això ha fet que es generi una comunitat molt important que comparteix els seus desenvolupaments i coneixements fins al punt d'utilitzar la plataforma per a base d'altres projectes.

La clau de l'èxit ha estat l'encert d'oferir:

Baix cost: Una de les versions més utilitzades d'Arduino és l'anomenada UNO, amb un preu de 20€. Un preu molt competitiu tenint en compte que no es necessita res més per començar a treballar amb ell.

Simplicitat: Tant al seu software com al seu hardware. Arduino ens brinda un llenguatge de programació i una llibreria molt simples i molt semblants a C d'alt nivell, en contraposició dels llenguatges emprats als microcontroladors fins llavors, com l'Ensamblador, i sobretot sense la necessitat de configuracions de multitud de registres que necessiten uns grans coneixements per tal de programar tasques senzilles al microcontrolador.

Obre un nou camp a tothom que tingui uns coneixements bàsics de programació.

El seu hardware compleix la mateix filosofia, connexió USB tant per programar-lo com per alimentar-lo, i uns microcontroladors de baix cost que en molts casos es permeten extreure de la placa per tal d'utilitzar-los directament en les teves pròpies plaques.

En general ha despertat un gran interès i es podria dir que actualment és l'exemple més clar de hardware lliure.

Arduino i Android

Les primeres i més conegudes versions d'Arduino no tenen suport directe d'Android.

Els entusiastes d'Arduino van començar a treballar en una manera de comunicar-se amb Android mitjançant l'USB.

En aquells moments l'única forma d'interactuar amb un dispositiu Android de manera remota per l'USB era fent servir l'*Android Debug Bridge* o ADB.

3.2 Android Debug Bridge

Es tracta d'un sistema disponible des de la versió 1.5 d'Android, que permet invocar instruccions remotament des de un PC mitjançant línia de comandes.

El tipus de comandes que es permeten fer són del tipus navegació pel sistema de fitxers, incloent-hi eliminació i inserció de fitxers, instal·lar aplicacions o la que ens interessa a nosaltres: poder crear un enllaç bidireccional mitjançant una connexió TCP per USB.

L'ADB també és el sistema empleat per retornar-nos els missatges d'error i de depuració a l'hora de depurar una aplicació.

La característica de crear un enllaç TCP ens obre una connexió client-servidor amb el dispositiu Android. El problema és que en comptes d'usar un PC a l'altre costat de la comunicació, nosaltres tenim un microcontrolador (Arduino) i el port USB d'ordinador és USB host mentre que l'Arduino no té cap port d'entrada USB Host.

3.3 Què significa USB Host?

USB Host és el hardware que ofereix la capacitat de gestionar una comunicació USB entre dos dispositius. És el que fa les tasques de dispositiu mestre a les comunicacions, en quest cas USB. Aquesta característica necessita d'electrònica addicional per ser capaç d'interpretar el dispositiu que es connecta a més d'alimentar-lo elèctricament.

El que es va començar a fer és crear una placa d'expansió USB Host per a sistemes Arduino, de manera que mitjançant una petita llibreria es pot connectar un dispositiu Android a qualsevol Arduino i comunicar-se entre ells mitjançant la connexió TCP que ofereix l'ADB.

Una de les plaques (shields) més famoses en fer això va ser la que va crear el rus Oleg Mazurov de Circuits@Home utilitzant el xip MAX3421E[2].

Figura 2: Oleg USB Host shield a sobre d'un Arduino UNO

Aquesta solució va ser en un moment l'única manera d'extreure una comunicació Android via USB, però no cal dir que és una solució fruit de l'esperit 'hacker' de la comunitat, ja que en un principi l'ADB no s'havia pensat per això i a més podria portar problemes de seguretat ja que s'ha de tenir activada l'opció de depuració USB.

3.4 Google fa arribar Android a Arduino

Google va veure tot això com una oportunitat per a la creació de hardware que interactués amb dispositius Android i al 2011 va presentar el que s'anomena *Accesory Development Kit* també conegut com ADK, disponible nativament des de la versió 3.1 d'Android.

L'ADK es tracta d'una plataforma per desenvolupar tota mena de dispositius que interactuïn amb altres dispositius Android.

Consta d'un paquet de software anomenat *Android Open Accessory* (AOA) que permet la comunicació via cable USB amb un hardware obert.

La sorpresa va ser que el hardware del kit era un Arduino Mega 2560 modificat afegint-hi un port USB Host ja integrat a la placa.

El que va fer Google bàsicament va ser aprofitar la llicencia gratuïta d'Arduino i fer la seva pròpia placa manufacturada al Japó, amb un preu de sortida de 400$.

Arduino no va trigar a respondre fent bàsicament la mateixa placa amb un preu inferior (49€) totalment compatible amb aquest ADK, el nom que li van posar va ser Arduino ADK.

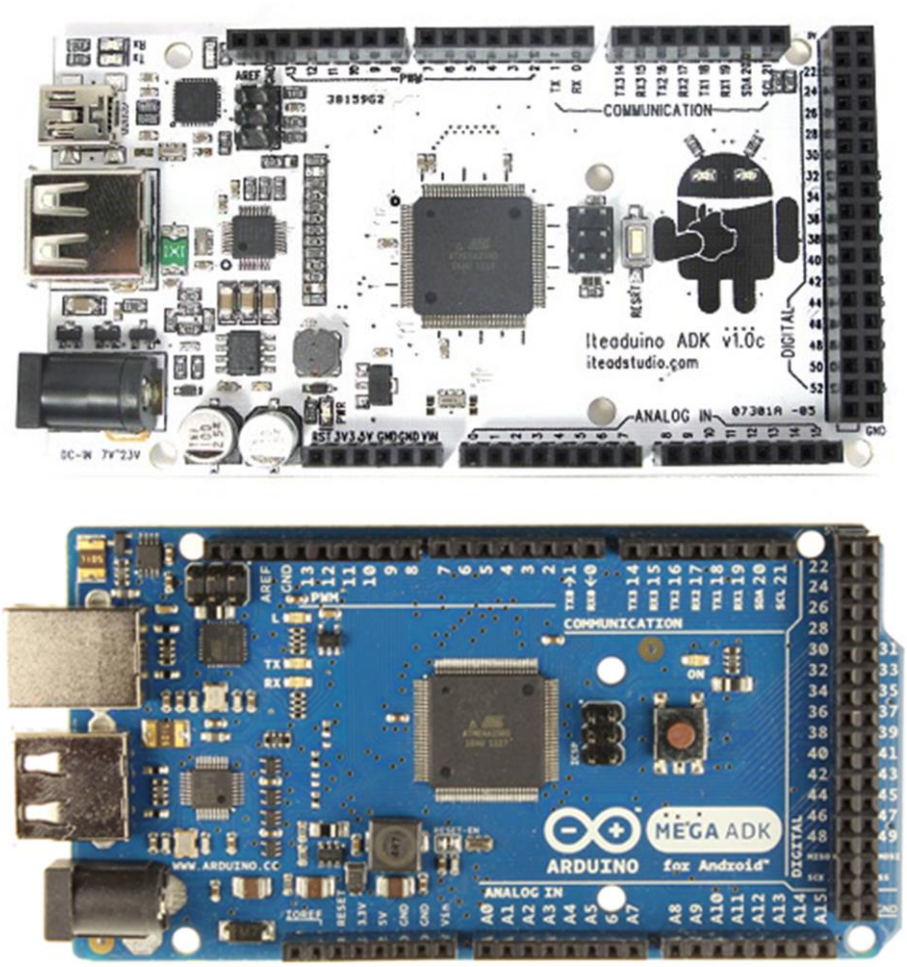


Figura 3: Google ADK (2011) a dalt, Arduino ADK a baix

Al següent enllaç es pot llegir l'explicació del que va passar des de el punt de vista de David Cuartielles, membre del equip de desenvolupament d'Arduino http://david.cuartielles.com/b/2011/05/google-adk-que-bicho-es-ese/

Amb el que ens hem de quedar de tot això és que Google va veure tot el que s'estava desenvolupant i va ser capaç d'adoptar la solució de la comunitat i el que és més important, ha "legalitzat" tota una branca de desenvolupament de hardware on tothom utilitzarà el mateix estàndard i es comunicarà sobre un protocol molt millor que el que s'usava mitjançant l'ADB.

3.5 IOIO

Creada per l'Israelià Ytai Ben-Tsvi al 2011 amb la col·laboració de SparkFun (empresa que ho comercialitza).

La IOIO també és de software i hardware lliure, pel moment només existeix un únic model, que ha anat sofrint petites revisions al seu hardware, té un preu de 40$.

El hardware emprat és un microcontrolador PIC24 que ja incorpora USB Host, probablement és la placa que suporta Android més petita fins al moment.

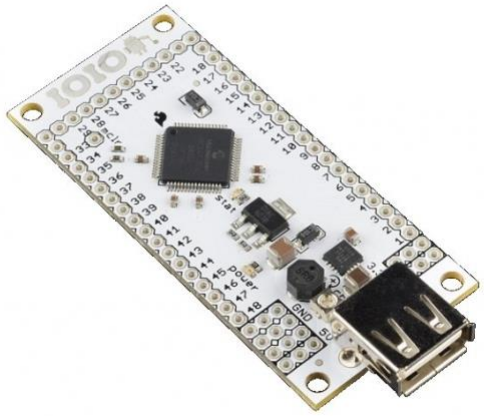


Figura 4: IOIO v1

Permet tant la comunicació per USB com per Bluetooth (afegint un Bluetooth dongle USB, no tots són compatibles) sense fer cap modificació al codi i podent canviar físicament d'una a l'altre en qualsevol moment sense aturar res (*hot swap*).

Es pot seleccionar que la IOIO faci us de la comunicació per ADB o per ADK, de manera que podrem sempre fer-la servir a partir de dispositius Android 1.5 cap endavant o aprofitar-nos dels avantatges de l'ADK a partir de dispositius amb Android 3.1.

La diferència principal amb l'Arduino és que només s'ha de saber un únic llenguatge de programació, Java en aquest cas, ja que l'aplicació es programa com una aplicació qualsevol d'Android i s'accedeixen al recursos de la IOIO mitjançant una llibreria, de manera que el codi realment s'està executant al dispositiu Android. Això fa que només podrem fer us de la placa quan el dispositiu Android estigui connectat a la placa i no podem programar la placa per a que funcioni independent com l'Arduino ADK.

Ytai actualment treballa a Google i IOIO es troba en constant actualització tant en hardware com en software.

3.6 BACKGROUND

Nowadays there are in the market different hardware platforms allowing a bidirectional communication between an Android device and a microcontroller. Despite this is still a recent topic, every day appears more commercial options.

We will talk mainly about the two options that showed more impact in the scene, we are talking about Arduino and IOIO.

3.7 Arduino

Born in 2005 at the institute of interactive design of Ivrea at Italy as a project to control small devices build by the students on a platform newer and cheaper than the options that had at that time.

Arduino provides general purpose hardware and a very simple programming language based in C and Processing.

The microcontroller used is an Atmel AVR[1], is low cost and low consumption although in the newer versions of Arduino they are using newer and more powerful microcontrollers.

Both, hardware and software are under free design and distribution license, that has made the grow up of a so important community that shares his developments and knowledge to the point of using the platform for the base of other projects.

The key of the success of Arduino resides in two points:

Low cost: One of the most used versions of Arduino is the one called UNO, with a price of 20€. A very competitive price given you don't need nothing else to start working out of the box.

Simplicity: Either his software or his hardware. Arduino gives a very simple programming language and a library similar to C, versus the languages used normally to code microcontrollers like Assembler with large amounts of register configurations that need a high knowledge to code easy tasks at microcontroller.

Arduino opens a new scene to everybody with basic knowledge of programming.

His hardware meets with the same philosophy, USB connection for programming and powering it, and a low cost microcontroller that you can use them out of the Arduino boards to your own boards.

In general seems that it awakens interest, we can say that is the clearest example of free hardware.

Arduino meets Android

The firsts and more known versions of Arduino didn't have direct support to Android.

The Arduino enthusiasts began working on a way to communicate with Arduino through USB.

At that time the only way to interact with an Android device through USB was using the Android Debug Bridge also known as ADB.

3.8 Android Debug Bridge

It's a system available since 1.5 version of Android that allows the user to invoke instructions remotely from a PC using a command line.
The types of commands that are allowed are the ones like browsing the file system (including insertion and deletion of files), install applications or the one that interests us, the ability to create a bi-directional pipe using a TCP connection through USB.
The ADB is also the system used to return us error and debug messages when we are debugging an application.
The feature of establish a TCP connection, create us a client-server link with the Android device, the problem is that instead of using a PC in the other side of the connection, we want to use a microcontroller (Arduino).
The main problem we have is that the USB port of computer is USB host and Arduino hasn't a USB Host port.

3.9 What does USB Host mean?

USB Host is the hardware that provides the capability to manage a USB communication between two devices, it is the master device in the communication process. This feature requires additional electronics to be able to interpret the device that connects to and power it electrically.

What people started doing is creating an USB Host expansion board for Arduino systems, so using a small library could connect an Android device to any Arduino, communicating with each other using TCP offered by ADB.

One of the most famous boards (shields) that do this is the one created by the Russian Oleg Mazurov from Circuits@Home using the MAX3421E chip[2].

Figure 2: Oleg USB Host shield on Arduino UNO

This solution was at that time the only way to extract an Android communication via USB, but this solution is the result from the hacker spirit of the community since initially the ADB is not thought for this and could bring security problems since you have to turn on *USB debugging* option.

3.10 Google approaches Android to Arduino

Google saw this as an opportunity to create hardware that interact with Android devices so in 2011 introduced what is called Accessory Development Kit also know as ADK, available natively since version 3.1 of Android.

The ADK is a platform to develop any kind of device to interact with other Android devices. It is a software package called Android Open Accessory (AOA), which enables communications through an USB cable with open hardware.

The surprise was that the hardware kit was an Arduino Mega 2560 modified by adding a USB Host already integrated on the board.

What Google did was basically take the Arduino free license and make his own board manufactured in Japan, with a starting price of $ 400.

Arduino answered quickly doing basically the same board with a lower price (49€) fully compatible with ADK, they called the board Arduino ADK.

