

Projecte Fi de Carrera

Monitorització de l'ús de diferents elements de comunicació en un dispositiu Android

Alberto Burgos Plaza

Director: Jordi García Almiñana

Barcelona, juny de 2012

Dades del projecte

Títol del projecte: Monitorització de l'ús de diferents elements de comunicació en un dispositiu Android

Nom de l'estudiant: Alberto Burgos Plaza

Titulació: Enginyeria Tècnica en Informàtica de Sistemes

Crèdits: 22,5

Director: Jordi García Almiñana

Departament: AC

Membres del tribunal (*nom i signatura*)

1. *Director:* Jordi García Almiñana
2. *President:* David López Álvarez
3. *Vocal:* Juan Trias Pairo

Qualificació

Qualificació numèrica:

Qualificació descriptiva:

Data:

Índex de contingut

Dades del projecte.....	i
Membres del tribunal (nom i signatura).....	i
Qualificació.....	i
Capítol 1. Introducció.....	7
1.1 Objectius.....	7
1.2 Motivació.....	8
1.3 Planificació.....	9
1.4 Metodologia.....	11
1.5 Estructuració de la memòria.....	11
Capítol 2. Entorn del projecte i estat de l'art.....	14
2.1 Sistema GPS.....	14
2.1.1 Funcionament.....	14
2.1.2 Tipus de missatges.....	16
2.2 Geolocalització a Android.....	17
2.2.1 Utilització dels recursos.....	17
2.2.2 Alternatives.....	18
2.3 Estudis relacionats.....	19
Capítol 3. Tecnologia utilitzada.....	21
3.1 Maquinari.....	21
3.1.1 Samsung Galaxy Nexus GT-I9250.....	21
3.1.1.1 SiRF SiRFstarIV GSD4t.....	21
3.1.2 LabJack U3-LV.....	23
3.1.3 TI INA 169.....	24
3.1.4 AMPLOC AMP50.....	24
3.2 Software.....	24
3.2.1 Android ICS 4.0.1.....	24
3.2.1.1 Nucli Linux.....	24
3.2.1.2 Funcionalitats afegides al nucli.....	25
3.2.1.3 Llibreries i serveis.....	26
3.2.1.4 Dalvik i Zygote.....	26
3.2.1.5 Marc d'aplicacions.....	27
3.2.1.6 Aplicacions.....	27
3.2.2 Programació de codi nadiu.....	29
3.2.3 Programació d'aplicacions per a Dalvik.....	29
3.2.3.1 Android SDK r18.....	29
3.2.3.2 Eclipse Indigo 3.7.2.....	30
3.2.4 LabJackPython.....	30
Capítol 4. Entorn de mesura del consum.....	31
4.1 Estudi previ.....	31
4.2 Disseny.....	31

4.2.1	Maquinari.....	31
4.2.1.1	Circuit amb sensor AMPLOC AMP50.....	32
4.2.1.2	Circuit amb xip TI INA 169.....	33
4.2.1.3	Circuit per a la mesura del voltatge de la bateria.....	35
4.2.2	Programari.....	36
4.3	Implementació.....	37
4.3.1	Maquinari.....	38
4.3.1.1	Circuit amb sensor AMPLOC AMP50.....	38
4.3.1.2	Circuit amb xip TI INA 169.....	39
4.3.2	Programari.....	46
4.4	Conclusió.....	48
	Capítol 5. Monitorització del funcionament intern.....	49
5.1	Estudi previ.....	49
5.1.1	Crides a sistema, dispositius i descriptor del fitxer.....	49
5.1.2	Capa d'abstracció del maquinari (HAL).....	50
5.1.3	UART.....	50
5.2	Intercepció de les crides a sistema al dispositiu de caràcters GPS.....	51
5.2.1	Camí seguit per una funció de localització.....	51
5.2.2	Modificació del nucli Linux.....	52
5.2.2.1	Obtenció de les fonts del nucli.....	53
5.2.2.2	Modificació, compilació i construcció del nucli.....	54
5.2.2.3	Instal·lació del nucli.....	55
5.2.3	Segrest de les crides a sistema.....	55
5.2.3.1	Compilació i construcció.....	78
5.2.4	Inserció del mòdul i obtenció dels resultats.....	80
5.3	Conclusions.....	80
	Capítol 6. Obtenció de les dades e interpretació.....	81
6.1	Relació entre funcions de l'API i les crides a sistema interceptades.....	81
6.1.1	Creació d'aplicació d'usuari.....	82
6.1.1.1	Creació d'un nou projecte.....	83
6.1.1.2	Edició de permisos.....	85
6.1.1.3	Edició de capes gràfiques.....	86
6.1.1.4	Edició d'activitat.....	88
6.1.2	Establiment de relacions.....	96
6.2	Establiment del consum de crides a sistema.....	97
6.2.1	Realització de proves.....	97
6.2.2	Tractament de resultats.....	98
6.2.2.1	Impacte del temps d'inabilitació sobre el consum energètic.....	98
6.3	Conclusions.....	102
	Capítol 7. Conclusions.....	103
7.1	Treball futur.....	103
	Capítol 8. Bibliografia.....	104

Llista de figures

Figura 1: Diagrama Gantt, esquerra.....	10
Figura 2: Diagrama Gantt, dreta.....	10
Figura 3: Intersecció d'esferes.....	15
Figura 4: Codificació i modulació GPS.....	16
Figura 5: Estructura d'un missatge de navegació GPS.....	17
Figura 6: Sumari de consum energètic.....	19
Figura 7: Variació del cost energètic per unitat de dades sobre 802.11.....	20
Figura 8: Samsung Galaxy Nexus.....	21
Figura 9: Localització del xip SiRF SiRFstartIV GSD4t al terminal SGN GT-I9250.....	22
Figura 10: Labjack U3-LV.....	23
Figura 11: TI INA 169.....	24
Figura 12: AMPLOC AMP50.....	24
Figura 13: Tux, logotip de Linux.....	25
Figura 14: Arquitectura d'Android.....	28
Figura 15: Efecte Hall.....	32
Figura 16: Relació entre corrent sentit i tensió de sortida (AMP50).....	33
Figura 17: Circuit amb sensor AMPLOC AMP50.....	33
Figura 18: Esquema xip TI INA 169.....	34
Figura 19: Circuit amb sensor TI INA 169.....	35
Figura 20: Circuit divisor de tensió per a la mesura del voltatge de la bateria.....	36
Figura 21: Circuit amb sensor AMP50 (connexions amb Labjack).....	38
Figura 22: Circuit amb sensor AMP50 (protoboard i connexions amb font).....	39
Figura 23: Circuit amb INA 169 (numeració de connexions).....	40
Figura 24: Connexions a la bateria.....	40
Figura 25: Soldadura del xip INA 169, les resistències de shunt i load, i el condensador de filtre	41
Figura 26: Pendent relatiu a les mesures del port FIO0.....	43
Figura 27: Font d'alimentació.....	43
Figura 28: Oscil·oscopi.....	44
Figura 29: Pendent relatiu a les mesures del port FIO1.....	46
Figura 30: Flux de la localització a Android.....	52
Figura 31: Menú de configuració del nucli.....	54
Figura 32: Creació d'un nou projecte, selecció d'un assistent.....	83
Figura 33: Creació d'un nou projecte, selecció de nom i tipus.....	83
Figura 34: Creació d'un nou projecte, selecció d'SDK.....	84
Figura 35: Creació d'un nou projecte, configuració.....	84
Figura 36: Creació d'un projecte, estructura.....	85
Figura 37: Gràfic d'obtenció de localització sense deshabilitament.....	99
Figura 38: Gràfic d'obtenció de localització amb 5s de deshabilitament.....	99
Figura 39: Gràfic de consum sense cobertura GPS.....	101

Llista de taules

Taula 1: Estimació de la durada del projecte.....	9
Taula 2: Freqüències assignades al sistema GPS i codificacions corresponents.....	15
Taula 3: Mesures del port FIO0 relatives a l'increment del corrent.....	42
Taula 4: Mesures del port FIO1 relatives a l'increment de la tensió.....	45
Taula 5: Comparació de consums energètics amb 5s i 10s de deshabilitació del servei.....	100
Taula 6: Comparació de consum generat a conseqüència de l'estat de l'efemèride.....	101

Llista de codis

Codi 1: Aplicació per a la mesura de la potència instantània (part 1).....	46
Codi 2: Aplicació per a la mesura de la potència instantània (part 2).....	47
Codi 3: Aplicació per a la mesura de la potència instantània (part 3).....	48
Codi 4: Mòdul (part 1).....	56
Codi 5: Mòdul (part 2).....	56
Codi 6: Mòdul (part 3).....	56
Codi 7: Mòdul (part 4).....	59
Codi 8: Mòdul (part 5).....	68
Codi 9: Mòdul (part 6).....	68
Codi 10: Mòdul (part 7).....	68
Codi 11: Mòdul (part 8).....	69
Codi 12: Mòdul (part 9).....	70
Codi 13: Mòdul (part 10).....	71
Codi 14: Mòdul (part 11).....	74
Codi 15: Mòdul (part 12).....	74
Codi 16: Mòdul (part 13).....	78
Codi 17: Mòdul (part 14).....	78
Codi 18: Makefile del mòdul (part 1).....	79
Codi 19: Makefile del mòdul (part 2).....	79
Codi 20: Edició de permisos, AndroidManifest.xml.....	86
Codi 21: Edició de capes gràfiques, res/layout/main.xml.....	88
Codi 22: Edició d'activitat, importació de classes.....	88
Codi 23: Edició d'activitat, funció onCreate i obtenció de referència del servei LM.....	89
Codi 24: Edició d'activitat, listener LocationManager i esdeveniments.....	92
Codi 25: Edició d'activitat, listener GpsLocationProvider i esdeveniments.....	95

Capítol 1. Introducció

Avui dia cada vegada son més els usuaris de dispositius mòbils, com ara telèfons o tablets, que utilitzen aplicacions de diferent índole a diari. Aquestes aplicacions a la vegada utilitzen una sèrie d'elements interns que permeten augmentar les possibilitats dels dispositius finals, com ara el xip GPS. Cadascun d'aquests elements comporta un consum associat a les accions que realitza i repercuteix en major o menor mesura en el consum general dels dispositius. Aquest fet inevitable acostuma a ser un dels principals inconvenients que tenen avui dia aquests dispositius arribant a provocar una gran dependència diària de fonts d'alimentació per carregar aquests. Amb aquesta problemàtica neix la motivació d'aquest projecte amb el qual s'intentarà contribuir en la millora d'aquest aspecte als dispositius mòbils.

Aquest projecte es basa per una part en entendre el funcionament del sistema Android així com el dels principals actors que hi participen en el funcionament d'aquest. Una vegada conegut el funcionament del sistema es pretén obtindre una mesura del consum que es produeix per part de les aplicacions d'usuari que utilitzen els diferents elements que té un dispositiu. Com a cas d'estudi concret per al desenvolupament d'aquest projecte s'utilitzarà un terminal mòbil i la implicació que té l'ús de les aplicacions d'usuari que utilitzen el sistema GPS sobre el consum global.

S'ha escollit Android per la millor accessibilitat al codi font que d'altres sistemes operatius i per la seva gran quota de mercat actual. En quant al xip GPS s'ha escollit perquè per experiència pròpia és un dels elements que més consumeixen al dispositiu.

A continuació es mostraran amb més definició els objectius marcats per a la realització d'aquest projecte:

1.1 Objectius

Aquest projecte neix amb uns objectius generals i personals que han sigut determinants per a la seva elecció: entendre el sistema operatiu Android, el nucli Linux i la interacció entre aquests. El fet que siguessin objectius poc específics per a un projecte de fi de carrera va comportar la cerca i selecció d'uns objectius definits que proporcionessin, de forma indirecte i parcial, el compliment dels objectius generals. Per a aquesta cerca es van intentar compaginar aquests objectius generals amb la millora de l'experiència d'un usuari amb un dispositiu Android i es va arribar a la conclusió que, tenint en compte la poca duració de la bateria en aquests dispositius, seria convenient entendre el funcionament d'un component intern del dispositiu a dintre del sistema operatiu i mesurar el consum produït per aquest segons les accions que realitza. Gràcies a la pròpia experiència amb aquests dispositius es va acabar seleccionant el xip GPS degut a que, amb una simple observació, es pot relacionar la utilització

del GPS amb un increment en el consum d'energia per part d'un dispositiu. Així doncs sorgeixen els següents objectius definits per a aquest projecte:

1. **Objectiu general:** mesurar el consum produït per una aplicació d'usuari que utilitzi el sistema GPS sobre el consum general d'un terminal mòbil.

2. Objectius parciais

- 2.1 Estudiar i comprendre el funcionament del sistema GPS d'una forma global i de forma específica a un dispositiu amb sistema operatiu Android.
- 2.2 Estudiar i comprendre la creació d'una aplicació d'usuari que utilitzi els serveis per la localització geogràfica del dispositiu.
- 2.3 Estudiar la interacció dels elements que conformen el sistema operatiu per a tindre una idea general del flux que segueixen les accions realitzades des d'una aplicació d'usuari fins al xip GPS.
- 2.4 Preparar l'entorn de desenvolupament necessari tant per a la creació d'applicacions d'usuari com per a treballar a nivells nadius.
- 2.5 Preparar l'entorn per a la mesura del consum d'energia per a poder realitzar la comprovació del consum energètic d'un dispositiu segons les accions que se realitzin en aquest.
- 2.6 Interceptar les crides a sistema originades per el controlador GPS al executar-se funcions de localització de l'API d'Android des d'una aplicació d'usuari.
- 2.7 Relacionar l'execució de funcions de localització geogràfica de l'API utilitzades des d'una aplicació amb patrons de crides a sistema interceptades que es generen.
- 2.8 Mesurar el consum energètic dels patrons obtinguts de crides a sistema realitzades per el controlador GPS.
- 2.9 Tenint per una banda la relació establerta entre les funcions de l'API executades des d'una aplicació d'usuari i les crides a sistema realitzades per el controlador, i per altra banda el consum energètic mesurat amb l'execució d'aquestes crides, estimar el consum energètic generat per una aplicació d'usuari a conseqüència de la utilització del xip GPS.