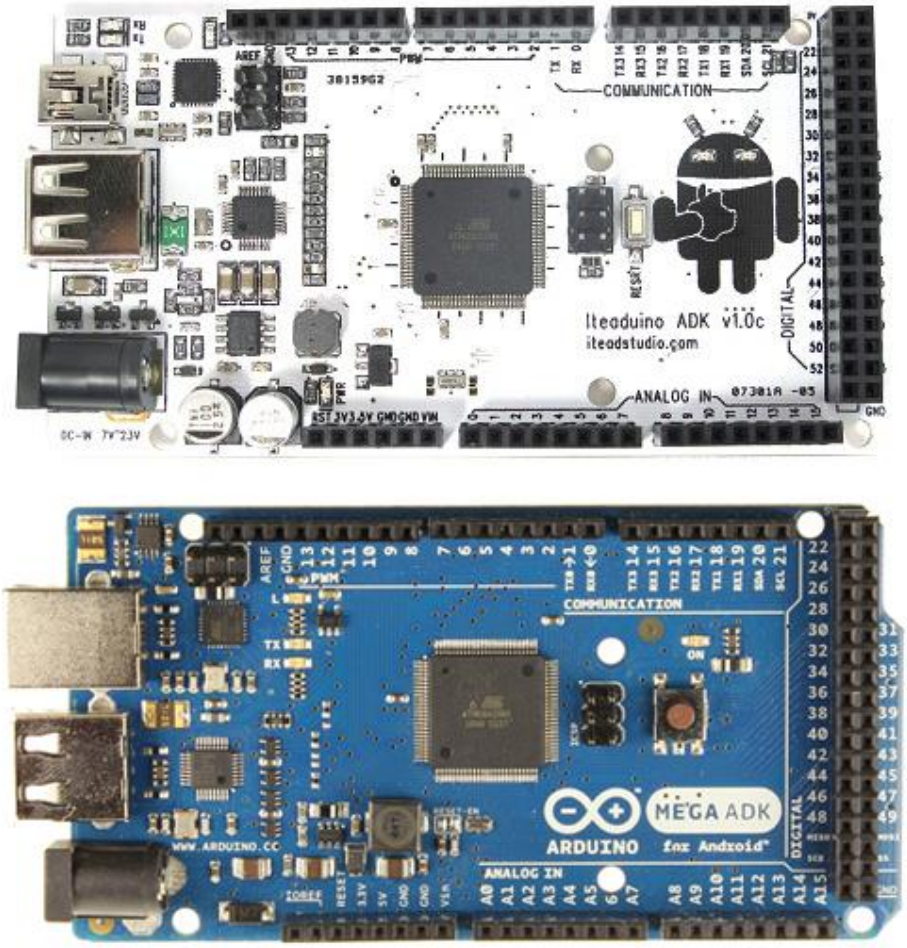


Figure 3: Google ADK (2011) on top, Arduino ADK at the bottom

At the following link you can read the explanation of what happened from the point of view of David Cuartielles member of the Arduino development team

http://david.cuartielles.com/b/2011/05/google-adk -bug-that-is-ese/

What we have to keep of all of this is that Google saw all that was being developed and was able to adopt the solution of the community and most importantly, has "legalized" an entire development branch of hardware where everyone will use the same standard protocol and communicate on a much better protocol than the ADB.

3.11 IOIO

It was created by the Israeli Ytai Ben-Tsvi at 2011 with the collaboration of Sparkfun (company that sells it).

The IOIO software and hardware are also free, at the moment there is only one model, which has been revised in his hardware, costs $40.

The hardware used is a PIC24 microcontroller that incorporates USB Host, is probably the smallest board that supports Android.

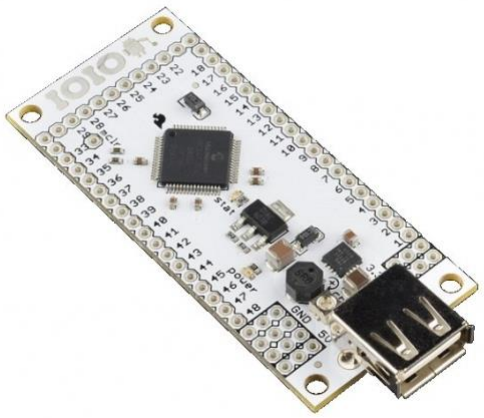


Figure 4: IOIO v1

It allows both, USB and Bluetooth communication (adding a Bluetooth USB dongle, not all are compatible) without modifications in the code, and we can physically change from one to the other at any time without switching off it (hot swap).

You can choose to make use of the ADB communication or ADK, so we can use it in any Android 1.5 or later devices or take advantage of the benefits from the ADK in devices since 3.1.

The main difference with the Arduino is that you will only need to know one programming language, Java in this case, because the application is coded as any Android application and we access the IOIO resources through a library, so the specific application code actually is running on your Android device. This means that we can only make use of the board when the Android device is connected to the board, we cannot program the board to work independently as the Arduino ADK.

Ytai currently works at Google and IOIO is constantly updated in hardware and software.

4. HARDWARE

4.1 Microcontrolador

El microcontrolador triat és un MSP430F5438 del fabricant Texas Instruments.

El motiu de l'elecció és perquè es tracta d'un microcontrolador de baix cost, capaç de cobrir les nostres necessitats. A més a més, ja hem treballat amb ell en una altra assignatura.

Les principals característiques que ens interessen són:

- CPU de 16 bits
- 16 bit Timer x 3. Per implementar bases de temps i per tant poder controlar processos temporalment.
- 12 bit ADC. Per poder fer mesures de magnituds analògiques.
- USCI x4 (UART,SPI, I^2C...) per implementar comunicacions sèrie
- Sortides PWM (Pulse With Modulation)
- DMA x3. Per transferir dades mesurades directament a memòria sense que treballi la CPU.
- Real-time clock
- 87 I/O pins
- Tensió d'alimentació entre 2.2 i 3.6v

Disposem d'una placa de desenvolupament, model MSP-EXP430F5438 que ja conté el microcontrolador anomenat abans, aquesta placa ens permet treballar amb més facilitat, disposa d'un connector JTAG pel qual podem programar el microcontrolador per USB.

A més disposa d'altres mòduls integrats com un petit LCD, acceleròmetre, polsadors i LEDs que ens serviran per depurar el software i fer proves de manera còmoda.

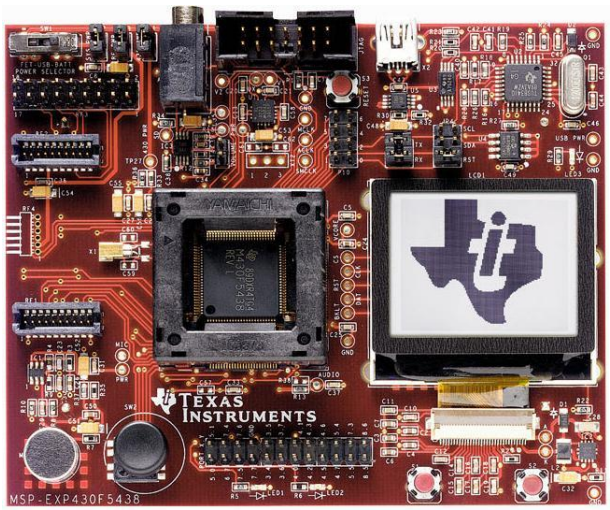


Figura 5: MSP-EXP430F5438

Figura 6: MSP430 USB-Debug-Interface 430UIF

4.2 Mòdul Bluetooth

El mòdul Bluetooth que farem servir és un HC-06, es tracta d'un mòdul molt econòmic i fàcil d'utilitzar que ens permetrà establir una connexió Bluetooth a través d'un port sèrie (RS232).

Encara que no hi ha gaire informació sobre ell, s'està començant a parlar molt ja que la gent l'està fent servir als seus projectes i es la opció més raonable en aquest moment.

El mòdul és de dimensions petites, té una bona cobertura i és configurable a través de comandes AT (comandes que es feien servir fa temps per la configuració de mòdems). Entre els diferents aspectes que podem configurar nosaltres destacarem 3:

1) el baud rate (velocitat de transferència)
2) el nom (identificador) del dispositiu quan el busquem
3) la contrasenya (PIN)

Un altra característica interesant d'aquest mòdul és que una vegada s'ha realitzat un enllaç amb un altre dispositiu és capaç de recordar-lo a la seva memòria i no sol·licita validació alguna (PIN 1234 per defecte). Si s'activa el pin 26 (KEY) aquesta informació s'elimina i el mòdul sol·licitarà una altra vegada validació del enllaç.

En quant a alimentació necessita 3.3v, els seu consum és baix, uns 8mA[3] en transmissió/recepció.

El nostre mòdul en qüestió està soldat a una placa base anomenada JY-MCU que ens permet una manipulació més simple ja que disposa d'uns pins on tenim totes les entrades/sortides que necessitem del mòdul, RX (recepció), TX (transmissió), VCC(alimentació), GND (massa), KEY(esborrar memòria i canviar el mode master/slave) i STATE (sortida per a saber si està emparellat o no), també incorpora un led connectat a la sortida STATE i un circuit que ens permetrà tenir més rang de voltatge a l'hora d'alimentar el mòdul, podrà ser de 3.3v fins a 6v.

El mòdul conjuntament amb la placa base, preparat per funcionar, té un cost de 6 euros i es troba fàcilment.

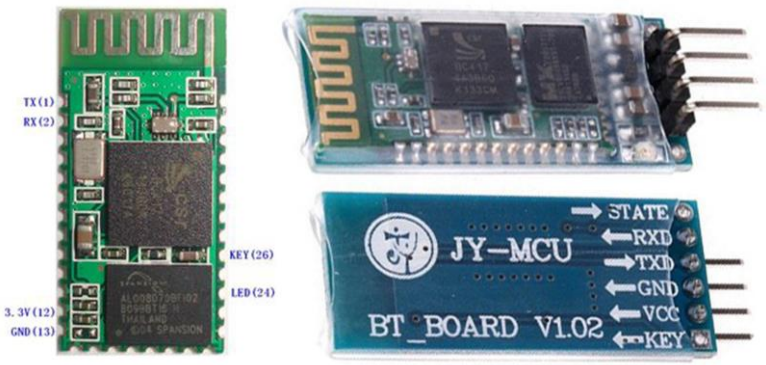


Figura 7: Mòdul Bluetooth HC-06 a l'esquerra i el mateix mòdul amb la placa base JY-MCU a la dreta

4.3 Mòdul USB Host

El microcontrolador que hem triat no té suport natiu per a oferir un port USB Host, per tant haurem d'afegir un hardware extern per tal de poder tenir aquesta capacitat.

Una de les opcions per solucionar aquest problema passa per utilitzar un xip de l'empresa Maxim, el MAX3421E, el qual ofereix un port USB Host 2.0. Aquest xip necessita una alimentació de 3.3v i es comunica amb altres dispositius mitjançant una connexió SPI de 3 o 4 cables, d'aquesta manera podem integrar una comunicació USB en qualsevol microprocessador que disposi de comunicació SPI, com és el cas del nostre MSP430.

Figura 8: Integrat MAX3421E

5. PROTOTIP

Tal i com fan els d'Arduino o IOIO nosaltres també hem volgut crear una placa que contingui el microcontrolador amb les seves entrades i sortides mitjançant connectors de fàcil ús; a més a més de tot el necessari per poder alimentar-lo externament amb una tensió d'entrada d'entre 7 i 30v.

El disseny a estat realitzat amb el programa DesignSpark PCB.

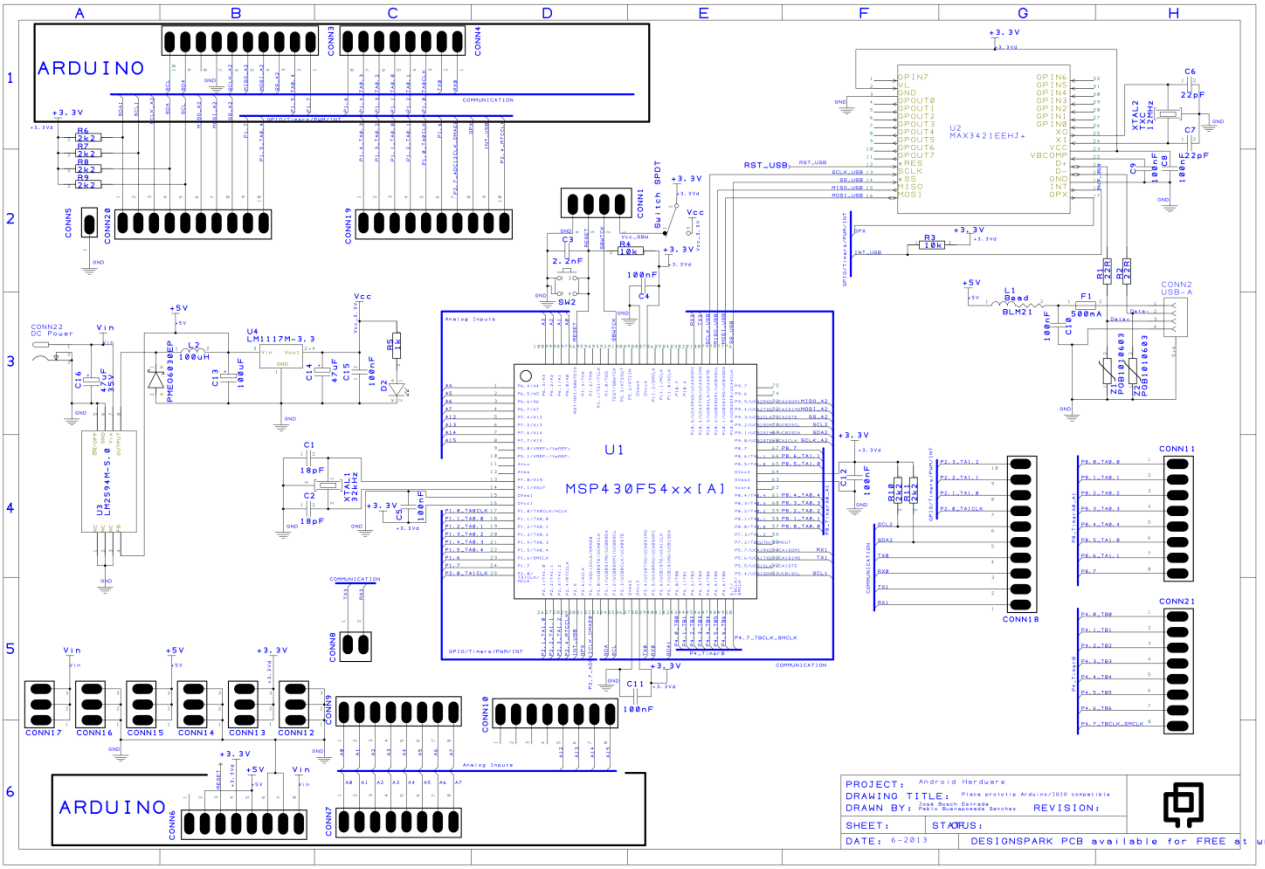


Figura 9: Esquema del prototip, es pot veure amb més detall a l'annex

Aquí podem veure una vista de la cara superior del PCB de com quedarà a la realitat:

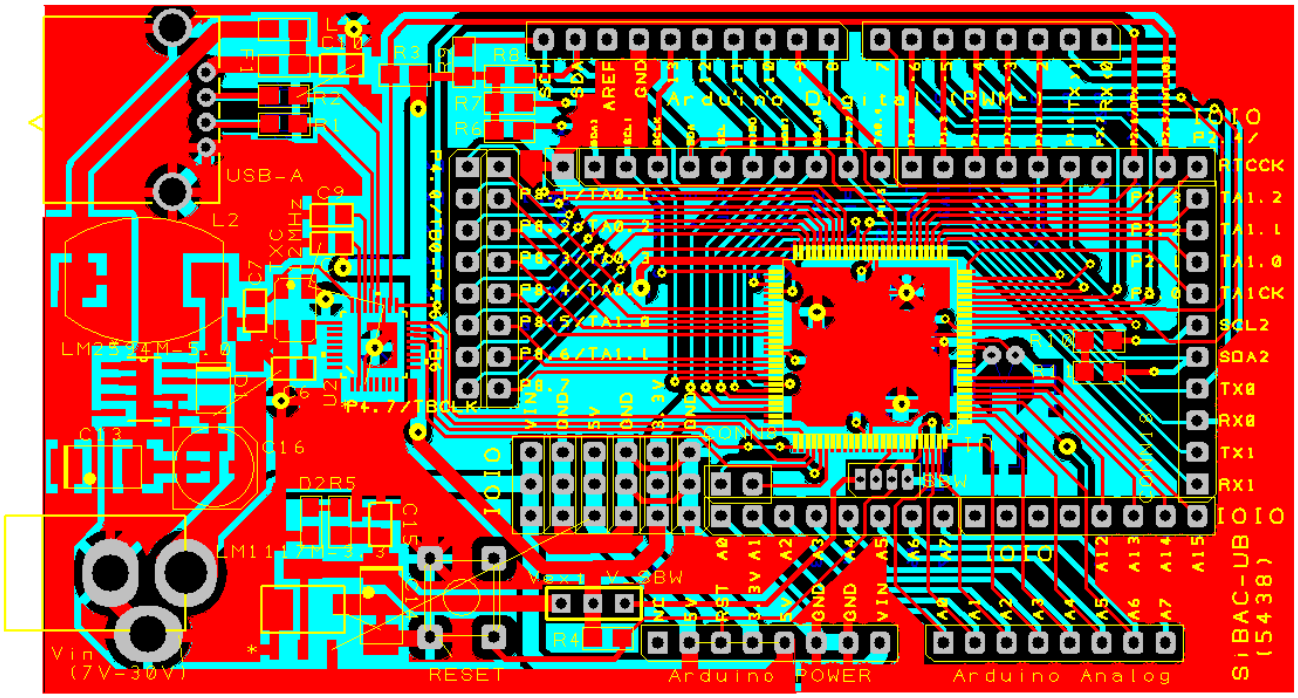


Figura 10: Prototip

Una característica important del disseny és que està pensat per a poder fer us de les *shields* disponibles per Arduino i IOIO, així que conté les dues formes característiques de les connexions d'aquests sistemes, a més del seu propi.

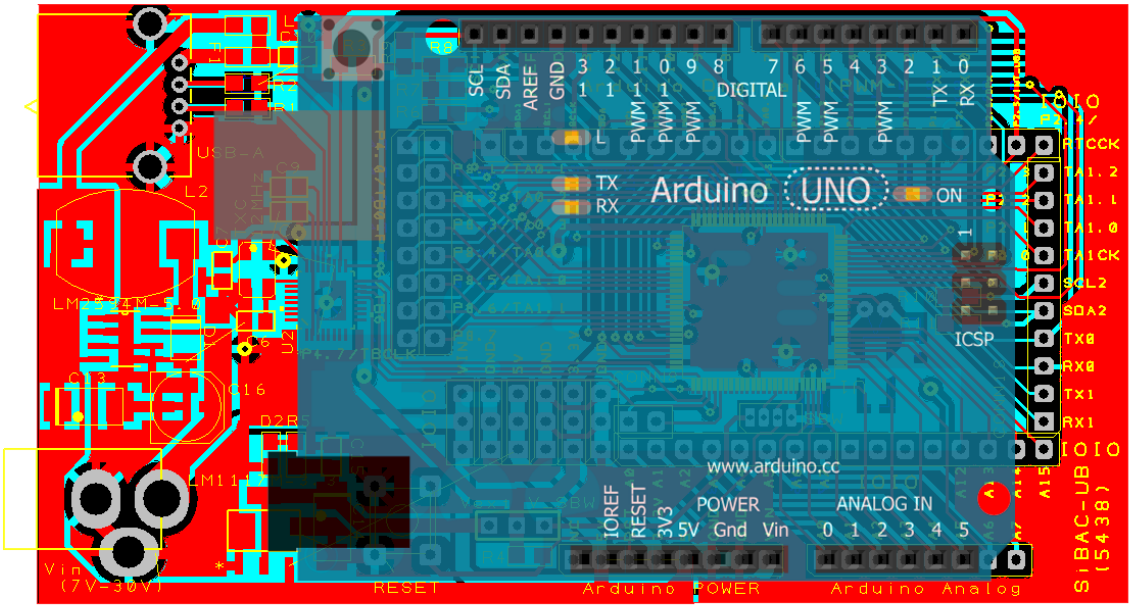


Figura 11: Arduino UNO a sobre de la nostre prototip

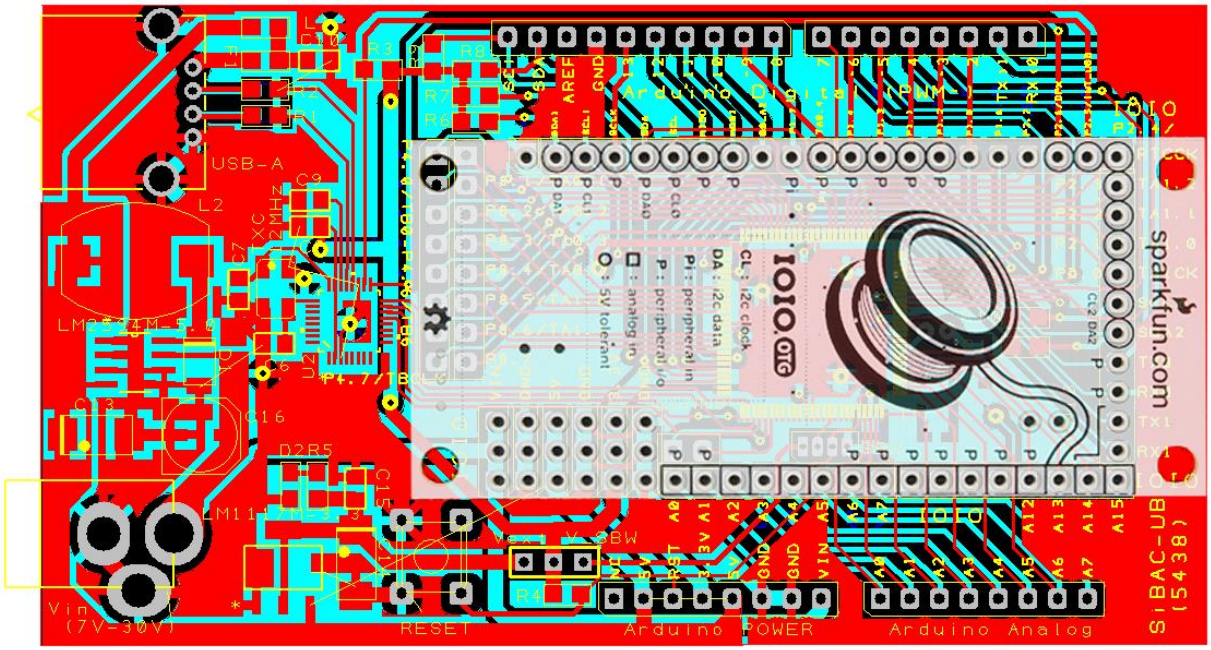


Figura 12: IOIO versió OTG a sobre de la nostre prototip

Els dos components principals són els descrits anteriorment, l'MSP430F5438A i el MAX3421E.

LM2594M-5.0 és un regulador de voltatge de Texas Instruments d'alta eficiència que ens converteix una entrada de fins a 60V a 5V amb una intensitat assegurada de 0.5A.

LM1117M-3.3 és un regulador de voltatge de Texas Instruments que ens converteix una entrada de 4.75V fins a 5.25V a 3.3V amb una intensitat de 800mA. Aquests 3.3V ens serviran per l'alimentació del MSP430 i del MAX3421E entre altres.

Els microcontroladors necessiten un dispositiu que marqui la freqüència a la que executem les instruccions (sincronisme), aquest component s'anomena rellotge oscil·lador o cristall, estan fets de cristall de quars, al nostre hardware fem servir un cristall de 32kHz, el microcontrolador generar freqüències superiors a partir d'aquesta.
Tenim també un altre cristall pel integrat MAX3421E, en aquest cas de 12MHz.

6. RECURSOS DEL MICROCONTROLADOR

El microcontrolador MSP430F5438A té diverses funcionalitats. Les més interessants per a nosaltres, i són les que mapejarem des d'Android, són:

-Entrades i sortides digitals

-Entrades analògiques (ADC)

-Sortida PWM

-Comunicació sèrie asíncrona (UART)

-Comunicació sèrie síncrona tipus I^2C

-Comunicació sèrie síncrona tipus SPI

A continuació entrarem en detalls de cadascun d'aquests recursos:

6.1 Convertidor analògic a digital (ADC)

El convertidor analògic a digital també conegut com ADC de les sigles angleses de *Analog-to-Digital Converter*, és un dispositiu electrònic capaç de convertir un senyal analògic en un valor binari, en altres paraules, s'encarrega de transformar senyals analògiques a digitals (0's i 1's).

El dispositiu estableix una relació entre la seva entrada (senyal analògica) i la seva sortida (digital) segons la seva resolució. La resolució determina la precisió amb la que es reprodueix el senyal original.

La formula que ens diu la resolució és la següent:

$$Resolució = \frac{Vref}{2^n}$$

Vref = voltatge màxim que pot llegir el ADC

n = nombre de bits de resolució que té el ADC

Suposem que tenim un ADC que pot llegir de 0 a 5v amb una resolució de 3 bits:

$$Resolució = \frac{5}{2^3} = \frac{5}{8} = 0.625\text{v}$$

És a dir, aquest ADC serà capaç d'identificar un voltatge analògic diferent en rangs de 0.625v

Vin (Analògic)	Vout (Digital)
4.375 a 5	111
3.75 a 4.375	110
3.125 a 3.75	101
2.5 a 3.125	100
1.875 a 2.5	011
1.25 a 1.875	010
.625 a 1.25	001
0 a .625	000

Figura 13: Taula de correspondències

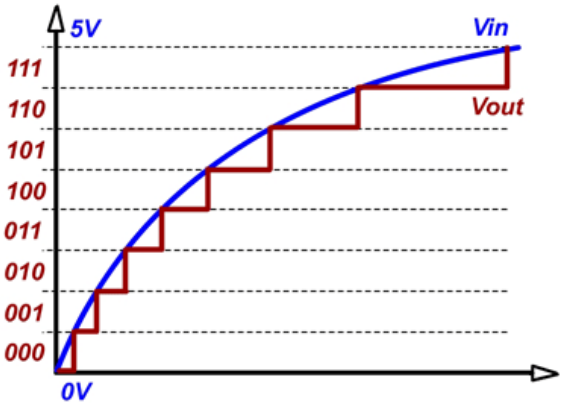


Figura 14: Gràfic Vin/Vout respecte temps

El nostre microcontrolador disposa de 14 entrades analògiques amb una resolució de 12bits, i la seva tensió de referència és de 2.5V. Això ens dona una resolució de:

$6.1 \cdot 10^{-4}$ V/LSB

(LSB és el bit menys significatiu que marca la mínima tensió que podem discriminar)

6.2 Modulació per amplada de pols (PWM)

La modulació per amplada de pols, molt coneguda per les sigles PWM de "Pulse Width Modulation", és una tècnica emprada als microcontroladors per aconseguir una tensió analògica en un món digital sense haver de fer ús de mòduls DAC (Digital-to-Analog Converter). Aquests mòduls DAC tenen una resolució major que la que s'aconsegueix mitjançant el PWM però no es solen incloure als microcontroladors que estem utilitzant per raó de costos. En canvi els DAC són àmpliament utilitzats en àmbits com generació de senyals d'àudio i vídeo.

El PWM consisteix en la generació d'un senyal quadrat de període fixe que commuta entre un valor mínim i una valor màxim, en molts casos 0 i 5V, aquesta senyal passarà per un filtre RC passa baixos, que destrueix les freqüències altes, quedant-se amb el senyal mig.

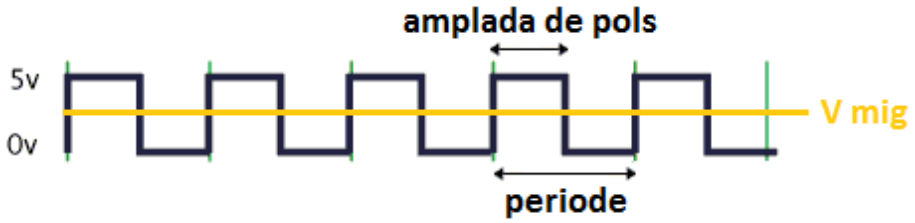


Figura 15: Senyal mig (groc) generat pel senyal quadrat (blau)

D'aquesta forma variant la porció de temps que mantenim el pols de 5V actiu (duty cycle), tindrem una tensió constant de sortida o una altra, és a dir, som capaços de generar tensió en tot el rang de 0 a 5V.

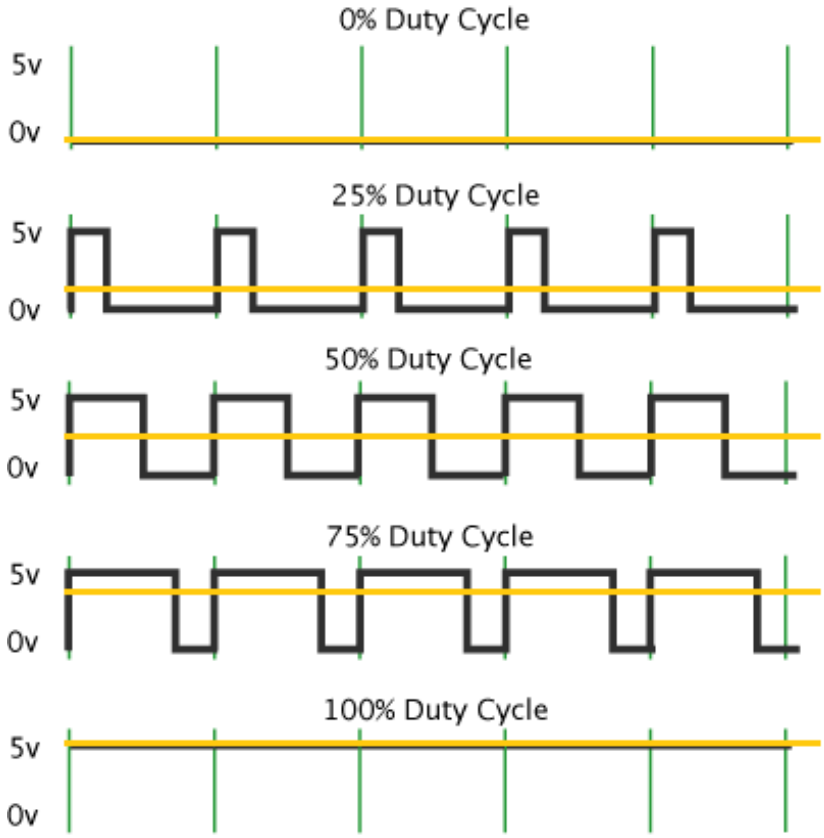


Figura 16: Representació de diferents duty cycle

En molts casos el propi dispositiu al qual apliquem el senyal PWM fa de "filtre". Per exemple, imaginem un motor de corrent continu, la seva velocitat depèn de la tensió amb que alimentem. Si en comptes de canviar la tensió d'alimentació el que fem es aplicar un senyal PWM amb que va des de la tensió mínima (L) a la màxima (H), l'efecte seria que quan el senyal està en "L" el motor "estaria" parat i si el senyal està en "H" el motor va a tota velocitat. Ara bé, si els canvis són suficientment ràpids al temps no li dóna temps ni de parar ni de agafar la màxima velocitat. L'efecte serà que el motor anirà a una velocitat de terminada únicament per la relació (temps en "H")/(temps en "L") (duty cycle), es a dir el mateix efecte que canviar la tensió analògica d'alimentació.

Aquest mateix principi es fa servir en multitud de dispositius quotidians, per exemple per il·luminació amb LEDs.

6.3 Comunicacions Sèrie

6.3.a UART

La comunicació sèrie asíncrona també és coneguda amb el nom d'UART (Universal Asyncronous Receive Transmit) o RS232.

Aquesta comunicació és punt a punt, no és tipus bus.

USCI és el nom del mòdul que controla les comunicacions sèrie al nostre microcontrolador, i que a més de en mode UART pot treballar en mode I²C i SPI.

RS232 és l'estàndard que es va definir als anys 1960s com a *Recommended Standard* per a comunicacions asíncrones i que encara es fa servir avui, la interfície i les seves característiques més interesants són les següents:

-Distancies curtes de fins a 15m

-Tassa de transferència de 20kbps (encara que avui dia es treballa molt més ràpid)

-Voltatges de -3 a 15V per a escriure un 0 lògic, i -3 a -15V per a escriure un 1

Les dades s'envien bit a bit un darrera del altre per una mateixa línia en paquets de Bytes amb un START BIT precedint-lo i amb un bit de paritat (opcional) i un o dos bits d'STOP.

Com que és una transferència asíncrona, no s'envia senyal de rellotge i es necessiten rellotges a l'emissor i al receptor de la mateixa freqüència.

En el nostre cas, quan treballem amb la transferència asíncrona només fem servir dos pins, un pin de recepció de dades (RX) i un pin de transferència de dades (TX).

Figura 17: Diagrama de comunicació serial

Com a comunicacions Síncrones (on el mestre envia un senyal de rellotge als esclaus) tenim dues opcions: I²C i SPI:

6.3.b I²C

L' I²C és un bus de comunicacions sèrie nascut als 1980s de la mà de Philips.

El seu nom prové de *Inter-Integrated Circuit* (Inter-Circuits Integrats), i serveix per comunicar diferents dispositius (circuits integrats entre si) on tots estan connectats a un mateix bus.

La característica principal és que només necessita 2 línies, una per a les dades (SDA) i una altra per a la senyal de rellotge (SCL). Per les característiques físiques dels controladors que fan les comunicacions aquestes línies porten una resistència de *"pull-up"* a Vdd. Això és un punt important ja que els dispositius que es comuniquen mitjançant aquest bus hauran de tenir la mateixa Vdd (en cas contrari s'han d'afegir adaptadors de tensió).

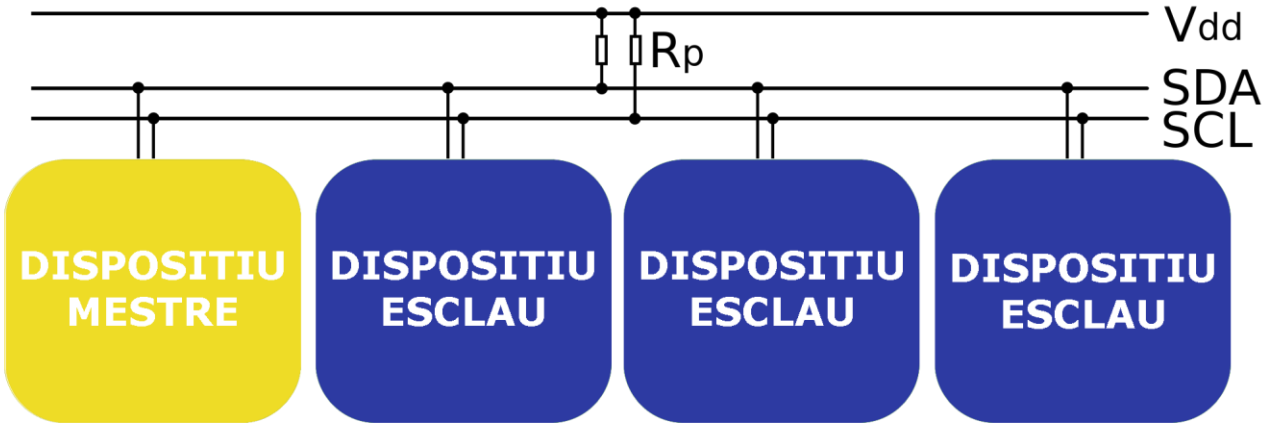


Figura 18: Diagrama de comunicació I²C

La velocitat estàndard del I²C *és de 100kbit/s, encara que poden trobar modes de baixa velocitat a 10kbit/s o d'alta velocitat a uns 3.4Mbit/s.*

Els dispositius connectats al bus tenen una adreça única per a cada un.

Els dispositius poden ser de dos tipus: Mestre o Esclau, el mestre és qui inicia la transferència de dades i proporciona el senyal de rellotge, no és necessari que el dispositiu mestre sigui sempre el mateix, aquesta característica se la poden anar passant els dispositius que tinguin aquesta capacitat (multimestre).

Un dels grans avantatges de tenir la possibilitat de fer servir aquest bus I²C al nostre projecte és que hi ha multitud de dispositius (i cada dia incrementa el número) al mercat que el fan servir. Avui en dia podem trobar des de sensors de temperatura, magnètics, acceleròmetres, humitat... de baix cost que l'implementen i que per tant amb una connexió molt molt senzilla els podem fer servir.

6.3.c SPI

L'SPI és un altre bus de comunicacions sèrie síncron, el seu nom és l'acrònim anglès de Serial Peripheral Interface. Al igual que el I²C , hi ha molts xips que el fan servir.

Els dispositius connectats a aquest bus també tenen dos possibles modes: mestre i esclau, el mestre és qui comença la transmissió de dades i selecciona quin dispositiu ha de rebre-les.

Necessita 4 línies;

SCLK *(Clock): rellotge de sincronització*

MOSI *(Master Output Slave Input)*: sortida de dades del mestre cap als esclaus

MISO *(Master Input Slave Output)*: sortida de dades del esclau cap al mestre

SS/Select: per a seleccionar un esclau

El fet d'utilitzar la línia SS complica el fet de que pugui ser multimestre encara que és possible.

És mes ràpid que l' I²C però usa més línies, el mestre ha de tenir una línia SS específica per cada esclau.

Utilitzar 2 línies de dades (recepció i transmissió) comporta que l'ús d'aquestes poguí ser simultani (Full-Dúplex) mentre que l' I²C és Semi-Dúplex.

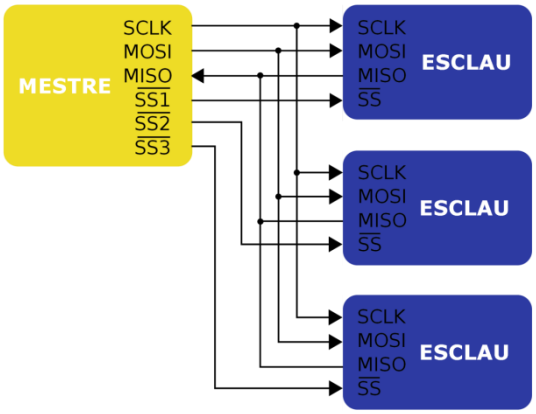


Figura 18: Diagrama de comunicació SPI

7. SOFTWARE

7.1 Costat MSP430 (Firmware)

Per programar el microcontrolador MSP430 farem servir l'IDE oficial que ens ofereix Texas Instruments anomenat Code Composer Studio. Aquest IDE agafa la base del famós Eclipse afegint-hi modificacions per a la reprogramació dels seus microcontroladors.

Una de les modificacions importants és la incorporació de tota la connexió amb la interfície de programació-debug USB. Es tracta d'un aparell que ens permet reprogramar i depurar el nostre codi al microcontrolador a través del port USB del nostre ordinador per un costat i per l'altre connectat mitjançant el connector JTAG a la nostra placa de desenvolupament.

També aporten unes llibreries on disposem de tots el recursos en forma de registres de configuració i de control.

Tot el codi es programa en C.

La versió utilitzada per aquest projecte és l'última fins al moment, 5.3.0, amb llicencia gratuïta limitada a 16KB de codi, que per al moment, ha sigut suficient.

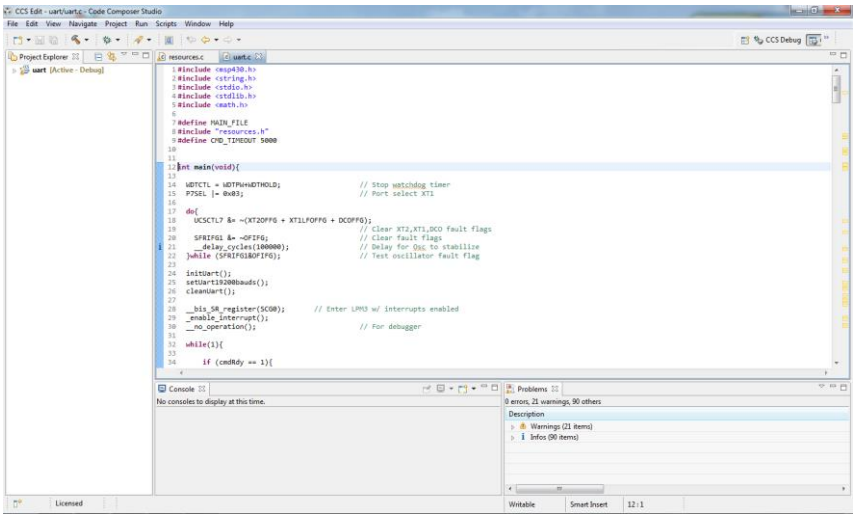


Figura 19: Captura de pantalla de Code Composer Studio 5.3.0

Estructura del codi

El codi consta de 3 fitxers:

main.c - Recepció d'instruccions des de Android, transmissió de resultats i maquina d'estats

resources.h - Funcions de configuració i interacció de tots els recursos

resources.c – Capçaleres del fitxer anterior

La màquina d'estats principal (main.c) té la següent estructura:

```
if (hem rebut una instrucció){
      if (connexió nova){
            tots els valors als inicials;
            enviarACK;
      }
      else if (instrucció de configuració de recurs){
            if (configurar sortida digital) {...}
            else if (configurar entrada digital) {...}
            else if (configurar entrada analògica) {...}
            ...
            enviarACK;
      }
      else if (instrucció d'interacció){
            if (sortida digital) {...}
            else if (entrada digital) {...}
            else if (entrada analògica) {...}
            ...
            enviarACK;
      }
      else {
            //instrucció no reconeguda
            enviarACK;
      }
}
else {
      timeOut++;
      if (portem massa estona esperant la resta d'una instrucció a mitges?){
            eliminem la instrucció que es troba a mitges;
            timeOut=0;
      }
}
```

7.2 Costat Android

A la part Android utilitzarem bàsicament l'SDK d'Android que ens proveeix d'una API amb la qual podrem realitzar la connexió Bluetooth.

Bàsicament utilitzarem la classe Bluetooth per realitzar la comunicació, més concretament els següents paquets:

android.bluetooth.BluetoothAdapter

android.bluetooth.BluetoothDevice

android.bluetooth.BluetoothSocket

Amb això, podem trobar el hardware Bluetooth del nostre dispositiu, també ens permet buscar els dispositius Bluetooth del nostre entorn i generar una connexió Bluetooth 2.0 amb ells.

Una vegada disposem d'un enllaç, tindrem dos buffers, un d'entrada de dades i un de sortida.

El perfil emprat en la nostra connexió es el SPP (*Serial Port Protocol*), una de les seves característiques es que proveeix de 2 nivells de CRC, un a la banda base i un altre a nivell L2CAP, ambdós nivells tenen mecanismes de retransmissió si es detecten corrupcions[4].

Això vol dir que no ens hem de preocupar per dades corruptes, perquè se'ns garanteix que les dades que rebrem són bones.

Estructura del codi

Board.java - Classe que hem d'instanciar a la nostra aplicació, conté totes les funcions de configuració i interacció amb la nostra placa

Classes internes

BluetoothComm.java – Conté la comunicació Bluetooth

AnalogInput.java

DigitalInput.java

DigitalOutput.java

PWM.java

OfflineTask.java

I2C.java

SPI.java

Serial.java

TimeoutException.java

NoBluetoothSupported.java

BluetoothDisabled.java

En aquesta part (Android) el codi no té gaire complexitat, la part més important és la comunicació Bluetooth així que passarem a comentar els mètodes d'enviar i rebre missatges.

Mètode enviar:

```
public synchronized void send(String data){
	try {
		BTout.write(data.getBytes());
	} catch (IOException e) {
		Log.e("ERROR", "Problem sending output");
	}
}
```

Per a enviar un string bàsicament el que fem és passar a bytes aquest string i enviar-ho fent us de la funció write(Bytes) que ens proveeix la classe OutputStream.

Parlem ara de la funció de rebre:

```
public synchronized String receive() throws TimeoutException{
	byte[] buffer = new byte[10];
	int bytesRead;
	String message="";
	long timeoutTime = System.currentTimeMillis()+TIMEOUT;

	while (!message.contains("/")){
		try {
			if (BTin.available() > 0){
				bytesRead = BTin.read(buffer);
				message = message + new String(buffer).substring(0, bytesRead);
			}
			else{
				if (System.currentTimeMillis() > timeoutTime){
					Log.e("ERROR", "Timeout reading input");
					throw new TimeoutException();
				}
			}
		} catch (IOException e) {
			Log.e("ERROR", "Problem reading input");
			throw new TimeoutException();
		}
	}
	return message;
}
```

Per a rebre strings el que fem és cridar a la funció read() de la classe InputStream fins que trobem el caràcter '/', que és el nostre identificador de fi de missatge del nostre protocol, hem d'utilitzar aquesta condició perquè no sabem quants bytes estaran disponibles quan s'executi la funció read() per tant és possible que no llegim la resposta sencera a la primera crida d'aquesta.
Si passem massa temps llegint, saltarà una excepció de timeout.