3. Objectius indirectes

- 3.1 Estudiar l'impacte que té sobre el consum de l'energia d'un dispositiu Android la utilització del GPS per part d'un usuari.
- 3.2 Estudiar la capacitat d'acció que té un programador, usuari de l'API d'Android, sobre el control del consum de l'energia produït per el xip GPS.

1.2 Motivació

Les raons de la selecció d'aquest projecte son variades i tenen un objectiu comú, l'estudi i la contribució a la comunitat del software lliure. Personalment per els meus ideals estic posicionat a favor del software lliure i tinc la intenció de retornar part del que m'ha sigut proporcionat aquests anys.

Primerament perquè jo mateix estic interessat en l'estudi del kernel, sobretot des de que vaig cursar l'assignatura PROSO la qual va ser detonant en la meva predilecció sobre l'estudi d'aquesta part del sistema operatiu. En quant a kernels, per la utilització que té avui dia i per poder inspeccionar el codi al ser aquest obert, el kernel Linux és el que hi vull dedicar-me a estudiar.

Una altre de les raons per les que hem vaig decantar per a aquest projecte és la creixent utilització de telèfons mòbils, tauletes i d'altres dispositius mòbils i encastats amb sistema operatiu avançat. Novament, per la seva utilització a aquests dispositius (més del 50% a inicis de l'any 2012) i, encara que parcialment, el codi és obert, el sistema operatiu Android és la meva primera opció d'estudi a aquests dispositius.

També cal dir que, tenint en compte les raons anteriors, el meu estudi en aquests àmbits no acaba amb aquest projecte. Aquest projecte és un primer pas en l'estudi dels sistemes operatius i em servirà per a poder afrontar nous reptes de recerca futurs.

1.3 Planificació

A la planificació del projecte es va aproximar en un principi la feina a realitzar segons el nombre de crèdits a realitzar. És una estimació sense convenients temporals i per tant un esquema totalment idílic:

<i>crèdits</i>	<i>crèdit/hora</i>	<i>hores</i>	<i>hores/dia</i>	<i>dies</i>	<i>mesos</i>
22,5	20	450	3,75	120	4,5

Taula 1: Estimació de la durada del projecte

D'aquesta forma, al iniciar el projecte al 29 de gener i havent realitzat els càlculs del temps a dedicar (taula 1) es preveia que al voltant de principis de juny s'acabaria aquest i es procediria a la lectura. Un cop estimat el temps es van assignar les hores a les diferents tasques a realitzar.

A continuació es mostra el diagrama Gantt amb les tasques planificades:

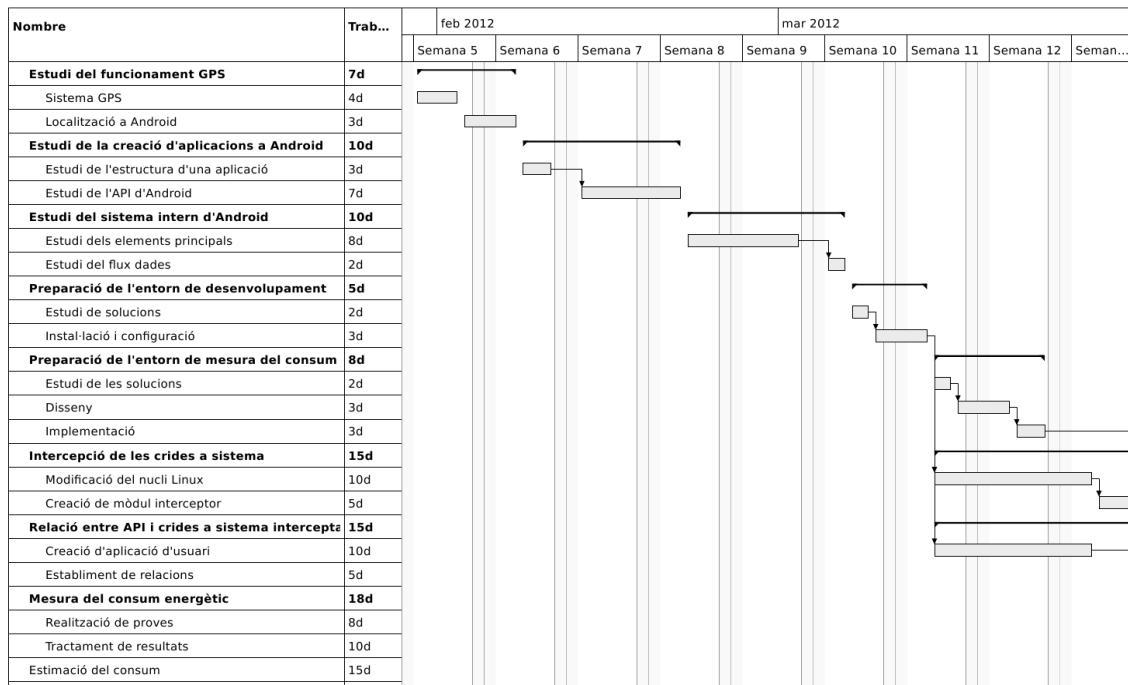


Figura 1: Diagrama Gantt, esquerra

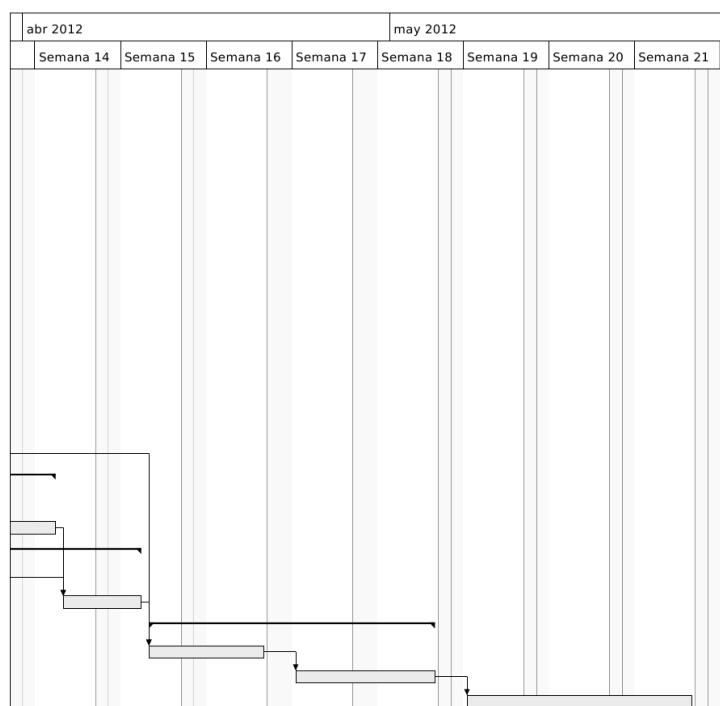


Figura 2: Diagrama Gantt, dreta

1.4 Metodologia

Per a la realització d'aquest projecte s'han seguit les següents pautes per a la realització de cada objectiu parcial i de l'objectiu general:

1. Estudi previ de les àrees amb les que es treballarà. Utilització de bibliografia, recerca personal, etc.
2. Treball previ sobre l'àrea de treball del projecte per a consolidar els coneixements adquirits a l'estudi previ. Realització de petits casos d'estudi i d'altres proves.
3. Treball sobre l'objectiu definit a cada àrea.
4. Comprovació de l'assoliment de l'objectiu marcat.
5. Extracció de conclusions sobre el treball realitzat.

1.5 Estructuració de la memòria

Amb aquesta secció s'intenta donar una visió estructural de la memòria per a que un lector pugui cercar ràpidament la secció corresponent al seu interès. Per a començar dir que aquesta memòria ha estat escrita en l'ordre cronològic en el que s'ha desenvolupat el projecte.

Primerament, i tot seguit a aquesta secció, es troba el capítol 2 “Entorn del projecte i estat de l'art”. En aquest capítol s'expliquen amb detalls suficients com funciona el sistema GPS i com es comunica amb els diferents dispositius que l'utilitzen, com funciona el sistema de geolocalització al sistema operatiu Android i quins elements hi participen, i quins estudis s'han realitzat que tinguin relació amb aquest projecte final de carrera.

Després d'explicar l'entorn del projecte es troba el capítol 3 “Tecnologia utilitzada” on es troben explicats cadascun dels elements maquinari i software que s'han seleccionat i s'han utilitzat per a la realització d'aquest projecte.

Una vegada explicats els punts anteriors es passen a explicar els capítols que conformen el gruix on s'ha desenvolupat el projecte. Començant per el capítol 4 “Entorn de mesura del consum” s'expliquen el disseny i la implementació de l'entorn que permetrà obtindre mesures del voltatge utilitzat per el dispositiu emprat a l'estudi, i que serà necessari per extreure relacions i conclusions entre la utilització del GPS al dispositiu i el consum d'aquest.

Tot seguit s'explica el capítol 5 “Monitorització del funcionament intern” on s'indiquen les passes seguides per arribar a interceptar les crides a sistema realitzades sobre el xip GPS del dispositiu utilitzat. D'aquesta forma es poden arribar a establir relacions entre les crides a sistema realitzades i les funcions cridades desde l'API d'Android per part d'una aplicació d'usuari.

Una vegada explicat el procés de monitorització de les crides a sistema es passa a explicar el capítol 6 “Obtenció de les dades e interpretació” on, una vegada obtinguda la informació anterior i tenint els entorns de treball necessaris, es passa a realitzar relacions entre accions realitzades i el consum utilitzat per el dispositiu.

Finalment es passa a extreure conclusions sobre el projecte, la seva realització i la visió de futur que es té sobre el treball realitzat.

Capítol 2. Entorn del projecte i estat de l'art

En aquest capítol es mostrerà com funciona el sistema GPS, com s'utilitza la localització geogràfica al sistema operatiu Android, i quins estudis s'han fet que tinguin relació amb aquest projecte.

2.1 Sistema GPS

En aquesta secció es tractarà de donar una visió global de com funciona el sistema GPS i quines estructures de dades s'envien a través d'aquest.

2.1.1 Funcionament

El sistema GPS, o Sistema de Posicionament Global, és un Sistema Global de Navegació per Satèl·lit (GNSS) desenvolupat al 1973 i instal·lat i controlat per al Departament de Defensa dels Estats Units des del 1994. Aquest sistema permet determinar la posició d'un objecte a qualsevol punt del món de forma gratuita per a usos civils, comercials i militars.

El sistema GPS està format per un número de 24 a 32 satèl·lits operatius, depenent del seu estat, els quals mantenen una inclinació de 55° respecte a l'equador. Aquests satèl·lits no son estacionaris i cobreixen tot el globus des de l'òrbita terrestre mitjana, MEO o *Medium Earth Orbit*, a una altura aproximada d'uns 20200 km de la Terra i a una velocitat aproximada de 4 km/s, realitzant una òrbita complerta en menys de 12 hores de forma síncrona entre aquests. A més del grup de satèl·lits també formen part d'aquest sistema 2 estacions de control, 4 antenes de terra, 6 estacions de monitoreig, i com no, els receptors del servei d'usuari.

Per a l'enviament de la informació aquest sistema compta amb 5 tipus de freqüències (*veure taula 2*) sent 1575,42 Mhz la utilitzada per a l'ús civil. Els satèl·lits envien contínuament missatges amb la informació modulada sobre aquesta freqüència i codificada per a l'emissió per el mateix canal amb el sistema d'Accés Múltiple per Divisió de Codi (CDMA). Aquest sistema utilitza un codi únic per satèl·lit, PRN o *pseudo-random number*, que serveix tant per a codificar els bits que formen les dades a enviar com per a descodificar-los per un dispositiu receptor de GPS. Els codis assignats a cada satèl·lit son ortogonals entre ells i estan formats per chips, els quals son polsos a l'amplitud d'una ona electromagnètica i a la vegada son divisions

Banda	Freqüència	Codificació
L1	1575.42 Mhz	Codi d'adquisició aproximada (C/A), codi de precisió xifrat P(Y), i codis L1 civil (L1C) i militar (M) als futurs satèl·lits de 3 ^a generació.
L2	1227.60 Mhz	Codi de precisió xifrat P(Y), i els codis L2 civil (L2C) i militar (M) als satèl·lits IIR-M i als de 3 ^a generació.
L3	1381.05 Mhz	Utilitzats per a deteccions de detonacions nuclears (NUDET).
L4	1379.913 Mhz	En estudi per a la correcció a la ionosfera.
L5	1176.45 Mhz	Proposta per a ús de la seguretat civil (SoL).

Taula 2: Freqüències assignades al sistema GPS i codificacions corresponents

iguals del temps d'un bit. La codificació CDMA utilitzada actualment per a l'ús civil en el sistema GPS és la d'Adquisició Aproximada (C/A), la qual transmet dades a 1,023 milions de chips per segon.