Ambdues funcions són *synchronized*, això vol dir que només es podrà executar una instancia d'aquesta funció al mateix temps, això és necessari ja que podríem tenir diversos threads a la nostra aplicació enviant i això provocaria que la placa rebria un missatge barrejat que no entendria, més o menys passaria el mateix amb la funció de rebre. Per tant és bona idea declarar aquestes funcions com a secció critica a l'hora de tenir múltiples threads mitjançant la paraula clau *synchronized*.

Per tal de que el codi sigui més robust s'han inclòs enumerats (enum) en la declaració de les funcions. Els enumerats ens permeten restringir el contingut d'una variable a una sèrie de valors predefinits de tal forma que evitem errors als codi.
Per exemple, s'han usat enumerats per indicar el pin de la placa que volem activar, d'aquesta manera podem filtrar quins pins es poden usar per determinades funcions i quins no, de manera que l'usuari no té marge per fallar.

public DigitalOutput(Board board, DigitalOutput.Pin pin)

El codi s'ha intentat empaquetar en un arxiu .jar, aquesta extensió és la més comú per distribuir llibreries Java, el problema és que encara no es poden fer .jar que continguin recursos natius d'Android[5], per lo tant la llibreria s'entrega en format llibreria-projecte d'Eclipse[6].

5. PROTOCOL DE COMUNICACIÓ

Android farà sempre de mestre i el msp430 farà de esclau. Així doncs qui iniciarà la transmissió sempre serà Android.

Totes les instruccions que enviem des de el dispositiu Android cap al msp430 tenen dues característiques bàsiques:

-Totes les instruccions acaben amb el caràcter '/', d'aquesta manera sabem quan acaba una instrucció i hem de passar a executar-la.

-Totes les instruccions necessiten un ACK*, en algunes serà el valor demanat seguit del caràcter '/' o només el propi caràcter '/', d'aquesta manera ens sincronitzem amb la velocitat del msp430.

El format dels missatges és bàsicament una cadena de *chars*, això es podria canviar per tal de que ocupessin menys bytes i que el seu significat fos el mateix però crec que seria una millora molt petita en la velocitat de transmissió i en canvi la interpretació de les comunicacions seria molt més complicada.

*ACK prové del angles acknowledgement, i es tracta d'un missatge que el destí de la comunicació (microcontrolador) envia al origen (Android) per a confirmar la recepció d'un missatge.

Llistat de missatges:

8.1 Connexió nova

Indiquem al msp430 que ens acabem de connectar

N	/

8.2 Entrades i sortides digitals

Configuració

Configuració del pin pp com a sortida digital

			port	pin	
C	D	O	p	p	/

Configuració del pin pp com a entrada digital

			port	pin	
C	D	I	p	p	/

Escriptura

Establir en mode alt la sortida digital del pin pp

		port	pin		
D	O	p	p	H	/

Establir en mode baix la sortida digital del pin pp

		port	pin		
D	O	p	p	L	/

Lectura

Llegir el valor de l'entrada digital del pin pp

		port	pin	
D	I	p	p	/

Retorna

0	/
1	/

8.3 Entrades analògiques

Configuració

Configuració del pin pp com a entrada analògica

			port	pin	
C	A	I	p	p	/

Lectura

Llegir el valor de l'entrada analògica del pin pp

		port	pin	
A	I	p	p	/

Retorna

x	x	x	x	/
	x	x	x	/
		x	x	/
			x	/

8.4 Modulació per amplada de pols (PWM)

Configuració

Configuració del pin pp com sortida PWM amb "lp" unitats de període i "ld" de duty cycle

				port	pin	long. periode	long. duty cycle	periode	duty cycle	
C	P	W	M	p	p	lp	ld	per	d	/

Escriptura

Establir un nou valor de duty cycle al pin pp

			port	pin		nou duty	
P	W	M	p	p	D	xxx	/

Establir un nou valor de període al pin pp

			port	pin		nou periode	
P	W	M	p	p	P	xxx	/

8.5 Tasques offline (Offline tasks)

Configuració

Configuració del pin pp com a tasca offline, entrada digital recollint yyyy mostres cada xxxxx min

					port	pin	unitat	període					número mostres				
C	O	T	D	I	p	p	M	x	x	x	x	x	y	y	y	y	/

Configuració del pin pp com a tasca offline, entrada digital recollint yyyy mostres cada xxxxx ms

					port	pin	unitat	període					número mostres				
C	O	T	D	I	p	p	U	x	x	x	x	x	y	y	y	y	/

Configuració del pin pp com a tasca offline, entrada digital recollint yyyy mostres cada xxxxx seg

					port	pin	unitat	període					número mostres				
C	O	T	D	I	p	p	S	x	x	x	x	x	y	y	y	y	/

Configuració del pin pp com a tasca offline, entrada analògica recollint yyyy mostres cada xxxxx min

					port	pin	unitat	període					número mostres				
C	O	T	A	I	p	p	M	x	x	x	x	x	y	y	y	y	/

Configuració del pin pp com a tasca offline, entrada analògica recollint yyyy mostres cada xxxxx ms

					port	pin	unitat	període					número mostres				
C	O	T	A	I	p	p	U	x	x	x	x	x	y	y	y	y	/

Configuració del pin pp com a tasca offline, entrada analògica recollint yyyy mostres cada xxxxx segons

					port	pin	unitat	període					número mostres				
C	O	T	A	I	p	p	S	x	x	x	x	x	y	y	y	y	/

Lectura

Llegir els valors emmagatzemats fins ara

O	T	/

Retorna (números separats per punts)

xxx	.	xx	.	xxxx	.	x	.	xx	.	/

8.6 Sèrie Asíncron (UART)

Configuració

Configuració del pin 10.4(tx) i 10.5(rx) com a port serial de comunicacions

C	U	A	R	T	A	3	/

Escriptura

Transmetre un missatge al port serial

							missatge	
U	A	R	T	A	3	T	xxxxxxxxx	/

Lectura

Rebre un missatge per el port serial

U	A	R	T	A	3	R	/

Retorna:

missatge	
xxxxxxxxx	/

8.7 I²C

Configuració

Configuració del pin 3.1(sda) i 3.2(scl) com a port I²C en mode mestre per parlar amb l'esclau amb adreça 0xhh

						mode	adreça hex	adreça hex	
C	I	2	C	B	0	M	h	h	/

Escriptura

Transmetre bytes en format hexadecimal

						data	
I	2	C	B	0	T	xx	/

Lectura

Rebre x bytes en format int (separats per punts)

						num. Bytes	
I	2	C	B	0	R	x	/

Retorna

missatge	
x.x.xx.xxx.x.x.	/

8.8 SPI

Configuració

Configuració del pin 10.3(CLK), 10.1(SIMO), 10.2(SOMI) i 10.0(STE) com a port SPI en mode mestre

C	S	P	I	B	3	M	/

Configuració del pin 10.3(CLK), 10.1(SIMO), 10.2(SOMI) i 10.0(STE) com a port SPI en mode esclau

C	S	P	I	B	3	S	/

Lectura

Rebre un missatge per el port serial

C	S	P	I	B	3	R	/

Retorna:

missatge	
xxxxxxxxx	/

Escriptura

Transmetre un missatge al port SPI

						missatge	
S	P	I	B	3	T	xxxxxxxxx	/

9. CRONOGRAMA

Aquí podem veure un exemple d'intercanvi de missatges per a la configuració i interacció d'una sortida digital al port 1 pin 0

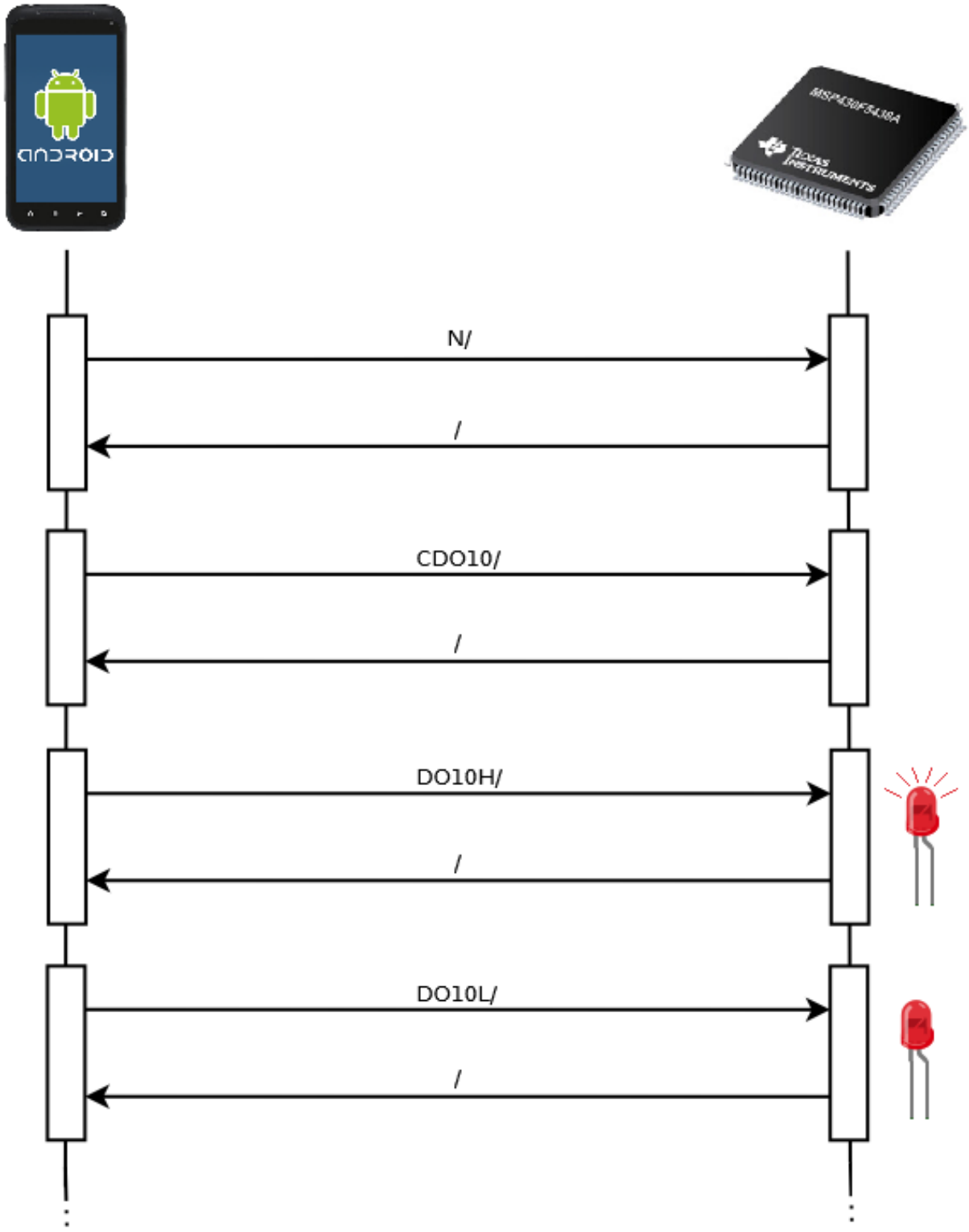


Figura 20: Diagrama de comunicació per a una sortida digital

Aquí podem veure un exemple d'intercanvi de missatges per a la configuració i interacció d'una entrada analògica al port 6 pin 7

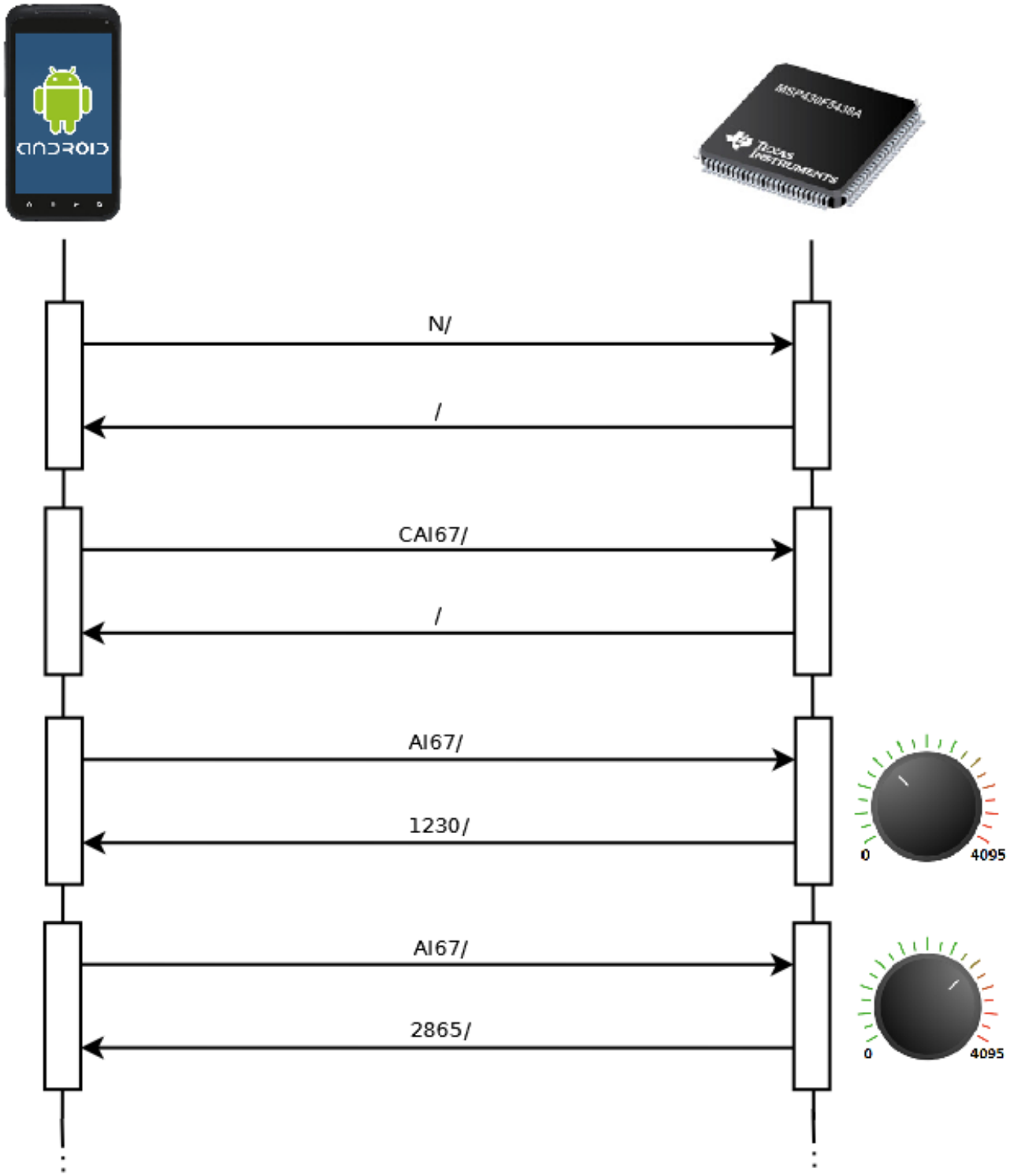


Figura 21: Diagrama de comunicació per a una entrada analògica

10. PRESSUPOST

10.1 Hardware

Dispositiu	Distribuïdor	Preu
Bluetooth HC-06	dealextreme.com	6,27 €
LM2594M-5.0	Farnell	1.90€
LM1117S-3.3	Farnell	0.35€
MAX3421EEHJ	Farnell	17.41€
MSP430F5438A	Farnell	9.6€
PCB	2CI	20,00 €
Resta material	Farnell	~4€ a 5€
	Total	~60€

S'ha de tenir en compte que aquests preus són per comprar molt poques unitats.