Una vegada desmodulada i descodificada la senyal obtinguda, rebutjant per cada satèl·lit la resta de la informació com a soroll, el receptor calcula la seva posició amb la tècnica de trilateració. Per a entendre aquesta tècnica hem d'imaginar que cada satèl·lit forma una esfera de cobertura on aquests hi son al centre i aquestes intersecten entre elles al punt on hi és el receptor (*veure figura 3*). Per a calcular la posició en termes de latitud i longitud es necessiten com a mínim 3

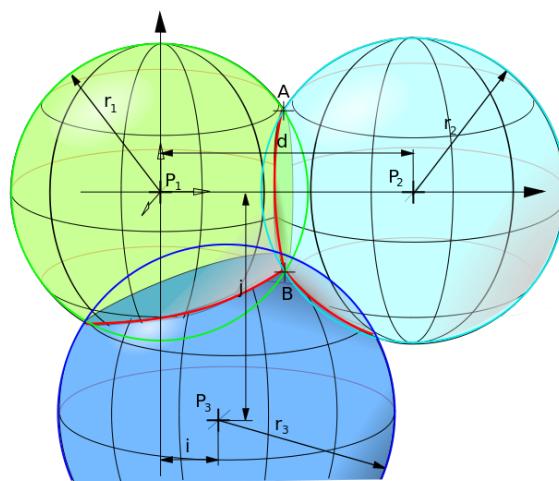


Figura 3: Intersecció d'esferes

satèl·lits les esferes de cobertura dels quals intersectin, i per a obtindre l'altura es necessiten com a mínim 4. El receptor també calcula el temps gràcies a la informació obtinguda dels rellotges atòmics que hi son als satèl·lits. La seva exactitud és aproximadament de 15 m a l'espai i de 1 ns al temps, no obstant és necessari realitzar certes correccions degut a la velocitat del senyal (300000 km/s) i el seu retard en arribar al receptor.

A més d'aquest sistema existeixen d'altres, un en ús i d'altres en desenvolupament. L'altre sistema en ús és el GLObal Navigation Satellite System rus (GLONASS), utilitzat inicialment en exclusivitat per a l'ús militar, i obert al 2007 per a l'ús civil. Els sistemes en desenvolupament són l'Europeu Galileo, el Xinès Compass i l'Indi Regional Navigational Satellite System.

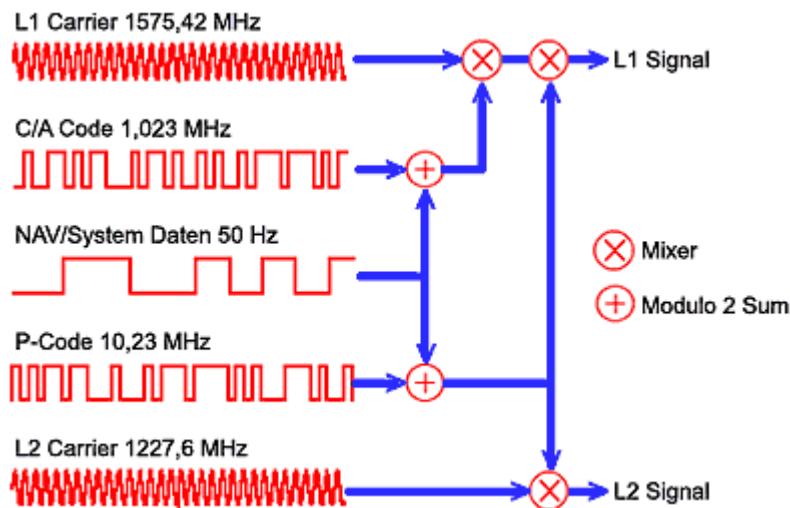


Figura 4: Codificació i modulació GPS

2.1.2 Tipus de missatges

La informació codificada i modulada al sistema GPS s'envia en forma de missatge de navegació. Un missatge té 1500 bits i s'envia a 50 bits/s contínuament des de cada satèl·lit. Cada missatge es divideix en 5 subrames de 300 bits i a la vegada aquestes estan dividides en 10 paraules. La primera paraula de cada subrama és la paraula de telemetria, *Telemetry Word* o (TLM), la qual permet la detecció de l'inici de subrama i la determinació del temps en el que la subrama es va enviar. La següent paraula és la paraula de lliurament, *HandOver Word* o (HOW), que proporciona el temps en el que les següents paraules seran transmeses. A les següents paraules trobem informació diferent dependent de cada subrama:

La primera subrama conté el número de setmana actual i la informació per sincronitzar el temps del dispositiu GPS amb el del satèl·lit. També conté l'estat del satèl·lit i la seva operabilitat.

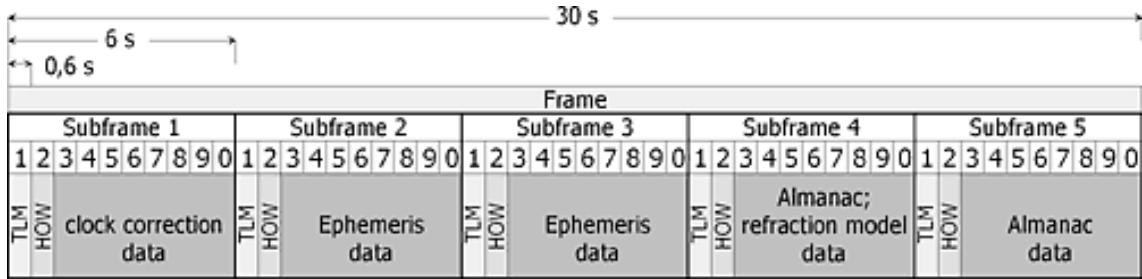


Figura 5: Estructura d'un missatge de navegació GPS

Tot seguit es troben les subrames 2 i 3 les quals contenen les dades d'efemèrides, o dades de la posició en òrbita. Aquestes dades son necessàries per a calcular de forma precisa on era el satèl·lit a l'inici del missatge. La informació és renovada cada 2 hores per cada satèl·lit i és valida durant no més de 4 hores. L'efemèride completa és rebuda en un temps aproximat de 18 a 36 segons i és rebutjada si es perd la senyal mentre s'està rebent cada part d'aquesta. El temps en obtindre l'efemèride serà clau en la obtenció de la primera posició.

Finalment, les subrames 4 i 5 contenen les dades de l'almanac. L'almanac consisteix en la informació de la òrbita seguida per els satèl·lits, l'estat d'aquests, un model ionosfèric, i la informació per a relacionar el temps de cada satèl·lit amb el Temps Universal Coordinat (UTC). L'almanac és renovat cada 6 hores i és valid fins a 180 dies des de la seva actualització. Cada missatge transporta una part de l'almanac en aquestes 2 subrames, sent 25 missatges els necessaris per a obtindre l'almanac complet, o el que es el mateix, 12'5 minuts. Aquests missatges són enviats per tots els satèl·lits contínuament. Un dispositiu GPS utilitza la informació de l'almanac per saber-ne de quins satèl·lits pot rebre la informació i així filtrar aquesta amb el PRN, per ajustar el seu temps a l'horari UTC, i per realitzar la correcció de l'error produït per la ionosfera.

2.2 Geolocalització a Android

En aquesta secció es mostrerà com s'obté la localització des d'un dispositiu amb un sistema operatiu Android amb els receptors GPS i les alternatives utilitzades complementàriament a aquest sistema.

2.2.1 Utilització dels recursos

Avui dia una gran part dels dispositius que utilitzen el sistema operatiu Android suporten el sistema GPS afegint-ne un xip receptor a dintre del dispositiu. Moltes aplicacions utilitzen aquest sistema, com per exemple Maps o Navigation de Google, per mitjà de les diferents Interfícies de Programació d'Aplicacions (API). Una d'aquestes API, en la que s'ha basat gran part d'aquest projecte, és l'API d'Android, on podem trobar el paquet **android.location** que conté diferents classes i interfícies per a la localització del dispositiu. A continuació es descriuen les diferents interfícies i classes que es posen a disposició per a la programació d'applications:

Interfícies

Les interfícies són classes amb objectes que fan de pont entre serveis i l'aplicació per a rebre actualitzacions de forma asíncrona i mètodes per a gestionar aquests canvis. En aquest cas concret el servei utilitzat és el servei **LocationManager** i les interfícies són les següents:

- **GpsStatus.Listener**: utilitzat per a obtindre canvis al servei GPS, com ara la seva aturada o la seva arrencada.
- **GpsStatus.NmeaListener**: utilitzat per a realitzar accions després d'haver-ne rebut una sentència NMEA¹.
- **LocationListener**: utilitzat per a rebre notificacions una vegada la localització canviï. Aquesta interfície funcionarà sempre que s'hagi definit la recepció d'actualitzacions amb el mètode **requestLocationUpdates** de la classe LocationManager.

Classes

Cada classe de l'API té una sèrie d'objectes entre els quals destaquem els mètodes que s'utilitzen per al tractament de dades i la realització d'accions. A continuació s'expliquen cadascuna d'aquests:

- **Address**: classe per al tractament d'objectes del tipus adreça i així obtindre dades com ara un telèfon associat o la localitat a la qual hi pertany.
- **Criteria**: és una classe útil per a la definició de criteris d'obtenció de proveïdors de localització, dels quals es parlarà més endavant al punt 2.2.2. Es poden definir criteris tals com la utilització d'energia o la precisió. Ha de ser utilitzada amb els mètodes que sol·liciten l'obtenció d'actualitzacions de localització.
- **Geocoder**: aquesta classe serveix tant per resoldre a partir d'una adreça el seu punt al mapa en termes de latitud i longitud, o per a convertir un punt en el mapa en una adreça indicativa i descriptiva.
- **GpsSatellite**: classe útil si el que es vol es obtindre informació dels satèl·lits GPS. Es pot obtindre l'orbital d'aquests o el seu PRN.
- **GpsStatus**: necessària per a utilitzar la classe anterior a l'hora d'obtindre els satèl·lits perceptibles per el xip GPS.
- **Location**: aquesta classe s'utilitza per al tractament de les localitzacions geogràfiques obtingudes en particular moment. L'objecte location, utilitzat per aquesta classe, consisteix en els camps latitud, longitud, marca de temps UTC, i opcionalment altitud, velocitat i rum.
- **LocationManager**: aquesta classe proveeix l'accés al serveis de localització. Aquests serveis permeten a les aplicacions obtindre actualitzacions de la posició geogràfica del dispositiu o produir un esdeveniment al complir-se unes condicions determinades.

2.2.2 Alternatives

A més del sistema GPS hi han més elements que poden millorar l'experiència de la localització geogràfica al sistema operatiu Android, millorant la precisió, disminuint el temps d'obtenció de localització, o reduint el consum energètic del dispositiu. Aquests elements són el sistema WiFi, la telefonia cel·lular o els sensors. Els 2 primers es poden utilitzar gràcies a que

1 NMEA 0813 és un estàndard de comunicacions entre dispositius, típicament a través d'un port sèrie.

Google recopila, emmagatzema i actualitza per mitjà del wardriving² les diferents adreces MAC que emeten per broadcast els punts d'accés WiFi, els identificadors de cel·les GSM/UMTS de telefonia, i les corresponents localitzacions geogràfiques. Aquesta informació és accessible a través de les seves APIs de localització. La precisió que es pot obtindre amb aquest sistema és de fins a 100 m amb els punts d'accés WiFi i de 150 m a 4 km amb les cel·les de telefonia. La utilització d'aquesta tècnica així com la utilització del GPS es defineix a les diferents classes de l'API de Google. En quant als sensors tenim l'acceleròmetre, els giroscopis i la brúixola. Aquests sensors són configurables des de la interície d'usuari i no des de l'API i permeten una precisió raonable a la vegada que estalvienc considerablement el consum.

2.3 Estudis relacionats

Avui dia s'han realitzat varis estudis on es cerca una millora en la eficiència energètica amb els diferents elements que té un dispositiu mòbil. Hi han alguns que es centren en l'estudi del consum d'un element concret del dispositiu i d'altres que van més enllà realitzant implementacions que rebaixen el consum. Entre els estudis més rellevants que tenen relació amb aquest projecte es destaquen els 2 següents:

Energy-Aware Location Provider for the Android

Aquesta tesi realitzada per Mohamed Amir, estudiant del màster en ciències de la computació a la Universitat d'Alexandria, entre gener del 2007 i abril de 2010, tracta de donar una alternativa a l'ús exclusiu del GPS utilitzant, de forma combinada amb aquest, l'acceleròmetre i la brúixola. A més tracta de mostrar el consum entre cada sistema plantejat i la fidelitat dels resultats de posicionament una vegada s'utilitza el sistema proposat.

http://www.mpi-inf.mpg.de/~mimir/mimir_masters_thesis.pdf

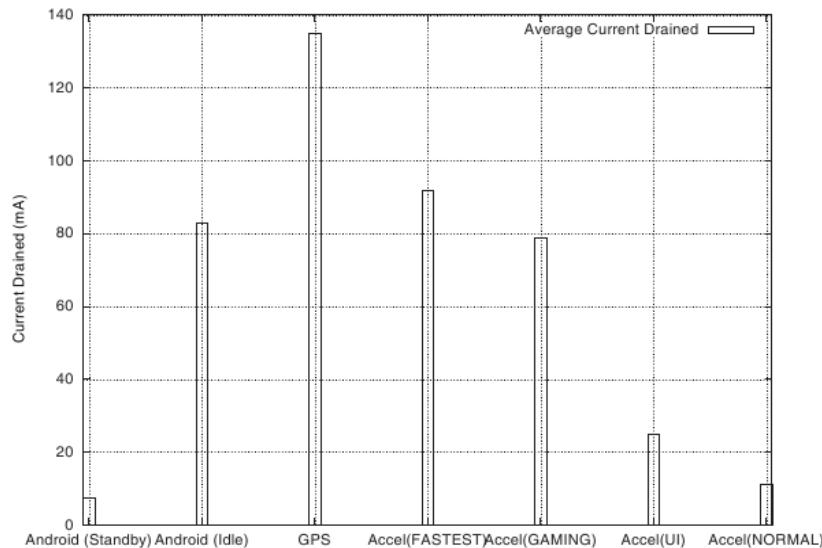


Figura 6: Sumari de consum energètic

2 El wardriving és la cerca de xarxes inal·làmbriques WiFi i cel·les de telefonia amb un vehicle en moviment per a determinar la seva ubicació.

Measuring mobile phone energy consumption for 802.11

Paper realitzat per Andrew Rice i Simon Hay, membres del Digital Technology Group del laboratori de computació de la Universitat de Cambridge, al juliol del 2010, mostra el consum produït per el xip WiFi i com influeixen les transferències de dades dependent de com es realitzen aquestes.

<http://www.cl.cam.ac.uk/~sjeh3/wireless/paper.pdf>

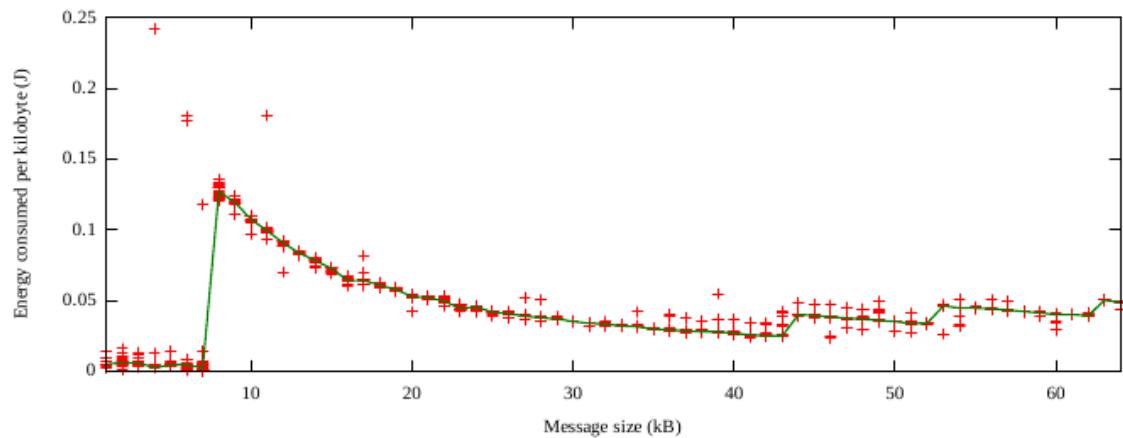


Figura 7: Variació del cost energètic per unitat de dades sobre 802.11

Capítol 3. Tecnologia utilitzada

En aquest capítol es tractarà d'explicar quins són els elements maquinari i software que s'han utilitzat per a la realització del projecte i la seva implicació sobre aquest.

3.1 Maquinari

En aquesta secció s'explicaran tots els elements maquinari que s'han utilitzat per a realitzar el projecte i la seva repercuSSIó sobre aquest.

3.1.1 Samsung Galaxy Nexus GT-I9250

Per a realitzar l'estudi d'un dispositiu mòbil s'ha triat un smartphone Samsung Galaxy Nexus ja que, en el seu moment, era l'únic terminal al mercat que oferí l'última versió sistema operatiu d'Android, ICS 4.0.1, i per aquesta raó es podrien utilitzar les funcions més avançades de l'API d'Android. Una altre de les raons és que aquest dispositiu té un volum raonable d'informació de com és el sistema per dins, tant el maquinari com el software, i això és clau per a l'estudi posterior. També s'ha triat el terminal perquè compta amb una gran varietat de xips, sensors, i d'altres elements que permeten l'explotació del telèfon a través de l'API.

3.1.1.1 SiRF SiRFstarIV GSD4t

El xip SiRFstarIV GSD4t GPS és l'encarregat de servir el sistema GPS al terminal. Aquest xip utilitza una CPU pròpia per a córrer les seves llibreries de navegació la qual va a una baixa freqüència gràcies als buffers i les cues internes al xip.

En quant a l'energia és rellevant conèixer que aquest xip es manté en calent a un baix consum, desestimant la possibilitat d'apagar-lo, disposit a realitzar una captura o actualització amb



Figura 8: Samsung Galaxy Nexus

només un corrent de 150 a 500 μ A. El mode de funcionament també pot ser configurat per l'usuari segons les necessitats d'aquest. Entre els diferents modes de funcionament cal destacar el Trickle Power Mode, en el qual el receptor GPS entra periòdicament en mode standby per a estalviar energia. En aquest mode, el xip envia la informació de posició al terminal a un ritme definit per l'usuari. Trickle Power Mode és una seqüència on es repeteix l'execució dels següent modes:

1. Track Mode: durant aquest mode, totes les parts del xip estan en funcionament consumint el màxim d'energia.
2. CPU Mode: durant l'execució d'aquest mode, solament el microcontrolador està en execució i no es rastrejen satèl·lits. En aquest mode es calcula la localització geogràfica del terminal a partir de la informació rebuda en el mode anterior.
3. Standby Mode: en aquest mode, tant el microcontrolador com el rastrejador de satèl·lits hi son apagats. No es sortirà d'aquest mode fins que no es produexi un esdeveniment programat o una petició externa.

Aquest cicle consumeix uns 8 mW a una freqüència d'actualització de 1 Hz.

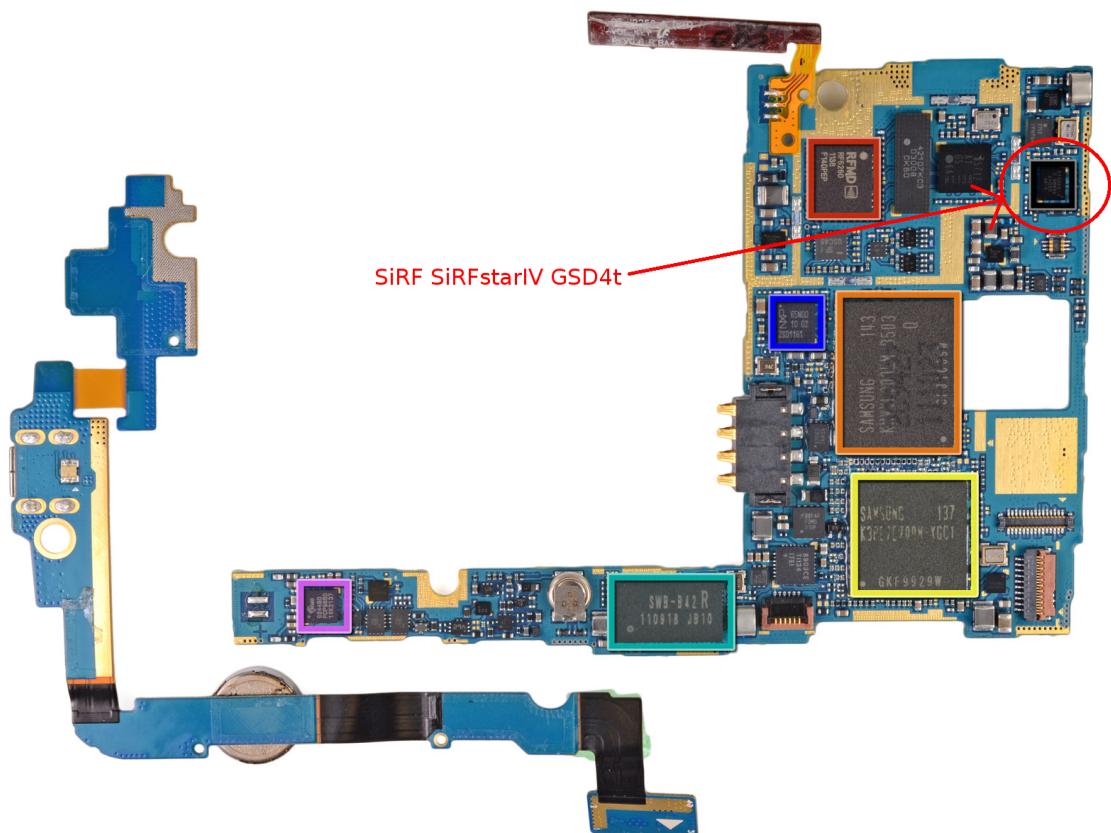


Figura 9: Localització del xip SiRF SiRFstarIV GSD4t al terminal SGN GT-I9250

3.1.2 LabJack U3-LV

Per a la mesura del consum energètic es va pensar en varies alternatives. Primerament es va pensar en aplicacions Android com PowerTutor³, les quals utilitzen una predicció del consum produït segons uns models establerts. Aquesta opció va ser descartada ja que és una tècnica realitzada per a uns certs terminals i no s'obtindria el consum real. L'altra opció era realitzar un consum real del terminal. Per això es va pensar en un primer moment en utilitzar un parell de multímetres⁴ amb interfície USB que permetrien emmagatzemar el consum directament a l'ordinador d'una manera econòmica. La opció va ser descartada ja que, per consell del director del departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial (ESAII), Antonio Benito Martínez Velasco, el més apropiat i formatiu seria la utilització d'un element de mesura amb interfície USB, amb entrada i sortida digital o analògica, i relativament econòmic, com és el LabJack U3-LV.



Figura 10: Labjack U3-LV

El LabJack U3-LV és un conjunt d'elements de mesura encapsulats en una carcassa de plàstic amb la que es facilita la connexió de fils conductors. Entre d'altres, les característiques més interessants per a aquest projecte son les següents:

- Connexió USB per a la transmissió de dades entre l'ordinador i el LabJack.
- 8 connectors d'entrada/sortida configurables per a la seva utilització analògica o digital amb la capacitat de lectura analògica de fins a 2,44 V.
- API per a la programació d'aplicacions on es poden configurar com funcionen els connectors d'entrada/sortida i la lectura d'aquests.
- Sortida de 5 V per a l'alimentació de circuits lògics externs.

³ <http://powertutor.org/>

⁴ <http://www.uni-trend.com/UT108.html>

3.1.3 TI INA 169

El xip INA 169⁵ realitza la conversió de la diferència entre 2 tensions d'entrada en un corrent de sortida. Aquesta conversió és necessària ja que, com s'ha citat abans, el LabJack U3-LV suporta una entrada de tensió de fins a 2,44 V, limit superat per la tensió comuna mesurada al dispositiu mòbil (entorn als 3,7 V). El seu rang d'entrada de tensió és troba entre 2,7 V i 60 V, suficient per a l'entrada de tensió del terminal i per la tensió de referència de 5 V obtinguda a partir del LabJack, concretament del seu port USB. Aquest xip està recomanat per a la realització de mesures en telèfons cel·lulars.

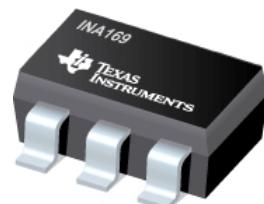


Figura 11: TI INA 169

3.1.4 AMPLOC AMP50

Una alternativa al xip INA 169 consistia en un sensor de corrent induïda, AMPLOC AMP50, que permetia la mesura del corrent enrotllant el cable conductor de sortida del terminal al sensor en forma de bobina. Al capítol 4 s'explica per què no va resultar la implementació d'un circuit amb aquest sensor.



Figura 12: AMPLOC AMP50

3.2 Software

En aquesta secció s'explicaran tots els elements software que s'han utilitzat per a realitzar el projecte i la seva repercuSSIó sobre aquest.

3.2.1 Android ICS 4.0.1

L'estudi d'aquest projecte està basat en el sistema operatiu Android. Aquest sistema operatiu és generalment utilitzat per dispositius intel·ligents, com terminals mòbils o tablets, i concretament per el dispositiu emprat per aquest estudi a la seva versió Ice Cream Sandwich 4.0.1. Android, com tot sistema operatiu, està compost per varíes parts, entre aquestes cal destacar el seu nucli Linux a la versió 3, el programari intermediari, o *Middleware*, i les diferents aplicacions d'usuari:

3.2.1.1 Nucli Linux

El nucli Linux és la part del sistema operatiu que aporta la base funcional d'aquest. Una de les seves funcions principals és la de actuar com una capa d'abstracció entre el maquinari i la resta del programari. Per a realitzar això el programari s'executa en diferents modes dependent de les necessitats d'accés al maquinari, i de la seguretat que això pot comportar. Aquests modes son possibles ja que el processador aporta aquesta funcionalitat i és explotada per el nucli. Els

5 <http://www.ti.com/product/ina169>

controladors de dispositius i les parts del nucli s'acostumen a executar en mode privilegiat, també dit a l'espai del nucli, tenint ple accés al maquinari sense cap restricció. La resta de programari s'executa en mode no privilegiat, també dit a l'espai d'usuari, tenint accés únicament al maquinari a través del programari executat en mode privilegiat. Utilitzant aquesta funcionalitat el nucli actua com a gestor de la memòria proveint al sistema de la seguretat i consistència necessària. Una altra de les funcionalitats del nucli és la de gestionar l'execució dels processos assignant a aquests espai de memòria i de temps d'execució als diferents processadors del dispositiu, i priorititzant les execucions. El nucli també és l'encarregat del tractament de les interrupcions al sistema combinant aquesta funcionalitat amb l'anterior comentada. El nucli Linux també permet la càrrega i descàrrega de part d'aquest dinàmicament en temps d'execució en forma de mòduls. Els nuclis Linux utilitzats a Android normalment tenen aquesta funcionalitat desactivada havent-hi de configurar i compilar novament el nucli per a utilitzar-la.