En cas de producció industrial el preu baixa molt. Per exemple, la placa PCB ens a costat aquest preu perquè hem fet una tirada de 12 unitats, però si demanem 100 unitats el seu preu baixa a uns 4€**.**

El mateix passa amb el microcontrolador, el seu preu és de 4.5$ quan es compren 1000 unitats. O el Host USB el MAX3421EEHJ que costa 4.85$ al comprar 1000 unitats.

10.2 Software

En aquest apartat volem respondre quant temps ha sigut necessari per desenvolupar el projecte.

Com és difícil donar una quantitat en hores, el que farem serà revisar per sobre com s'ha dividit el temps durant el llarg del quadrimestre.

Durant el primer mes i mig el treball va estar més centrat en la investigació, es va aconseguir una IOIO per tal de provar-la i veure com la feien funcionar.

Gracies a això es va poder fer una primera aproximació de la comunicació USB, al següent enllaç del repositori GitHub es pot veure el codi que es va fer per comunicar un dispositiu Android amb un PC:

https://github.com/pablobuenaposada/androidUSBComunicationProject

Una vegada es va saber que necessitàvem un port USB Host es van buscar solucions i es va estudiar l'integrat MAX3421E.

Durant aquest període també es va treballar amb el mòdul Bluetooth, primer es van fer proves amb l'Arduino i la IOIO per veure com funcionava el mòdul, als següents links es poden veure els codis que fan d'interfície entre aquestes plataformes i el mòdul Bluetooth HC-06, Arduino: https://github.com/pablobuenaposada/arduino-HC-06 , IOIO: https://github.com/pablobuenaposada/ioio-HC-06 .

A partir de finals de Març es va començar amb la programació del microcontrolador i la de la part Android al següent repositori https://github.com/pablobuenaposada/MSP430-Android .

Aquí podem veure un gràfic dels *commits* del projecte i més o menys quina part es trobaven desenvolupant segons la data, cal remarcar que això és simplement aproximat.

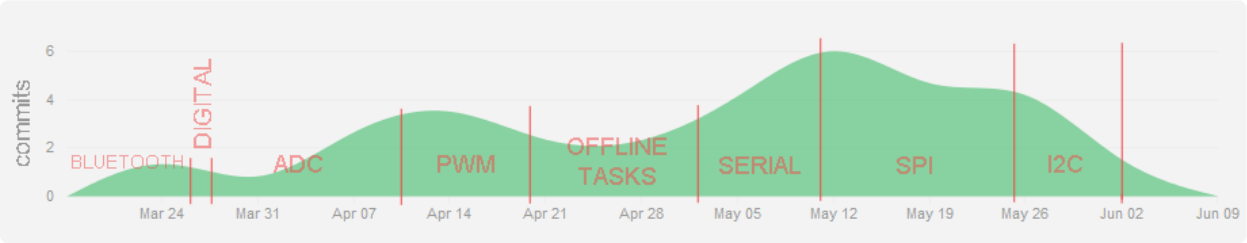


Figura 22: Gràfic de commits respecte temps

Així doncs, el temps total del codi Android i del MSP430 final a estat d'uns 2 mesos i mig. Es pot dir que en promig s'han dedicat unes 5 hores cada dia així que en total del projecte seria; 4h x 120dies = 480 hores.

11. CONCLUSIONS

En termes generals crec que és pot dir que ha sortit un bon treball encara que s'ha anat molt just de temps, principalment perquè tractant-se d'un projecte amb una part de prototip hardware et trobes condicionat a les esperes de l'arribada dels diversos components, a més dels típics problemes que surten a qualsevol projecte. Una mica més de temps s'hagués agraït ja que es podria haver fet més proves, detectar més problemes i acabar d'enllestir coses que han quedat a mitges.

Passem llavors a analitzar els resultats respecte als objectius inicials.

1º Objectiu: ***L'objectiu bàsic és desenvolupar el hardware i software per poder gestionar l'adquisició de dades i control de processos des de dispositius Android.***

Objectiu complert. En termes generals podem parlar sense cap tipus de problema que l'objectiu principal s'ha aconseguit, podem interactuar amb un microcontrolador des d'un dispositiu Android mitjançant una llibreria Java senzilla.

2º Objectiu: ***Enllaçar el microcontrolador a un dispositiu Android per USB i Bluetooth***

La comunicació per Bluetooth s'ha superat amb èxit en contraposició de la comunicació per USB que no s'ha aconseguit.

Des del primer moment sabíem que havíem de treballar amb el xip MAX3421E de Maxim per tal d'establir una connexió USB Host, el problema és que es tracta d'un xip molt petit el qual s'ha de dissenyar una placa per poder fer proves o be aconseguir la placa de desenvolupament que té Maxim sobre aquest xip que és difícil de trobar.

La dificultat de treballar amb el MAX3421E ha fet que paral·lelament s'hagi treballat més amb la connexió Bluetooth fins al punt de convertir-se en la connexió principal, el mòdul Bluetooth no és tan problemàtic en aquest aspecte i s'ha anat avançant gracies a aquest.

De totes formes, a la placa hem posat el MAX3421E per si en un futur es vol seguir continuant el projecte ja es te connectat i llest per treballar a la part software d'aquesta connexió, de fet, tant el protocol com la resta de software desenvolupat seria totalment compatible amb aquest integrat.

3º Objectiu: ***Crear un protocol de comunicació***

S'ha definit un protocol de comunicació segur, senzill i fàcil d'explicar que cobreix les necessitats del projecte. Permet modificacions sense reestructuracions complicades i seria totalment compatible amb la connexió USB si en un futur s'acaba desenvolupant.

4º Objectiu: ***Crear una llibreria Android (Java) per controlar el microcontrolador***

Era una tasca implícita en el funcionament del projecte, s'ha intentat que les funcions fossin el més lògiques possibles. S'entrega en format de projecte d'Eclipse i es pot utilitzar a partir de la versió 2.0 d'Android.

5º Objectiu: ***Programar el microcontrolador per a que entengui la llibreria anterior (Firmware)***

Ha estat la part més difícil però s'ha aconseguit que el microcontrolador entengui tot el compendi de protocols, llibreries i senyals que l'enviem des d'Android.

6º Objectiu: Dissenyar una placa que contingui el microcontrolador i tots els dispositius necessaris per poder treballar amb el projecte

Gracies a l'ajuda dels directors del projecte s'ha aconseguit aquesta part, ja que personalment es tracta d'un tema bastant allunyat del que es dóna a l'enginyeria informàtica. Encara que es podria haver obviat aquesta part, la realització d'aquesta dona el "toc" final al projecte.

7º Objectiu: ***Mapejar el màxim número de recursos del microcontrolador. És a dir, fer accessible el màxim número de recursos del microcontrolador des de les aplicacions desenvolupades al dispositiu Android.***

La comunicació SPI encara no acaba de funcionar del tot bé però amb una mica més de treball s'hauria de poder arreglar, per un altra costat, tots els altres recursos semblen funcionar be amb les proves realitzades.

8º Objectiu: ***Programació de tasques de recollida de dades sense necessitat d'estar sempre connectats, el que denominarem treballar off-line***

Es poden realitzar tasques de recol·lecció de dades d'entrades analògiques i digitals en intervals de temps a definir per l'usuari de minuts, segons o milisegons i obtenir-les quan es desitgi connectant-se momentàneament al sistema.

Es podria treballar per poder definir intervals d'unitats diferents sense gaire problema.

Crec que en conclusió el resultat final està força bé. Si mirem el repositori de la IOIO podem veure que des de el primer *commit* al 2011, el major espai de temps sense actualitzacions al seu codi es de dues setmanes, això ens dona una idea de que estem ficats en un projecte que no es pot finalitzar en un quadrimestre.

11.1 Valoracions personals

La programació del MSP430 ha estat bastant dura, es tracta d'una programació a bastant baix nivell on per a qualsevol cosa es necessita una bona lectura del *user guide* del microcontrolador i una cerca de codis d'exemple per internet. Prova i error ha estat la pedra filosofal de tota aquesta part de la programació.

Per un altre costat, l'aprenentatge de com funciona el microcontrolador és el millor que m'emporto, m'obre les portes a qualsevol altre processador ja que la manera de funcionar són molts semblants en general.

Haver treballar en un projecte que es troba entre mig de l'electrònica i la informàtica m'ha donat la possibilitat de veure moltes coses de la branca de l'electrònica que enriqueixen el meu coneixement, sempre venen bé coneixements d'altres branques i més com la electrònica que van lligades a la informàtica però que a la carrera no s'interioritzen tant.
Personalment estic molt content del resultat, primer perquè he aconseguit un treball de fi de grau sobre un tema relativament novedós i fet quasi a mida per mi, i on he pogut prendre decisions creatives, i per un altra part perquè he aconseguit el que abans de començar mai hagués pensat, aconseguir una millora respecte als competidors com IOIO o Arduino encara que sigui en un petit aspecte, com és el PWM on aquí es pot configurar lliurement el període i en canvi a l'Arduino està fixat a 2ms[7] i a IOIO es pot configurar però només a la inicialització[8], una vegada inicialitzat ja no es pot reconfigurar.

També m'aventuro a dir que hi ha una gran oportunitat cobrint el sector Apple, per el moment no hi ha cap tipus de hardware comercial (del tipus Arduino, IOIO o aquest

mateix projecte) que es pugui connectar a dispositius com l'iPhone o l'iPad per connexió USB. Sens dubte aquests dos terminals estan molt extensos, tant es pot dir que hi ha una possibilitat al mercat.

Per finalitzar vull donar les gracies a Jose Bosch Estrada i a Juan Daniel Prades García per les seves aportacions.

12. POSSIBLES MILLORES

D'entrada s'ha de dir que el projecte es troba en un estat en que el que van sortint petits "bugs" per lo tant la primera millora implícita es arreglar aquests petis errors, l'únic problema es que aquests aniran apareixent a mida que es vagi fent us del projecte.

Connexió USB

L'idea del projecte era fer us d'una connexió USB per a comunicar-se amb la placa, això no ha estat possible així que la millora més important seria continuar amb el desenvolupament de la connexió USB.

Comunicació asíncrona

Actualment el sistema està desenvolupat de tal manera que el dispositiu Android sempre inicia la comunicació, una millora seria buscar alguna aplicació on fos el microcontrolador qui iniciés una comunicació.

Un exemple podria ser la implementació de Timers, és a dir, que d'alguna manera el microcontrolador pogués avisar cada cert temps a mode d'alarma, d'aquesta manera seria indispensable que el microcontrolador inicies una comunicació, fer "polling" tota l'estona des de el dispositiu Android per veure si el Timer ha arribat al temps preestablert no tindria sentit.

Compatibilitat amb productes Apple

Una millora podria ser afegir comptabilitat amb iPhone, encara que no seria possible per nosaltres.

Apple té un programa anomenat MFi el qual permet l'accés al hardware, eines, documentació i suport tècnic per tal de crear un accessori electrònic que es pugui connectar a l'iPhone, l'iPad o l'iPod.

Apple ha d'aprovar el teu de dispositiu per assegurar-se que no interfereix amb el sistema iOS.

Es parla que es necessiten uns $20,000 per tal de poder fer front a tot el que comporta el programa MFi[9].

Compatibilitat amb versió escriptori (PC)

Com que la llibreria Android desenvolupada està escrita en Java, seria molt fàcil adaptar-la a Java d'escriptori simplement utilitzant una altra API de Bluetooth per a escriptori en comptes de l'API Bluetooth d'Android.

D'aquesta manera podríem realitzar aplicacions d'escriptori que fessin us del projecte

Modes de baix consum

El microcontrolador emprat en aquest projecte té diferents modes de consum on podem desactivar certes característiques per tal d'aconseguir un nivell més baix de consum molt interesant quan estem alimentant el sistema amb bateries.

Connexió perduda

Podria ser interesant per a certes aplicacions que el microcontrolador sàpigues que ha perdut la connexió amb el dispositiu Android i fes alguna cosa en aquesta situació.

Actualment quan es perd la comunicació per qualsevol motiu (llunyania o tancament de l'aplicació al dispositiu Android) el microcontrolador no en sap res, en el pitjor dels casos

es queda amb una instrucció a la meitat (no li arriba el final de la instrucció amb el caràcter '/') i aquesta seria esborrada de memòria per timeout i seguiria esperant a rebre una altra.

Una manera de fer això podria ser dirigir una sortida d'estat del mòdul Bluetooth cap a una entrada del microcontrolador.

No ACK

Com s'ha comentat a l'apartat de protocol de comunicació, totes les instruccions que enviem esperen un ACK (excepte les nonACK experimentals), això es degut a que el microcontrolador te un únic fil d'execució i no podem realitzar una instrucció fins que no hem acabat l'anterior, l'ACK en el nostre cas ens serveix per això, per saber que la instrucció s'ha realitzat i el microcontrolador es troba lliure.

Una millora en aquest aspecte podria ser fer un buffer d'instruccions i anar-les fent el més ràpid possible, de manera que des de Android no caldria esperar cap ACK.

Missatges més reduïts

Com s'ha comentat al capítol de protocol de comunicació, es podria buscar alguna altra sintaxis que ocupes menys bytes per tal de millorar la velocitat de transmissió, encara que no sigui un gran augment, segurament no faria mal que els missatges fossin més curts.