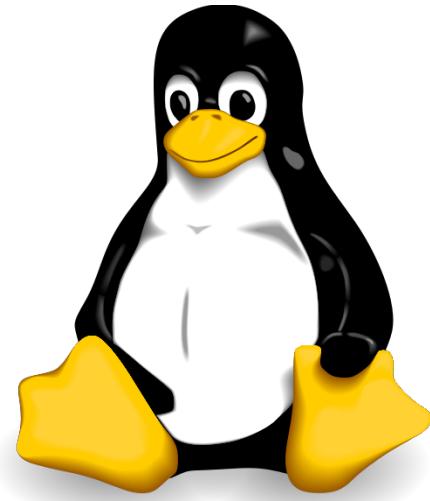


Figura 13: Tux, logotip de Linux

3.2.1.2 Funcionalitats afegides al nucli

Al nucli Linux d'Android s'ha afegit o canviat el codi que s'ha creut necessari per a que aquest compleixi les condicions de treball necessàries en un dispositiu mòbil o encastat amb el sistema operatiu Android. Els principals canvis han sigut:

- **Correccions d'errors i habilitació de nou maquinari:** s'ha afegit el mòdul **pmem** per a la gestió de la memòria física, el controlador **adb** (de l'anglès android debug bridge) per a la depuració del programari al dispositiu, el controlador **GPIO** (en anglès General Purpose Input/Output, entrada/sortida de propòsit general), i d'altres canvis i correccions necessàris.
- **Millora de la gestió de l'energia:** s'ha afegit el mòdul **early suspend** per a la gestió de l'estat de suspensió del dispositiu, el mòdul **wakelocks** per al control de la suspensió per part d'aplicacions o serveis, i el mòdul **alarm timer**, necessari per a la gestió i funcionament del sistema d'alarma d'Android.
- **Millora del registre d'errors:** el mòdul **logger** per a la gestió de registres i els mòduls **ram_console** i **apanic** que permet desar els missatges del nucli a RAM i permetre la seva recuperació davant un kernel panic.
- **Increment de la seguretat:** la opció del nucli **paranoid network** que permet accedir-hi a les diferents funcionalitats de xarxa depenent del grup del procés que la requereix.
- **Millora del rendiment:** el mòdul **ashmem** per a una millor gestió de la memòria virtual en dispositius amb un recurs RAM reduït, el sistema d'alliberació de memòria virtual **LowMemoryKiller**, el sistema de fitxers **Yaffs2 fs**, i el sistema Binder que serà explicat a continuació.

Android està dissenyat per a que tots els processos a l'espai d'usuari es comuniquin per mitjà del seu propi mecanisme **IPC** (de l'anglès *Inter-Process Communication, Comunicació Entre Processos*). El nucli d'aquest mecanisme és el controlador Binder el qual és un mecanisme **RPC** (de l'anglès *Remote Procedure Call, Crida a Procediment Remot*) lleuger que s'encarrega de traspasar dades entre processos.

3.2.1.3 Llibreries i serveis

Una capa per sobre del nucli es troben les diferents llibreries nadiues del sistema operatiu. Aquestes llibreries proveeixen, entre sí mateixes i a capes superiors del sistema, diferents funcionalitats. Entre d'altres es troben les següents:

- **Llibreria de C del sistema:** una implementació de la llibreria estàndard de C per Android, anomenada Bionic, que permet l'ús d'operacions tals com poden ser el tractament de frases, computacions matemàtiques, processament d'entrada/sortida o assignació de memòria.
- **SQLite:** potent i lleuger motor de bases de dades relacionals per a poder ser utilitzat per qualsevol aplicació.
- **Llibreries multimèdia:** aquestes llibreries, basades en PacketVideo's OpenCORE, suporta la reproducció i generació de la major part dels formats d'àudio, vídeo i imatges.
- **Gestor de pantalla:** permet l'accés al sistema de pantalla i genera capes gràfiques en 2D i 3D.
- **SGL:** motor de generació de capes 2D.
- **Llibreries 3D:** una implementació d'OpenGL per a la utilització de l'acceleració per maquinari o per programari, aquesta última molt optimitzada.
- **LibWebCore:** modern motor de navegació web.
- **FreeType:** renderització de mapes de bits i fonts.

Al mateix nivell poden trobar-se els dos sistemes principals d'Android que s'encarreguen de la creació i gestió de tots els serveis que proporciona l'API d'Android:

- **Servei del sistema:** és l'encarregat de gestionar la creació dels diferents proveïdors de serveis que poden trobar-se a l'API d'Android.
- **Gestor dels serveis:** és l'encarregat de la realització de crides a funcions entre els diferents proveïdors de serveis i les aplicacions. També s'encarrega de l'intercanvi de dades entre els dos. La comunicació entre les dos parts s'ha d'establir a través del Binder.

3.2.1.4 Dalvik i Zygote

A més de les diferents llibreries del sistema operatiu, també cap explicar que en aquesta capa hi és el sistema on s'executen les diferents aplicacions. Aquest sistema és Dalvik, una implementació d'una màquina virtual de Java basada en registres, realitzada per Google i optimitzada per a dispositius intel·ligents. Cada aplicació d'Android executa la seva pròpia instància de Dalvik amb executables .dex, optimitzats per a un millor ús de la memòria. La màquina virtual pot córrer classes compilades en Java transformades prèviament en .dex amb l'eina dx.

També s'ha de tindre en compte la funció del servei Zygote. Aquest servei carrega una instància de Dalvik durant l'inici del sistema i sobre aquesta els diferents tipus de proveïdors de serveis per mitjà del servei del sistema. Quan una aplicació s'inicia es realitza un clon d'aquesta instància i es referencien les zones de memòria assignades als diferents proveïdors de serveis. Si una aplicació requereix escriptura sobre aquestes zones es realitzarà una copia d'aquestes per a la seva utilització local.

3.2.1.5 Marc d'applicacions

Tot programador a Android pot aprofitar les diferents funcionalitats que posa a disposició l'API d'Android. Aquesta capa permet la programació d'applicacions i d'altres components d'una forma senzilla que permet que siguin reutilitzables per altres aplicacions, simplificant així la reutilització de programari. Un programador pot utilitzar fàcilment recursos del maquinari, accedir-hi a la informació de localització geogràfica, córrer processos en segon pla, configurar alarmes, afegir-ne notificacions, etc. Depenent de la versió d'Android es poden utilitzar els diferents mètodes que existeixen.

Per a possibilitar aquest funcionament existeixen serveis i sistemes com poden ser:

- Les diferents vistes que poden ser utilitzades per a la creació d'una aplicació, com llistes, gralles, caixes de text, botons, etc.
- Proveïdors de contingut que habiliten a les aplicacions l'accés a dades d'altres aplicacions o la compartició de les seves pròpies.
- El gestor de recursos, que proveeixen accés a recursos com texts, gràfics i fitxers de capes per a la utilització d'aquests segons la configuració i utilització del dispositiu. Per exemple, es poden utilitzar diferents imatges quan el dispositiu és en posició vertical o horitzontal.
- El gestor de notificacions, que habilita la possibilitat a les aplicacions de mostrar alertes configurades a la barra d'estat.
- Un gestor d'activitat, que gestiona el cicle de vida de les activitats de les aplicacions i proveeix una forma de funcionament comuna.

3.2.1.6 Aplicacions

Al sistema Android hi han diverses aplicacions d'usuari escrites en Java. Quan un programador crea una aplicació ha de generar el paquet apk (*Android package*), el qual cont l'aplicació i els recursos necessaris, per a que un usuari pugui instal·lar fàcilment aquest al seu dispositiu. Per a que una aplicació convisqui al sistema amb d'altres es compleixen els següents punts per a cadascuna:

- A cada aplicació se li assigna un usuari del sistema.
- Cada recurs de l'aplicació té establerts els permisos per a que únicament aquesta pugui accedir-hi.
- Cada aplicació corre el seu procés a una instància pròpia de la màquina virtual Dalvik, independentment de la resta.
- Per defecte, cada aplicació corre el seu propi procés. Android inicia el procés quan qualsevol dels components de l'aplicació necessita ser executat, i el tanca quan no hi ha cap en funcionament o quan el sistema necessita obtindre memòria lliure.

Per a la compartició de dades entre diferents aplicacions existeixen dos mètodes diferents:

- És possible que dos aplicacions comparteixin el mateix usuari, podent-hi arribar a utilitzar-ne el mateix procés i la mateixa instància de la màquina virtual.
- L'altra opció és l'establiment de privilegis per part de l'usuari al ser instal·lada l'aplicació. Permisos com l'accés als contactes, als missatges SMS, a la targeta SD, etc son un exemple.

Una aplicació pot estar formada per diferents components que poden ser complementaris entre ells o actuar de manera independent. Cadascun dels següents tipus tenen una funció associada:

- **Activitats**: son una simple interfície d'usuari per a la interactivitat amb aquest. En una mateixa aplicació pot haver-hi una activitat per a cada una de les diferents accions que pot realitzar un usuari. Per exemple, en una aplicació client de correu pot haver-hi una activitat per a llegir el correu, una altre per a redactar-ne, i una altre per a llistar-ne.
- **Serveis**: és un component que corre en segon pla per a executar tasques d'una llarga durada. Per exemple, una aplicació pot realitzar descàrregues en segon pla. Aquest component pot ser arrencat i utilitzat a partir d'altres.
- **Proveïdors de contingut**: gestionen un conjunt de dades compartides de l'aplicació. Les dades poden ser desades al sistema de fitxers, en una base de dades SQLite, al web, o qualsevol altre localització. A través d'aquest component altres aplicacions poden

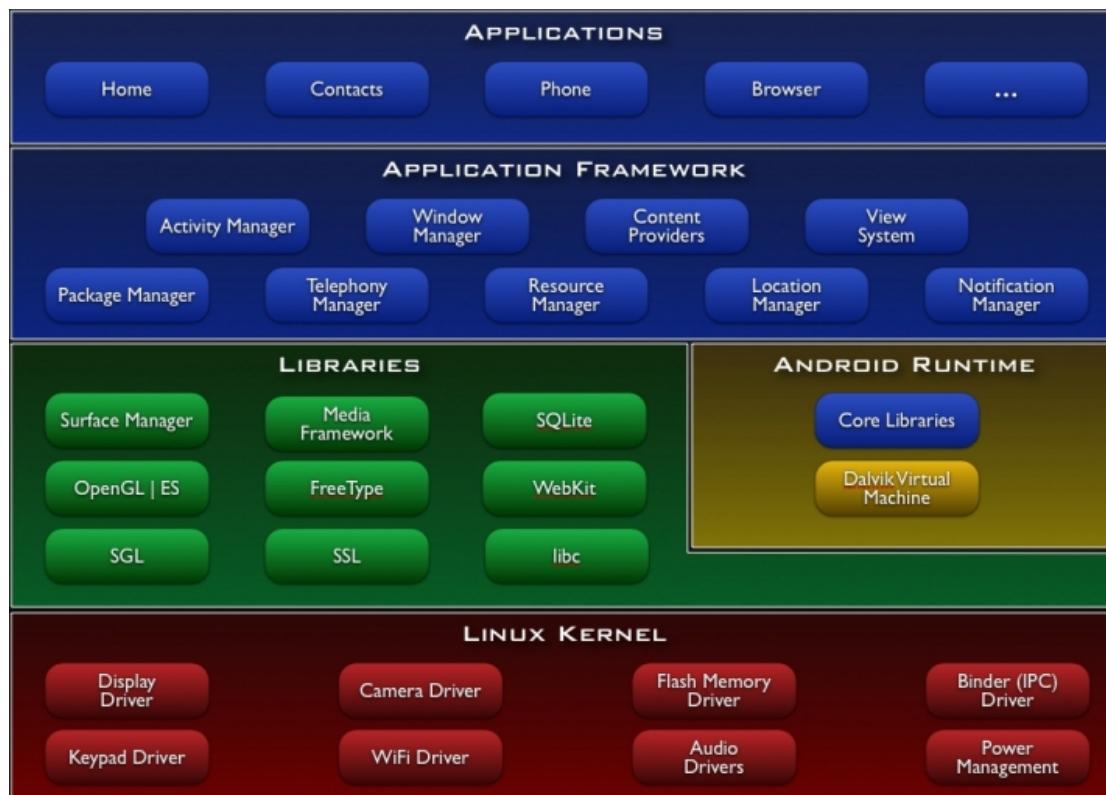


Figura 14: Arquitectura d'Android

accedir-hi a aquestes dades. Els accessos a les dades han d'haver-hi sigut acceptats per l'usuari.

- **Receptors de difusió múltiple:** és un component que serveix per a tractar els anuncis de difusió múltiple emesos per el sistema o per aplicacions. Aquests components acostumen a anar acompanyats d'una notificació a la barra de notificacions de l'usuari.

Un aspecte concret del sistema Android és que qualsevol aplicació pot iniciar qualsevol component d'una altre, amb permís de l'usuari. Aquesta és una més de les mesures per a reaprofitar el programari ja fet. Per exemple, des d'una aplicació es pot realitzar una fotografia amb una activitat d'una altre aplicació sense haver-ne de realitzar la programació d'aquesta. Quan una aplicació inicia un component d'una altre, s'inicia el procés corresponent a aquesta aplicació i les classes necessàries per a la utilització d'aquest component. Les aplicacions d'Android no tenen un component principal com s'acostuma a realitzar en la resta de programacions.

3.2.2 Programació de codi nadiu

Per a la construcció i compilació de codi per a la arquitectura ARM des d'una altre arquitectura des d'on es treballa en una aplicació, com pot ser i386 o AMD64, s'ha d'utilitzar un compilador creuat. El codi font d'Android té, entre d'altres, un set de binaris per a realitzar les compilacions per a ARM des d'un sistema GNU/Linux x86. Aquests compiladors permeten construir programaris nadius en C/C++ com poden ser mòduls.

Si es necessités, el codi d'Android també proveeix de bionic, llibreria estàndard de C comentada anteriorment, que seria necessari compilar-la per a realitzar l'enllaç a aplicacions nadiues. Per a treballar de forma senzilla a l'hora d'enllaçar la llibreria es pot utilitzar l'script <http://plausible.org/andy/agcc>. Per aquesta possibilitat és més recomanable la utilització del *Native Development Kit* (NDK). Aquest kit permet la creació d'aplicacions utilitzant les diferents llibreries nadiues del sistema operatiu comentades anteriorment, formant un paquet apk autoinstal·lable. Si s'utilitza aquesta opció les aplicacions creades estaran subjectes a les mateixes condicions de seguretat que les aplicacions creades per a córrer sobre Dalvik.

El codi nadiu acostuma a ser més eficient però més complex.

3.2.3 Programació d'aplicacions per a Dalvik

Per a la programació d'aplicacions per a córrer a la màquina virtual Dalvik s'ha d'utilitzar el kit de desenvolupament de programari, o *Software Development Kit* (SDK) i un entorn de desenvolupament.

3.2.3.1 Android SDK r18

Android SDK son les sigles d'Android Software Development Kit, un kit de desenvolupament amb el que es pot des de desenvolupar aplicacions fins a executar-les en un emulador del sistema Android a la versió que es necessiti. Aquest kit compta amb aplicacions

com adb, per a la comunicació via USB amb el dispositiu, documentació, exemples, o llibreries i imatges corresponents a una versió del sistema.

3.2.3.2 Eclipse Indigo 3.7.2

Per a aquest projecte s'ha triat Eclipse com a entorn integrat de desenvolupament, o *Integrated Development Environment* (IDE), ja que l'SDK està preparat per a integrar-se amb aquest amb el plugin ADT d'Android. ADT augmenta les capacitats d'Eclipse per a permetre una ràpida creació de projectes d'aplicacions, crear interfícies gràfiques, afegir paquets basats en el marc d'aplicacions d'Android, depurar aplicacions, exportar aplicacions amb o sense signatura, etc.

3.2.4 LabJackPython

Per al desenvolupament del programari necessari per a la monitorització del voltatge obtingut per al dispositiu LabJack existeix un modul per a la comunicació amb aquest mitjançant aplicacions programades en Python. A Linux, Windows o Mac és necessari la instal·lació del controlador corresponent per a la comunicació amb el dispositiu. A la guia del LabJack U3 es troben explicats els diferents mètodes per a la programació d'aplicacions que configurin el dispositiu i s'estableixi el flux de dades entre el LabJack i l'ordinador.

Capítol 4. Entorn de mesura del consum

En aquest capítol s'explicaran cronològicament els passos seguits per a la obtenció d'un entorn de mesura del consum energètic. Primerament es veurà quina era la idea i els coneixements inicials, després es passarà a explicar el disseny tant de la circuiteria com del programari necessari, i finalment s'explicarà la implementació de cada part.

4.1 Estudi preví

Inicialment, la idea d'obtindre el consum energètic del terminal mòbil es va extreure dels diferents estudis relacionats en els que s'utilitzaven diferents tècniques però amb una certa coherència entre aquests. La idea base era la mesura d'una banda, del corrent instantani utilitzat per el terminal mòbil, i d'altra banda la mesura de la tensió subministrada per la bateria. El corrent al terminal canvia ja que els diferents components s'activen i es desactiven, s'utilitzen de diferent formes, i tenen una certa afectació de l'entorn. En quant a la tensió de la bateria, per el que s'ha arribat a estudiar, se sap que amb la bateria en descàrrega va disminuint lentament fins a arribar a una tensió de tall on la bateria ja no pot subministrar el voltatge necessari al terminal mòbil. Una vegada es tenien les dos mesures, es multiplicaven aquestes per a obtindre la potència instantània.

4.2 Disseny

En aquest apartat s'explicaran els diferents dissenys que s'han dut a terme per a la circuiteria i per al programari.

4.2.1 Maquinari

A continuació es mostraran cronològicament els dos models que es van dissenyar per a obtindre el consum del corrent utilitzat per el terminal mòbil i després el circuit necessari per a la mesura de la tensió de sortida de la bateria. El primer model de mesura del corrent del terminal mòbil no va resultar satisfactori per això s'expliquen cadascun dels dos. Més endavant, a la implementació es comentarà perquè no va funcionar el primer disseny.

4.2.1.1 Circuit amb sensor AMPLOC AMP50

Una alternativa que es va considerar per a obtindre una mesura del corrent utilitzat per el terminal mòbil consisteix en la utilització d'un sensor AMPLOC AMP50. Aquest sensor permet obtindre una mesura fiable i molt estable del consum sense haver-ne d'interposar cap element al camí del flux del corrent. La idea consistia en enrotllar un cable conductor del corrent de sortida del terminal mòbil al sensor formant una bobina que crearia un camp magnètic degut al flux del corrent que passa per el conductor. Aquest camp magnètic crearia una variació al flux d'electrons que passen a través del sensor, degudament alimentat a 5 volts, que provocaria una diferència de tensió, això es coneix com a efecte Hall. A la següent figura es pot observar el seu funcionament:

A la imatge A, una càrrega negativa apareix a la vora superior del sensor Hall (simbolitzada amb el color blau), i una positiva a la vora inferior (color vermell). A B i C, el camp elèctric o el magnètic estan invertits, causant que la polaritat s'inverteixi. Invertir tant el corrent com el camp magnètic (imatge D) causa que la sonda assumeixi de nou una càrrega negativa a la part superior.

Llegenda:

1. Electrons
2. Sensor Hall
3. Imants
4. Camp magnètic
5. Font d'energia

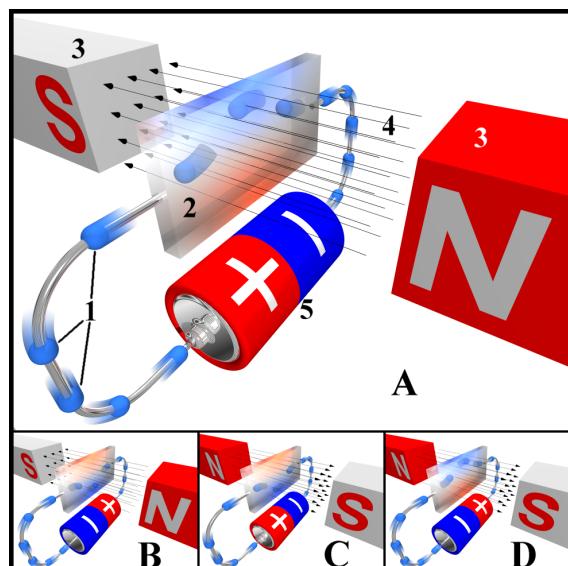


Figura 15: Efecte Hall

Degut a que era un sensor de mesura de fins a 50 amperes, i a que les mesures a realitzar eren d'un amperatge molt inferior (0 – 1,5 amperes), es van haver de donar fins a 50 voltes a la bobina per a obtindre una sortida de mesura adient. Això és degut a que el camp magnètic augmenta a cada volta que es realitza a la bobina.

Tenint en compte els límits de mesura de tensió al LabJack, de 0 a 2,44 volts, es va escollir que el flux del corrent sigues negatiu, ja que degut a això la tensió de sortida del sensor disminuiria a partir del valor inicial de tensió, uns 2,5 volts (tensió d'alimentació / 2) sense corrent induït. A la següent figura extreta del full de dades del sensor es pot veure la relació entre el corrent sentit i la tensió de sortida:

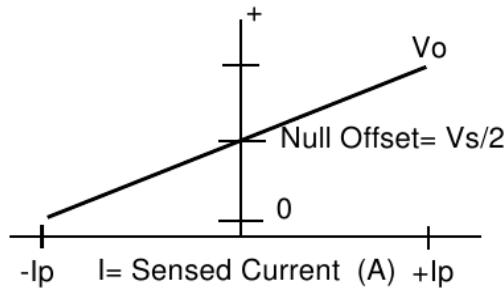


Figura 16: Relació entre corrent sentit i tensió de sortida (AMP50)

Així doncs, solament faltava ajustar la tensió amb un divisor de tensió a partir de 3 resistències per a obtindre una sortida de tensió per sota del límit de mesura del LabJack. El disseny del circuit és el següent:

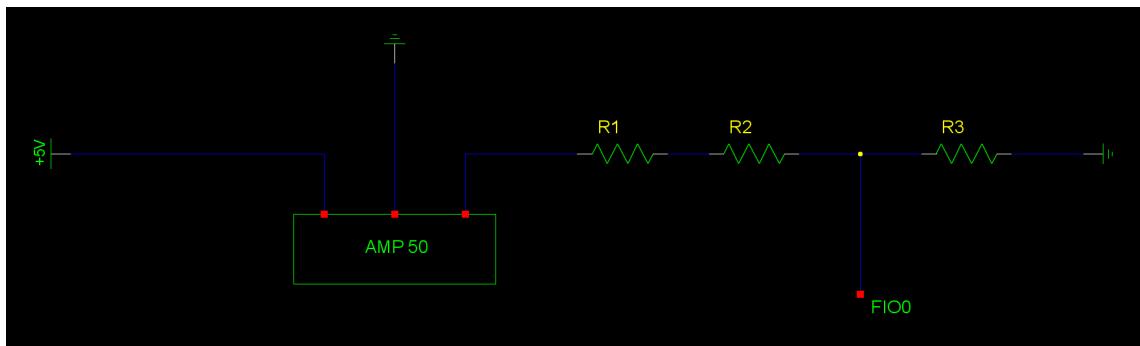


Figura 17: Circuit amb sensor AMPLOC AMP50

Més endavant, a la implementació es comentarà perquè es va descartar aquesta alternativa.

4.2.1.2 Circuit amb xip TI INA 169

A conseqüència de que el primer model no va resultat satisfactori per a l'objectiu de mesura del corrent es va optar per utilitzar un model més complex amb un xip TI INA 169. Aquest xip mesura una diferència de tensió a partir dels voltatges obtinguts a cada born d'una resistència externa, coneguda com resistència de *shunt*, i extreu un corrent sortint que pot ser mesurat a partir de la disposició d'una resistència externa, coneguda com a resistència de *load*, tal i com s'indica en el full de dades del xip amb la fórmula $V_{SORTIDA} = I_S R_S R_L / 1K\Omega$. A partir d'aquesta fórmula es poden calcular els valors de les resistències externes segons els límits de

la tensió de sortida. En aquest cas el límit ve fixat per la tensió màxima que pot mesurar el LabJack. L'esquema del xip i de les resistències externes és el següent:

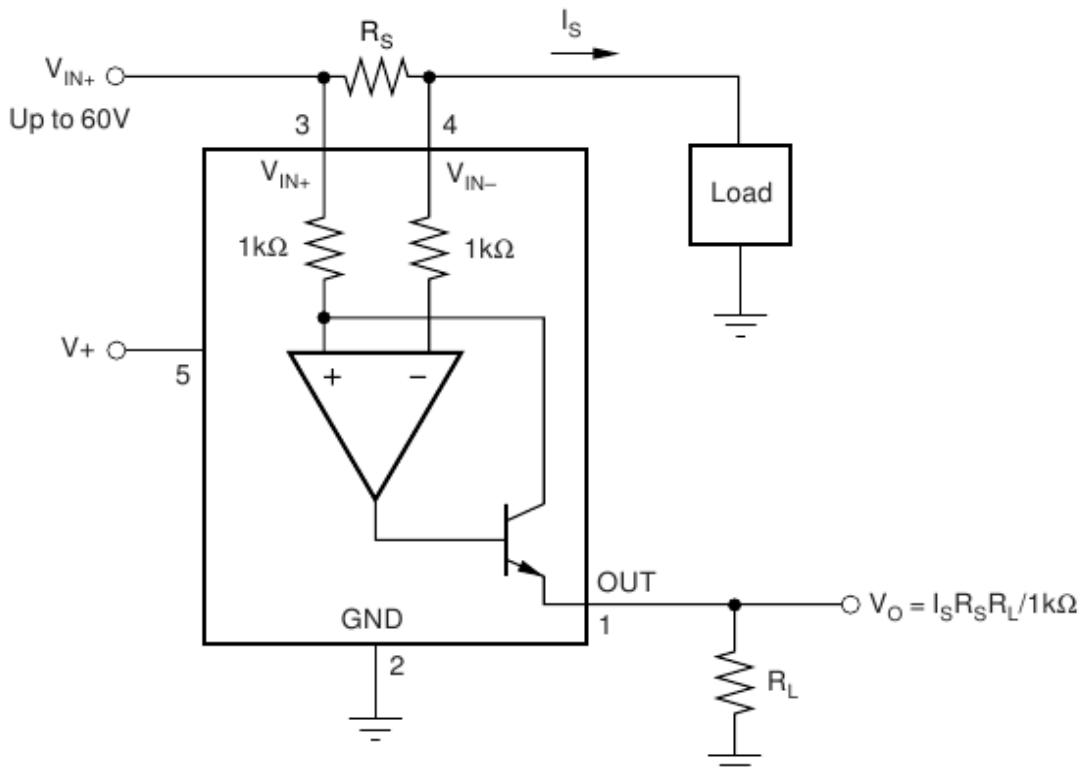


Figura 18: Esquema xip TI INA 169

A més de la disposició de les pertinents resistències externes es va afegir al disseny del circuit un condensador per a filtrar i estabilitzar la sortida (aquest xip té una sortida menys estable que el sensor comentat al model anterior). L'esquema resultant del circuit mesurador del corrent utilitzat per el terminal mòbil va ésser el següent:

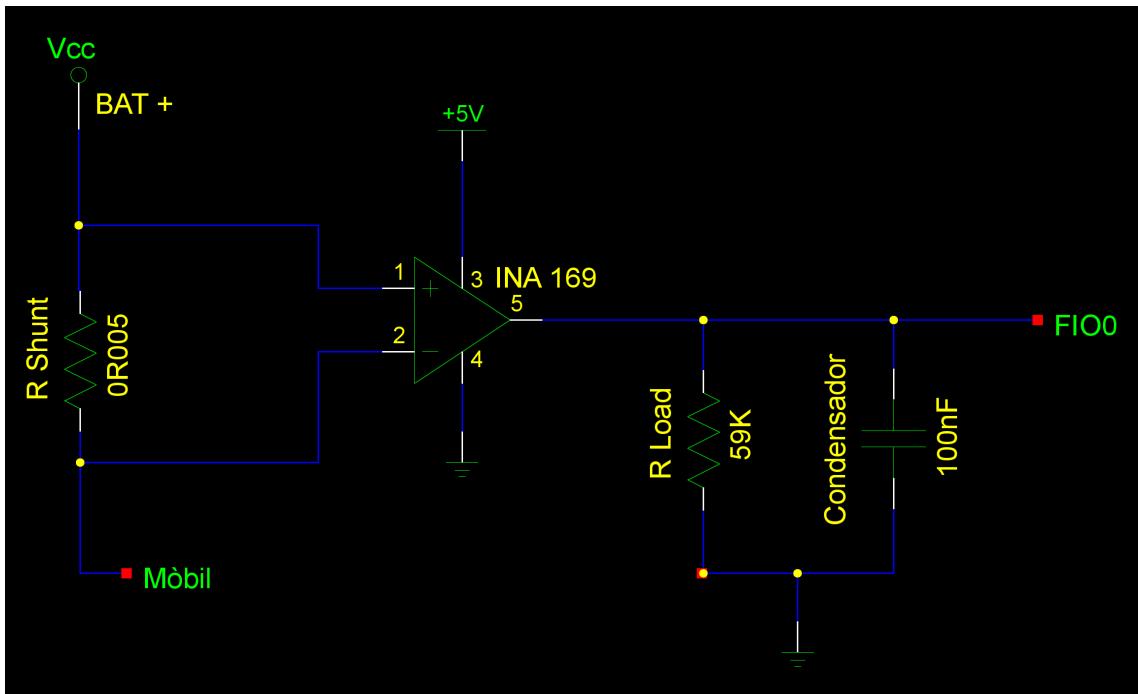


Figura 19: Circuit amb sensor TI INA 169

4.2.1.3 Circuit per a la mesura del voltatge de la bateria

A més de la mesura del corrent utilitzat per el terminal mòbil, per a obtindre la potència instantània, es necessita obtindre la tensió subministrada per la bateria. Aquesta tensió és variable en el temps degut a la descàrrega de la bateria per això és necessari saber-ne el voltatge en cada moment.

Per a la mesura del voltatge de la bateria es va utilitzar simplement un circuit divisor de tensió degut al límit que el LabJack té per a mesurar la tensió d'entrada. En aquest cas es va escollir el port FIO1 per a l'entrada del senyal analògic. L'esquema utilitzat és el següent:

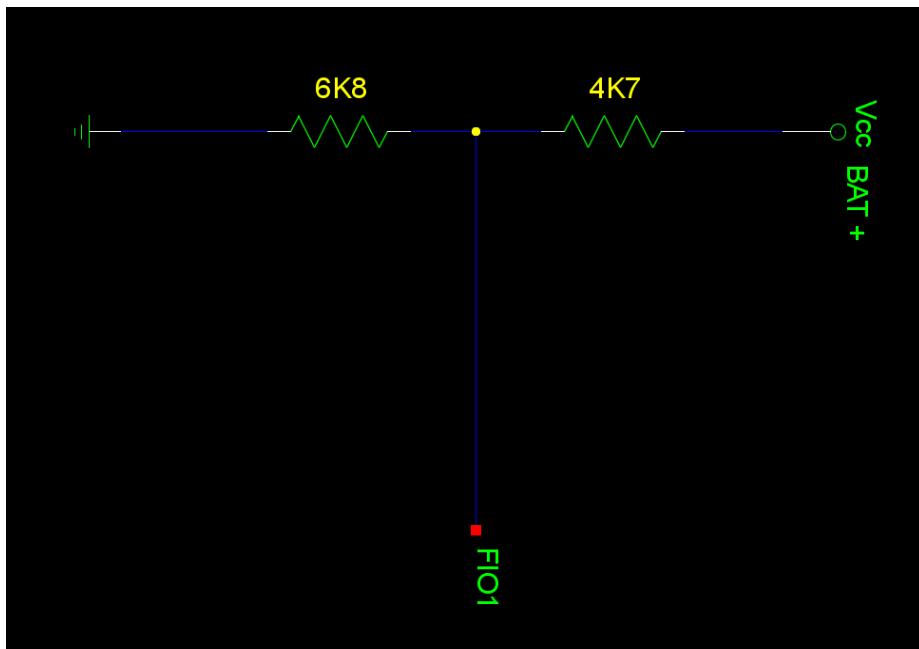


Figura 20: Circuit divisor de tensió per a la mesura del voltatge de la bateria

4.2.2 Programari

Tot circuit dissenyat i comentat abans no tindria sentit sense un programari que obtingui les dades des del LabJack. Aquest dispositiu té la possibilitat de realitzar programari orientat a objectes amb Python utilitzant els seus propis objectes i mètodes. Per a realitzar la mesura del consum era necessari realitzar els següent passos:

1. Establir la connexió amb el LabJack: abans de res s'ha de realitzar la creació d'un nou objecte del tipus U3, tipus concret per al model LabJack U3-LV i U3-LH, amb el que s'establirà la connexió amb el LabJack i haurà de ser utilitzat per a la execució dels diferents mètodes.
2. Configurar els ports FIO0 i FIO1: el fet de configurar les entrades és necessari ja que aquestes es poden definir com analògiques o digitals, els senyals poden ser d'entrada o sortida al LabJack, i es pot definir el permís d'escriptura sobre aquests. També es pot configurar com obtindre les mostres amb entrades analògiques, més ràpidament i amb més soroll o a la inversa. Les entrades FIO0, entrada del corrent provinent del terminal mòbil, i FIO1, entrada del corrent provinent de la bateria, hauran de ser configurades com a entrades analògiques, amb capacitat d'escriptura i amb la major precisió. Aquesta última opció va ser seleccionada a posteriori al veure que el número de mostres per segons, aproximadament 20, era suficientment gran com per a permetre obtindre un resultat fiable.
3. Obtindre les dades dels dos ports: una vegada feta la configuració, es podran obtindre les dades mesurades per el LabJack. Aquestes dades son obtingudes primerament en binari, amb una resolució de 12 bits, i després convertides a decimal per mitjà d'un

mètode concret. Aquest mètode s'haurà de realitzar per a cadascun dels ports a mesurar per a emmagatzemar la tensió mesurada a cadascun.

Abans d'explicar el següent punt és necessari explicar com es tractaran les mesures obtingudes. El LabJack realitza la mesura dels dos voltatges, provinents del terminal mòbil i de la bateria, però aquests no son exactament els mateixos que els originals, sinó que han sigut disminuïts per a poder ser mesurats amb el LabJack. És per aquesta raó que s'ha d'establir una correlació entre el corrent i la tensió proveïts per una font del laboratori que farà de terminal amb les mesures obtingudes al LabJack. D'aquesta forma es podrà obtindre la recta de regressió que faciliti la conversió d'una mesura en un altre valor al programari. En el cas del terminal mòbil, s'haurà d'obtindre la correlació entre el voltatge mesurat per el LabJack i el corrent proveït per la font ja que el que interessa és obtindre la mesura del corrent. Amb la bateria, és necessari obtindre la correlació entre la tensió proveïda per la font i la tensió mesurada per el LabJack. Aquest procediment serà detallat a la secció d'implementació.

4. Calcular la potència instantània: havent obtingut les dues tensions provinents de cada port es podran realitzar les conversions necessàries entre els voltatges mesurats i les variables que es volen obtindre per a cada cas, corrent per el terminal mòbil i voltatge per la bateria. A partir d'aquests dos variables es podrà obtindre la potència instantània amb la coneguda fórmula $P = V * I$.
5. Treure per pantalla les dades per al seu posterior tractament: després d'obtindre la potència instantània es podrà mostrar aquesta per pantalla per a redirigir la sortida a un fitxer de text pla. Les dades podran ser tractades posteriorment en un full de càlcul. També serà necessari acompañar cada dada de potència amb una marca de temps en format POSIX time⁶ i amb una precisió en milisegons per a poder obtindre posteriorment amb un full de càlcul el consum en watts * hora.

Els punts del 3 al 5 es repetiran indefinidament al programari per a obtindre contínuament mostres. La selecció d'un bucle infinit és degut a que les diferents mostres que s'han de mesurar requeriran molta flexibilitat a l'hora de definir la finalització d'aquestes, és per això que no s'ha decidit definir un comptador de temps o de mostres. A més, ja que les mostres han de durar un temps prou extens disminueix l'error de precisió de la mostra al haver-se de tancar la seva mesura a mà.

4.3 Implementació

En aquest apartat s'explicaran les implementacions realitzades a partir dels dissenys anteriors que s'han dut a terme per a la circuiteria i per al programari.

⁶ POSIX time és un sistema per a descriure instants de temps, definit com a el nombre de segons que han passat des de la mitja nit UTC de l'1 de Gener de 1970.

4.3.1 Maquinari

A continuació es mostraran cronològicament els dos models que es van implementar per a obtindre el consum del corrent utilitzat per el terminal mòbil a més del circuit necessari per a la mesura de la tensió de sortida de la bateria. També s'explicaran les correlacions realitzades a cadascun. El primer model de mesura del corrent del terminal mòbil no va resultar satisfactori per això s'expliquen cadascun dels dos.

4.3.1.1 Circuit amb sensor AMPLOC AMP50

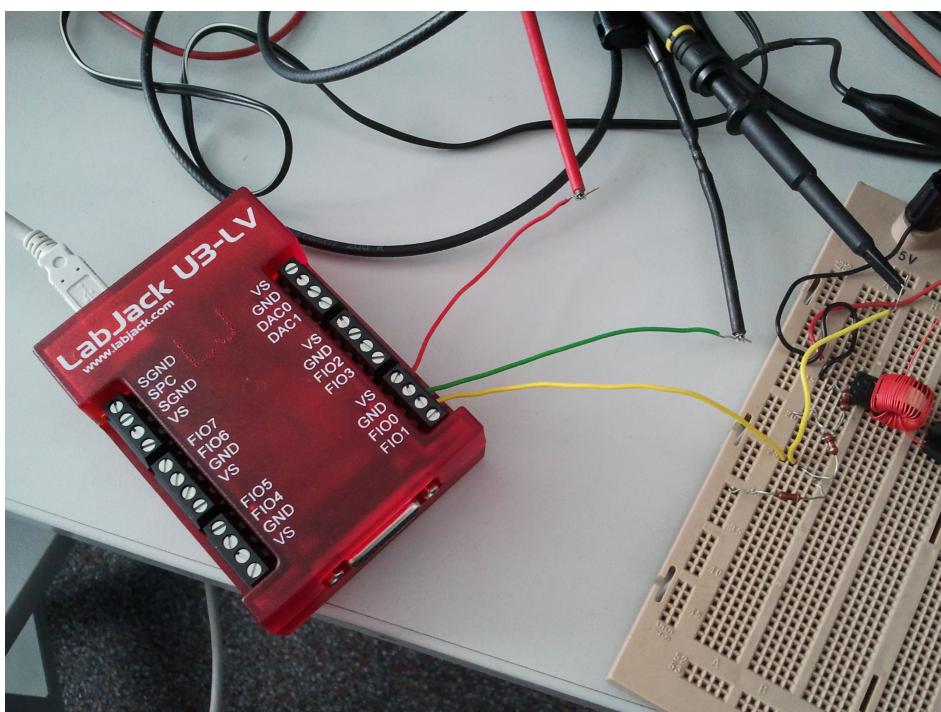


Figura 21: Circuit amb sensor AMP50 (connexions amb Labjack)

El primer model de circuit que es va realitzar per a la mesura del corrent utilitzat per el terminal mòbil va ser el que utilitzava com a component clau el sensor AMPLOC AMP50. Aquest circuit va ser provat sobre una protoboard per a establir la correlació necessària entre el corrent elèctric proveït per la font d'alimentació del laboratori i el consumit per el terminal mòbil. A l'hora d'establir els resultats va sorgir el problema que el terminal mòbil s'apagava quan aquest requeria un pic de corrent. Després d'investigar el problema es va concloure que el problema era degut a que el cable necessari per a l'ampliació del camp magnètic, un metre de cable aproximadament per a fer les 50 voltes al sensor, generava una resistència prou gran com per a provocar una caiguda sobre la tensió del terminal mòbil i fer que aquest no disposés del suficient voltatge com per a funcionar.

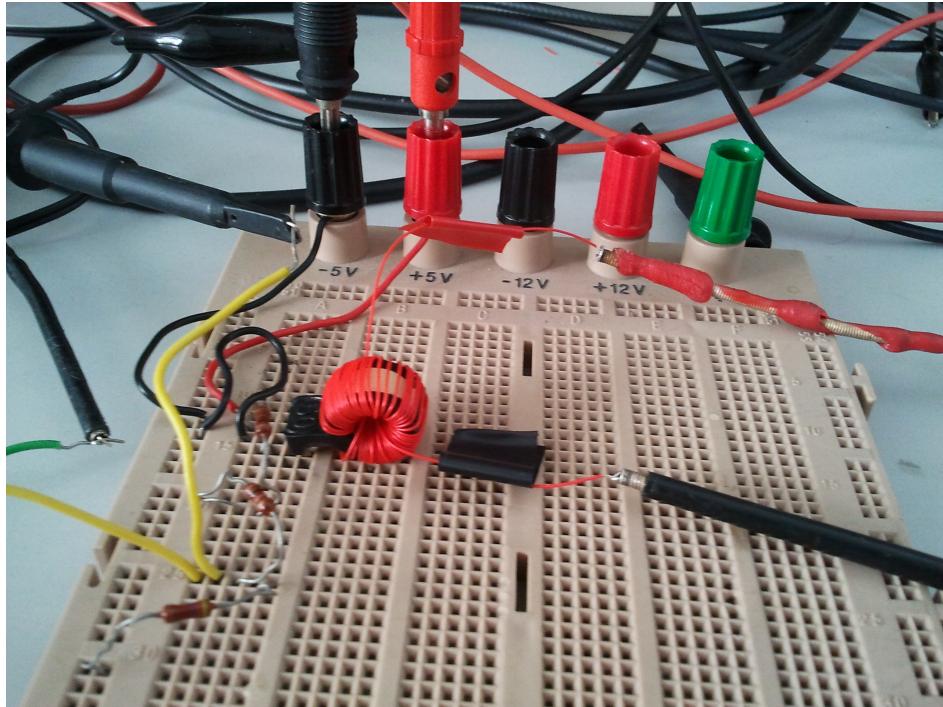


Figura 22: Circuit amb sensor AMP50 (protoboard i connexions amb font)

4.3.1.2 Circuit amb xip TI INA 169

El segon model, l'utilitzat per a les mesures, va ser realitzat reutilitzant una placa utilitzada per a altres finalitats. A continuació s'explicaran les connexions i soldadures realitzades i s'associarà a cadascuna un número que podrà ser identificat a les fotografies intercalades a l'explicació.