13. BIBLIOGRAFIA

User Guide: http://www.ti.com/lit/ug/slau208m/slau208m.pdf

Codi exemples del MSP430 de Texas Instruments (disponibles a CCS 5.3.0)

Fòrum sobre el MSP430 www.43oh.com

Informació general

http://www.diyphonegadgets.com/p/solutions.html

http://deeea.urv.cat/public/PROPOSTES/pub/pdf/814pub.pdf

http://david.cuartielles.com/b/2011/05/google-adk-que-bicho-es-ese/

http://en.wikipedia.org/wiki/Firmware

Teoria

http://meteoyelectronica.blogspot.com.es/2010/12/pwm-una-manera-sencilla-de-controlar-un.html

http://www.embebidos-cidetec.com.mx/profesores/jcrls/doctos/aplic_cidetec.pdf

http://es.wikipedia.org/wiki/Baudio

http://es.wikipedia.org/wiki/RS-232

http://es.wikipedia.org/wiki/I%C2%B2C

http://es.wikipedia.org/wiki/Serial_Peripheral_Interface

http://www.ate.uniovi.es/fernando/Doc2003/SED/SCI_asincrono.pdf

http://www.gammon.com.au/forum/?id=10892

http://www.cursomicros.com/avr/conversor-adc/adc-de-aproximaciones-sucesivas.html

Android

http://qtcstation.com/2011/03/connecting-android-to-the-pc-over-usb/

http://www.anothem.net/archives/2010/10/15/android-usb-connection-to-pc/

https://github.com/user370305/Android-USB-Communication

http://android-developers.blogspot.com.es/2009/05/painless-threading.html

http://developer.android.com/tools/help/adb.html

MSP430

http://homepages.ius.edu/RWISMAN/C335/html/Syllabus.htm (curs de la universitat d'Indiana molt ben explicat)

http://www.referencedesigner.com/tutorials/msp430/msp430_30.php

http://labarqui.wordpress.com/author/nvdanoun/

https://github.com/glitovsky38412/msp430uartdriver

http://markoez.pirategames.co.uk/index.php/projects/blutooth-front-door-lock-using-msp430/

http://blog.ilric.org/2011/09/10/msp430-ti-launchpad-calculator/

http://dbindner.freeshell.org/msp430/

http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/110940.aspx

http://myweb.wit.edu/johnsont/Classes/462/ADC%20for%202%20sensors.htm

http://microembebidos.wordpress.com/2012/12/06/tutorial-msp430-adc10-conversor-analogico-digital-de-10-bits/

http://www.swarthmore.edu/NatSci/echeeve1/Class/e91/Lectures/E91%286%29A2D.pdf

http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/19686.aspx

http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/118548.aspx

Mòdul Bluetooth

http://byron76.blogspot.com.es/2011/09/one-board-several-firmwares.html

http://letsmakerobots.com/node/35489

http://tallerarduino.wordpress.com/tag/hc-06/

http://bellcode.wordpress.com/2012/01/02/android-and-arduino-bluetooth-communication/

http://projectproto.blogspot.com.es/2010/09/android-bluetooth-oscilloscope.html

http://www.hobbyist.co.nz/?q=bluetooth-module-configurations

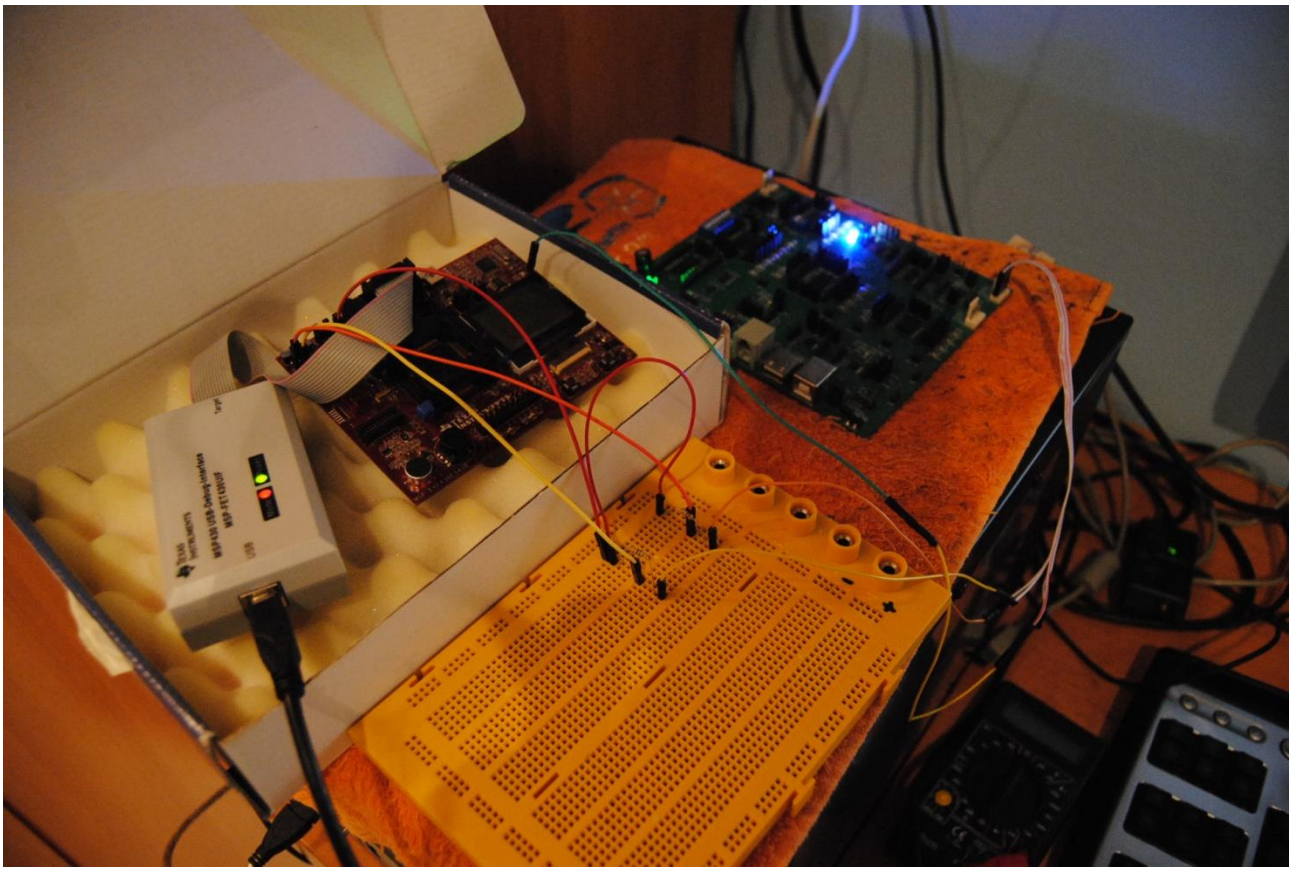
http://www.ebay.es/itm/251101892223

USB Host

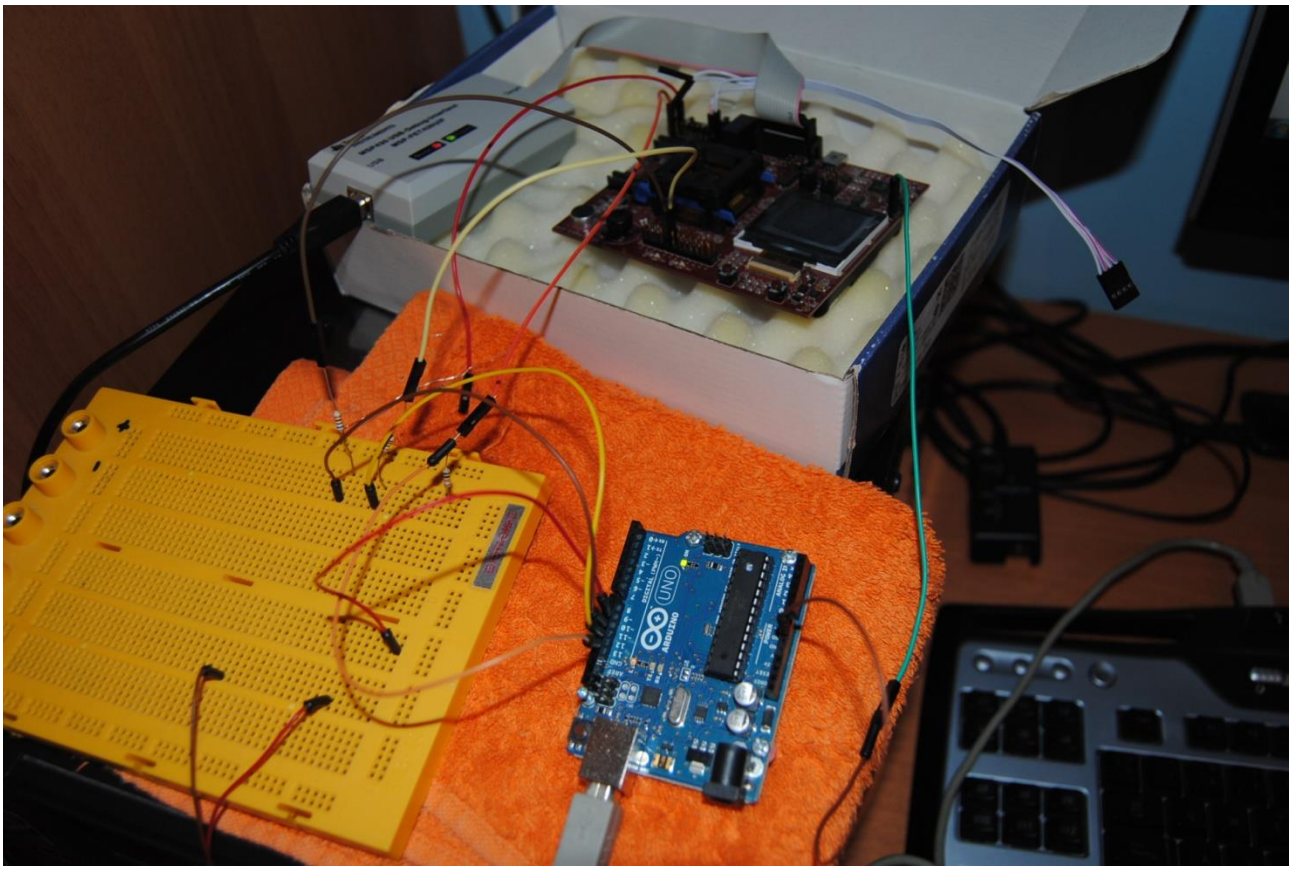
http://arduino.cc/forum/index.php?topic=78165.0

http://developer.android.com/guide/topics/connectivity/usb/index.html

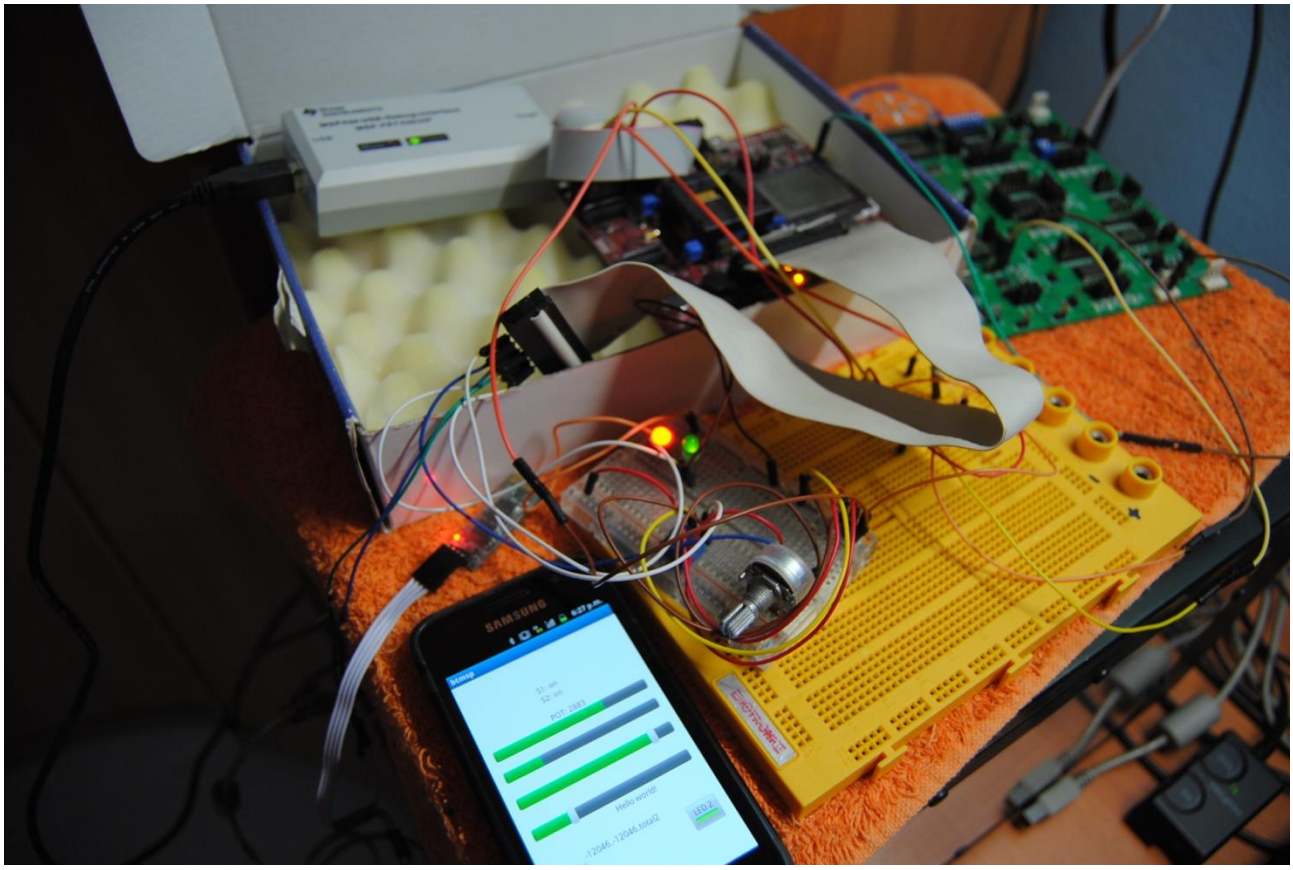
14. ANNEX



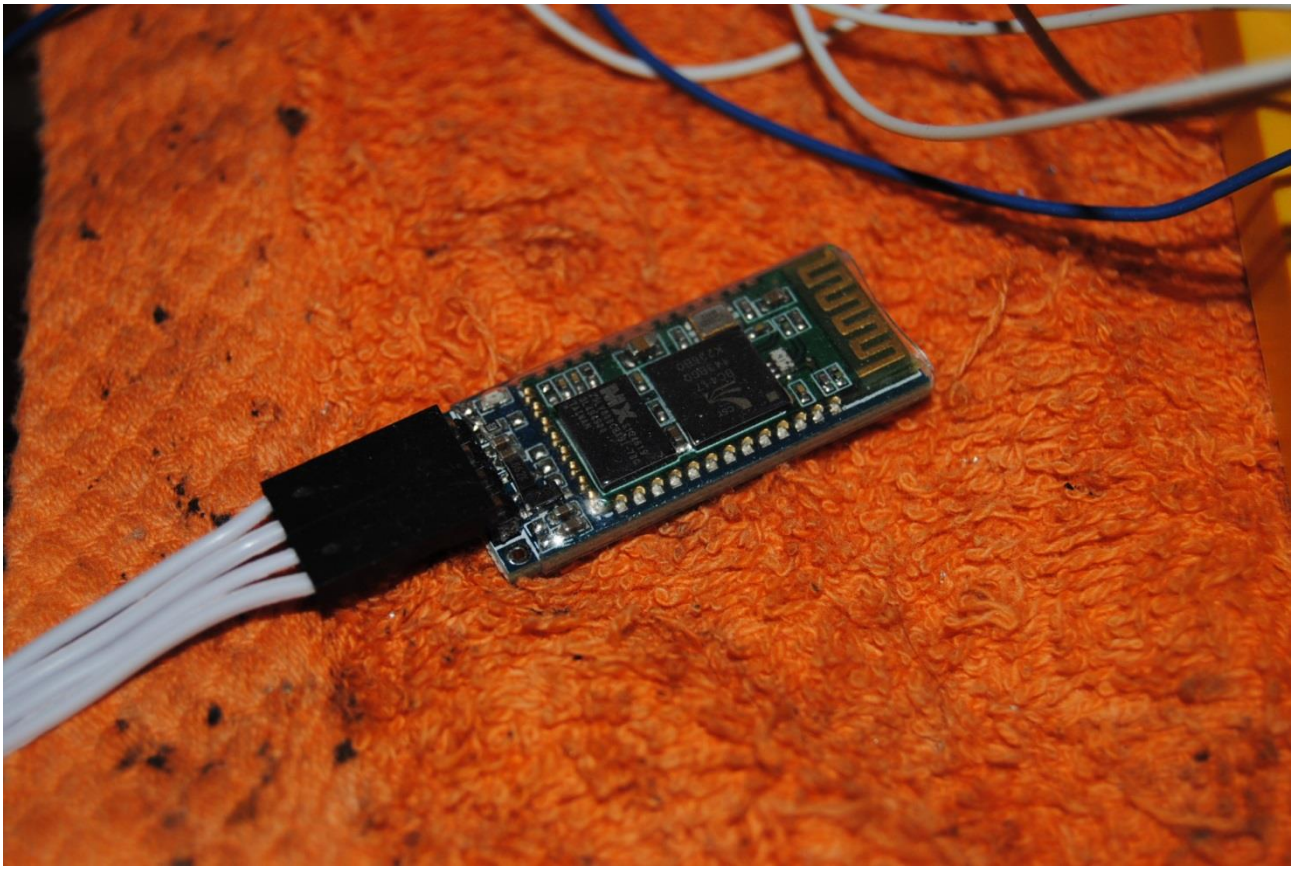
MSP430 comunicant-se amb una placa (I2C2002-1A) amb diversos mòduls I²C(leds, sensors de temperatura, ports d'expansió...)



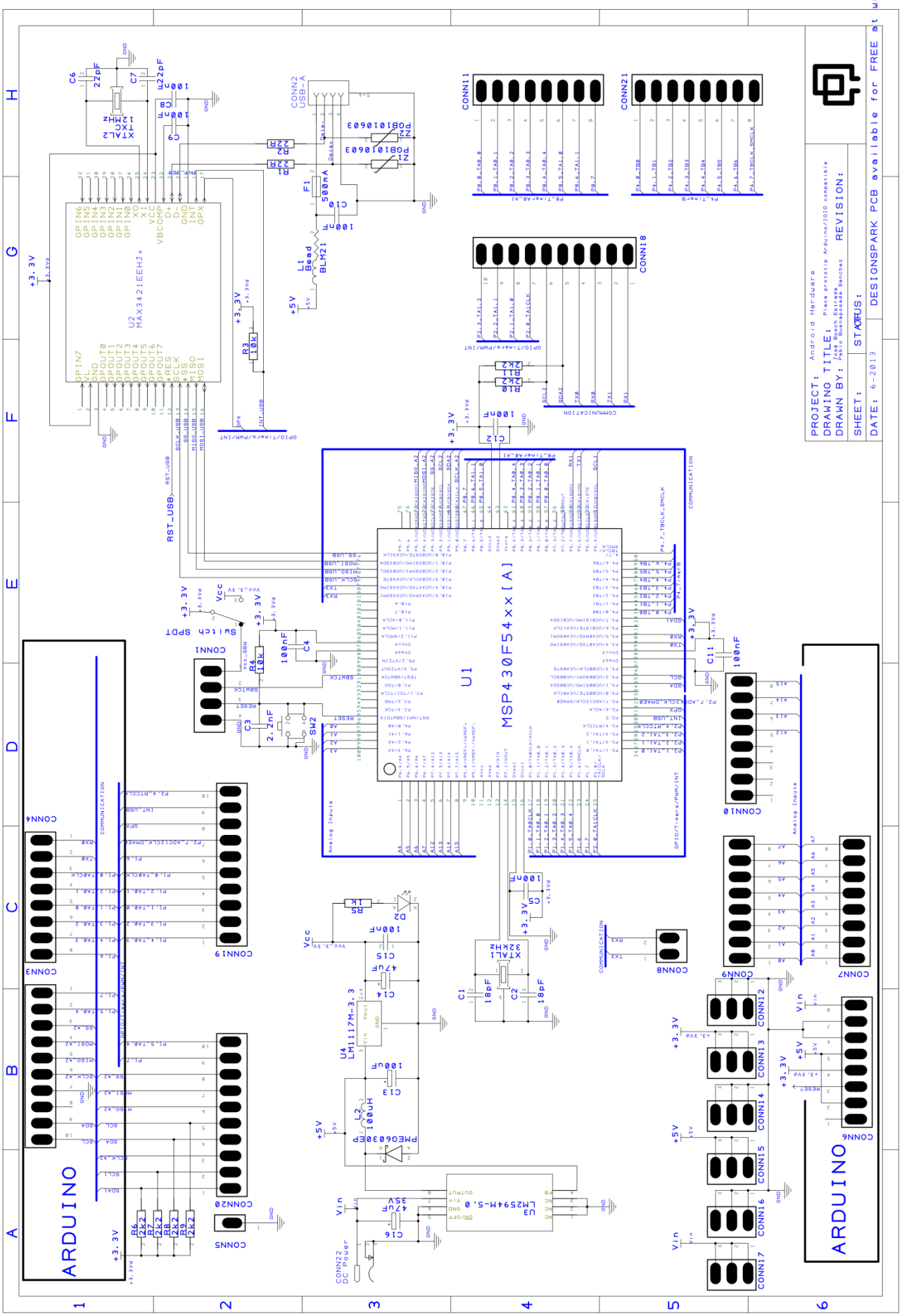
MSP430 "parlant" amb un Arduino UNO per SPI



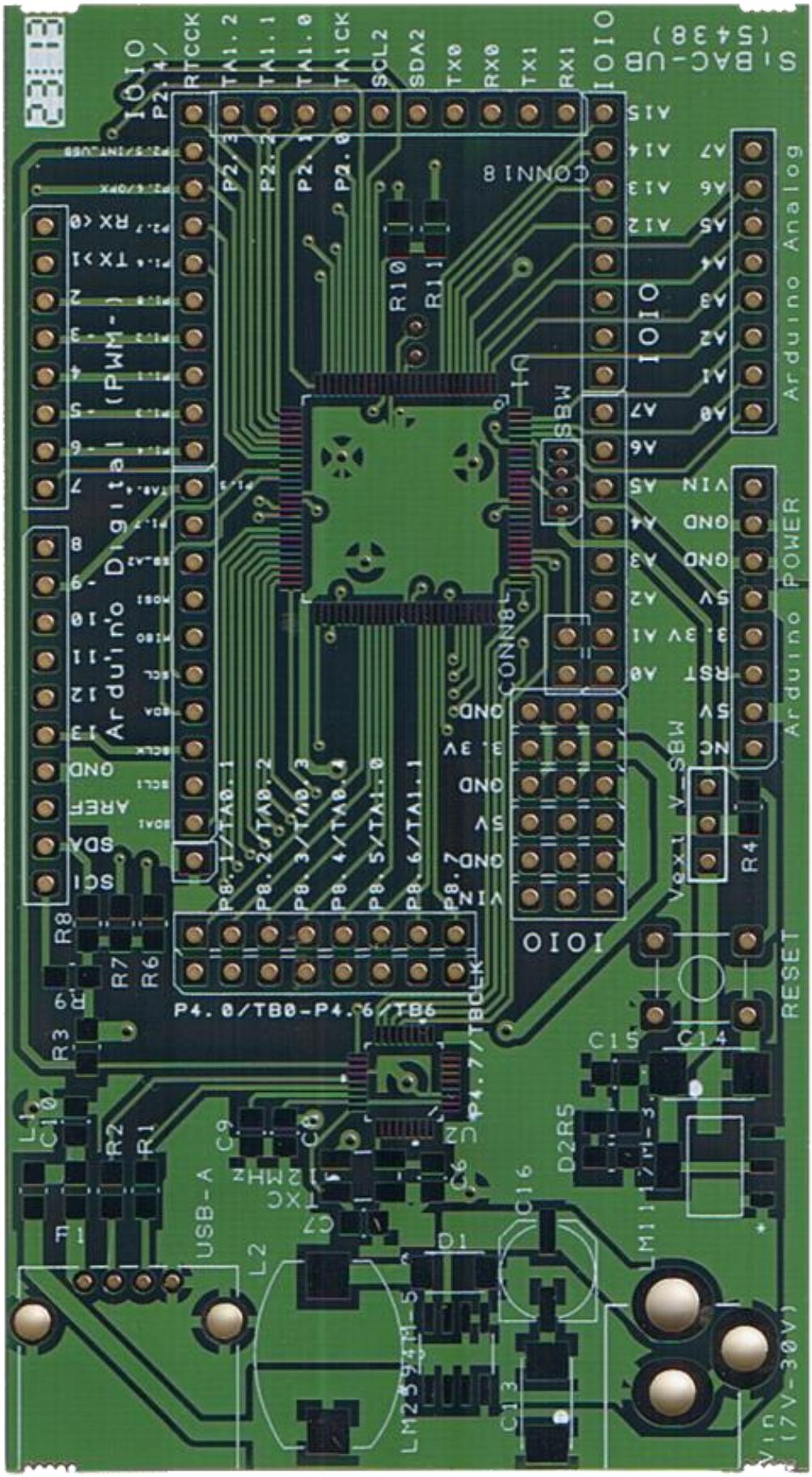
MSP430 connectat per Bluetooth amb unSamsung Galaxy i9000 monitoritzant 2 potenciòmetres, 2 polsadors, 2 leds per PWM, 2 leds per sortida digital i un sensor de temperatura per I²C



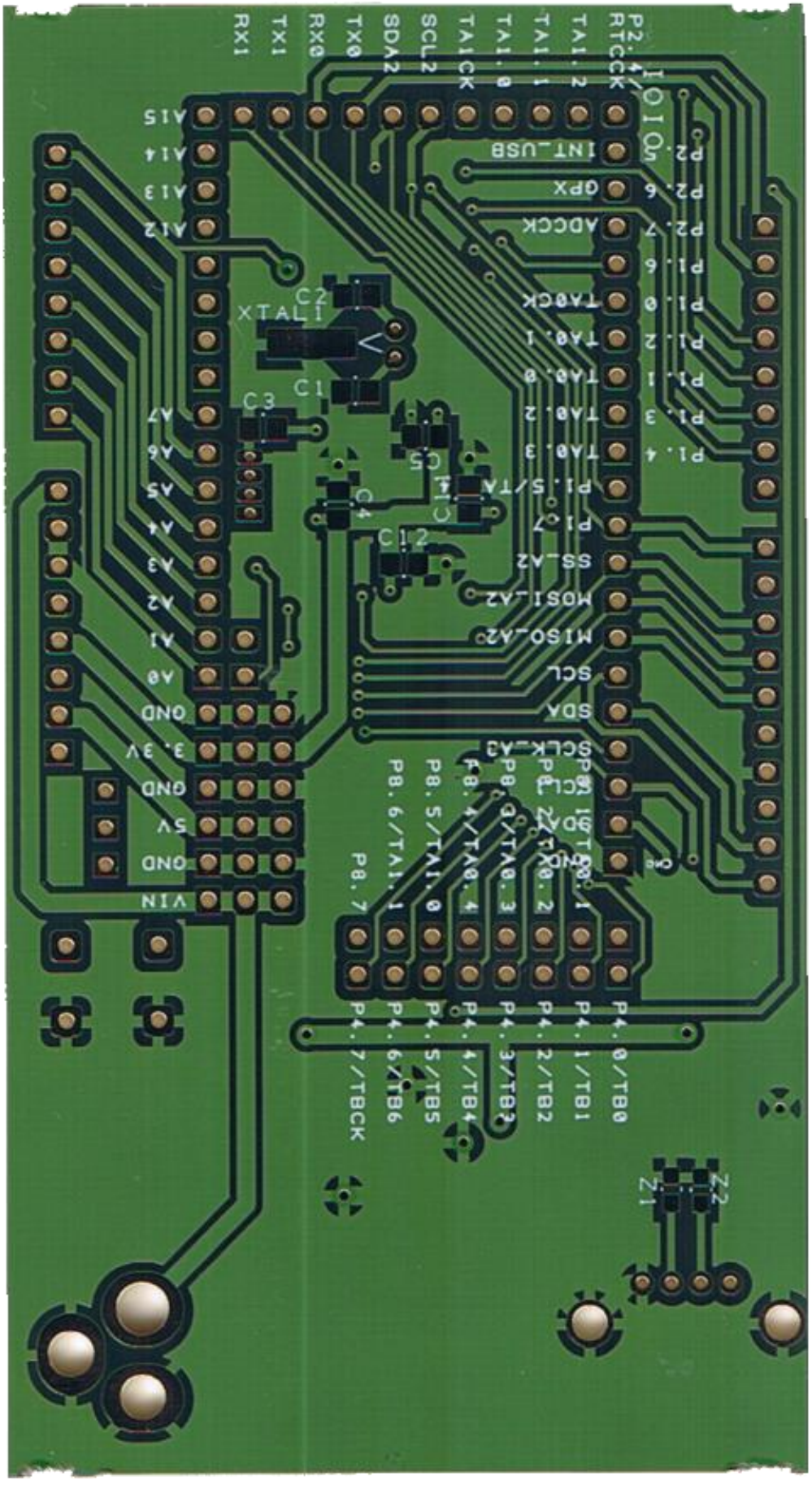
Mòdul Bluetooth HC-06



Esquema del prototip



Circuit imprès del prototip sense els components soldats (vista anvers)



Circuit imprès del prototip sense els components soldats (vista revers)

[1] http://arduino.cc/en/Main/arduinoBoardUno

[2] http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino

[3] http://www.neoteo.com/modulo-bluetooth-hc-06-android

[4] http://stackoverflow.com/questions/14498530/bluetooth-android-rfcomm-spp-error-handling-suggestions

[5] http://stackoverflow.com/questions/7973822/how-to-convert-an-android-library-project-to-an-external-jar

[6] http://www.nosinmiubuntu.com/2012/03/reutiliza-tu-codigo-android.html

[7] http://arduino.cc/es_old/AnalogRead/ADC

[8] https://github.com/ytai/ioio/wiki/PWM-Output

[9] http://apple.stackexchange.com/questions/11794/can-a-hobbyist-or-individual-apply-for-apples-mfi-program