A la placa utilitzada es van soldar les connexions i components necessaris per a l'objectiu de mesurar tant el corrent del terminal com la diferència de tensió de la bateria. A les següents figures poden observar-se les dos entrades als ports FIO0 i FIO1 del Labjack, amb els cables groc (1), de sortida de l'INA 169 (9), i taronja (7), provinent del divisor de tensió, respectivament. També pot observar-se el cable verd (2) que alimenta a l'INA 169 amb 5 V provinents del LabJack, i a la seva vegada provinents de la connexió USB amb l'ordinador. En quant a les masses, van soldar-se tant la del terminal mòbil (4) com la del LabJack (3) al mateix punt de la placa. En qualsevol circuit totes les masses han d'anar juntes per a evitar diferències de potencials. Finalment també es poden veure els dos cables vermells (5 i 6) connectats a cada born de la resistència de shunt (10).

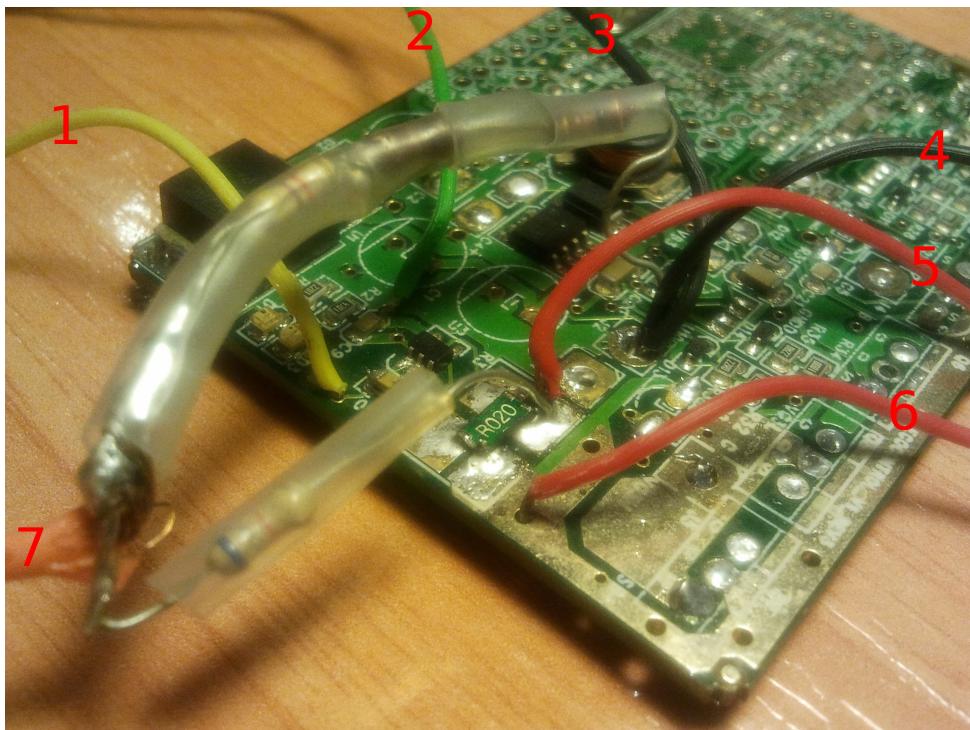


Figura 23: Circuit amb INA 169 (numeració de connexions)

La connexió entre el terminal mòbil i el born positiu de la bateria va aïllar-se amb cinta adhesiva fent que el corrent fluís des de la bateria (5), passés per la resistència de shunt (10) i tornès al terminal mòbil (6), possibilitant així la mesura del corrent utilitzat per el terminal.



Figura 24: Connexions a la bateria

A més del xip INA 169 (9) i la resistència de shunt (10), va soldar-se la resistència de load (8 a sota) i justament a dalt el condensador de filtre (8 a dalt).

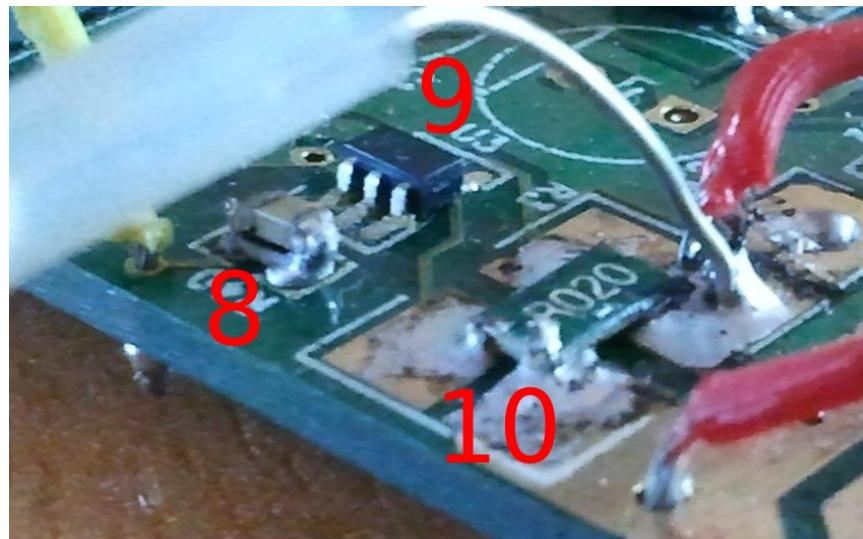


Figura 25: Soldadura del xip INA 169, les resistències de shunt i load, i el condensador de filtre

Una vegada es van tindre totes les connexions i tots els components connectats i soldats es va procedir a obtindre les rectes de regressió per al càlcul del corrent en el terminal mòbil i de la tensió a la bateria. Per a realitzar això va haver-se d'intercanviar el terminal mòbil i la bateria per la font d'alimentació del laboratori i realitzant diferents increments sobre uns rangs definits anotant les diferents mesures preses per el LabJack. També es van obtindre paral·lelament mesures amb l'oscil·loscopi per a comparar aquestes amb les obtingudes per el LabJack i així detectar possibles problemes.

Els següents càlculs son realitzats a partir de les mesures obtingudes a través del port FIO0 del LabJack amb un interval de 0 a 1,5 amperes i amb un increment de 0,1 amperes. Cal destacar que el xip INA 169 necessita un mínim de tensió per a funcionar ja que té un transistor mosfet intern que permet el flux d'electrons a partir d'un mínim voltatge. També cal dir que prèviament es va comprovar amb l'oscil·loscopi quin és l'interval de corrent consumit per el terminal sent aproximadament de 0 a 1,2 amperes aquest.

Prenent les mesures a partir dels 0,2 amperes, el pendent resultant és igual a 1,1430989011 i l'ordenada a l'origen és igual a -0,0882769231. Això forma la següent recta de regressió:

$$V_{LABJACK} = 1,1430989011 * A_{MÒBIL} - 0,0882769231$$

Com es necessita obtindre el corrent del terminal mòbil, s'aïllen els amperes a la fórmula resultant:

$$A_{MÒBIL} = (V_{LABJACK} + 0,0882769231) / 1,1430989011$$

Les següents son les mesures preses al laboratori:

<i>Corrent d'entrada (A)</i>	<i>Mesura FIO0 (V)</i>
0	0
0,1	0
0,2	0,157
0,3	0,274
0,4	0,385
0,5	0,461
0,6	0,593
0,7	0,711
0,8	0,807
0,9	0,941
1	1,028
1,1	1,149
1,2	1,271
1,3	1,401
1,4	1,535
1,5	1,654

Taula 3: Mesures del port FIO0 relatives a l'increment del corrent

Al següent gràfic mostra el pendent de les mesures obtingudes:

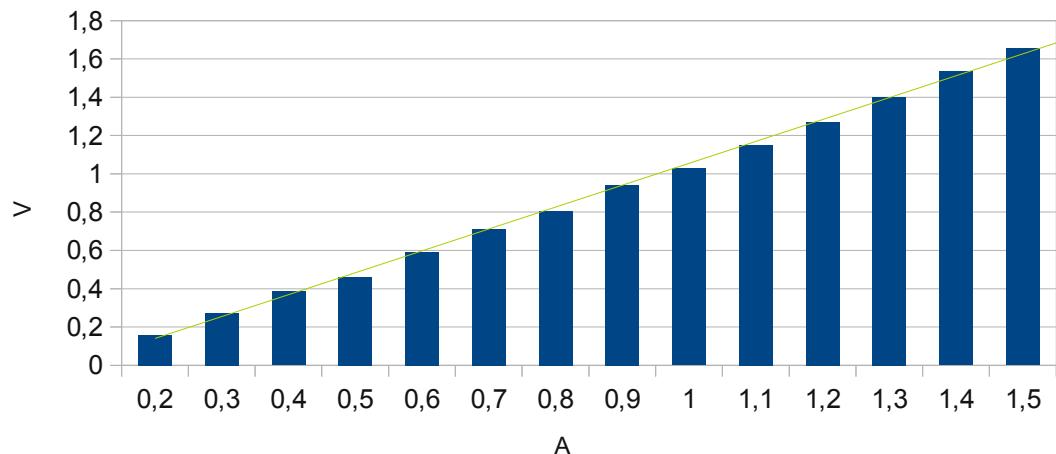


Figura 26: Pendent relatiu a les mesures del port FIO0

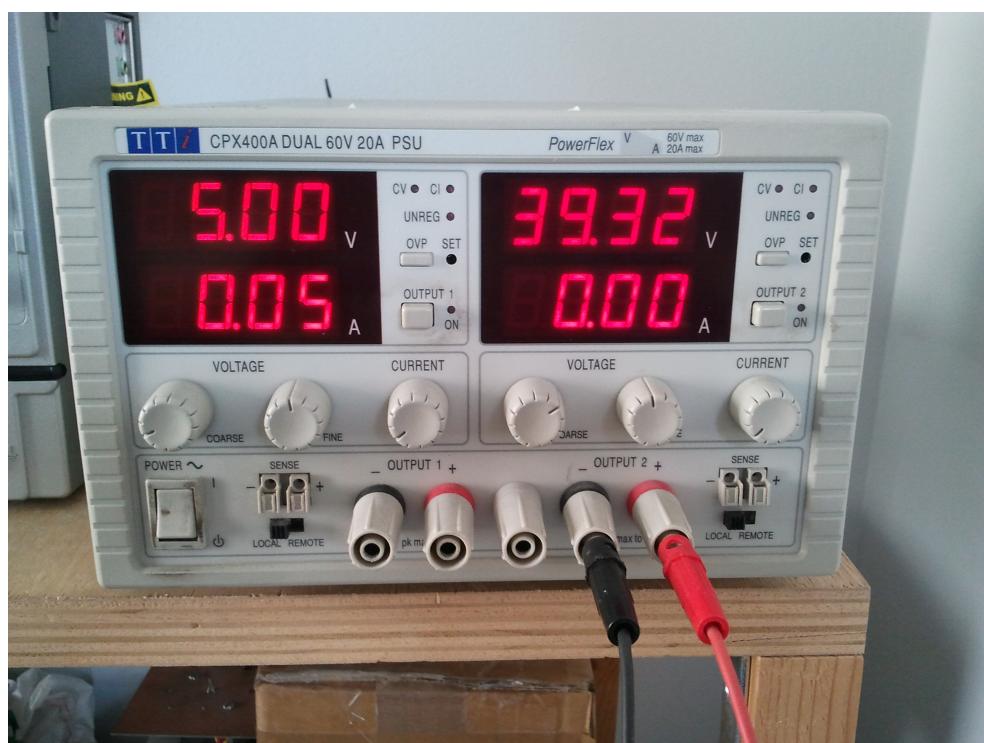


Figura 27: Font d'alimentació

A continuació es mostrerà igualment el càlcul sobre les mesures obtingudes a través del port FIO1 del LabJack. En aquest cas s'utilitzarà l'increment de la tensió de la font com a referència en un interval de 3 a 4 volts i amb un increment de 0,1 volts. El rang de tensions possibles va ser mesurat prèviament amb l'oscil·loscopi.

Prentent les mesures des dels 4,4 volts fins als 3 volts, el pendent resultant és igual a 0,3570357143 i l'ordenada a l'origen és igual a 0,1100345238. Això forma la següent recta de regressió:

$$V_{LABJACK} = 0,3570357143 * V_{BATERIA} + 0,1100345238$$

Com es necessita obtindre la tensió que proporciona la bateria, s'aïlla aquesta resultant la fórmula següent:

$$V_{BATERIA} = (V_{LABJACK} - 0,1100345238) / 0,3570357143$$

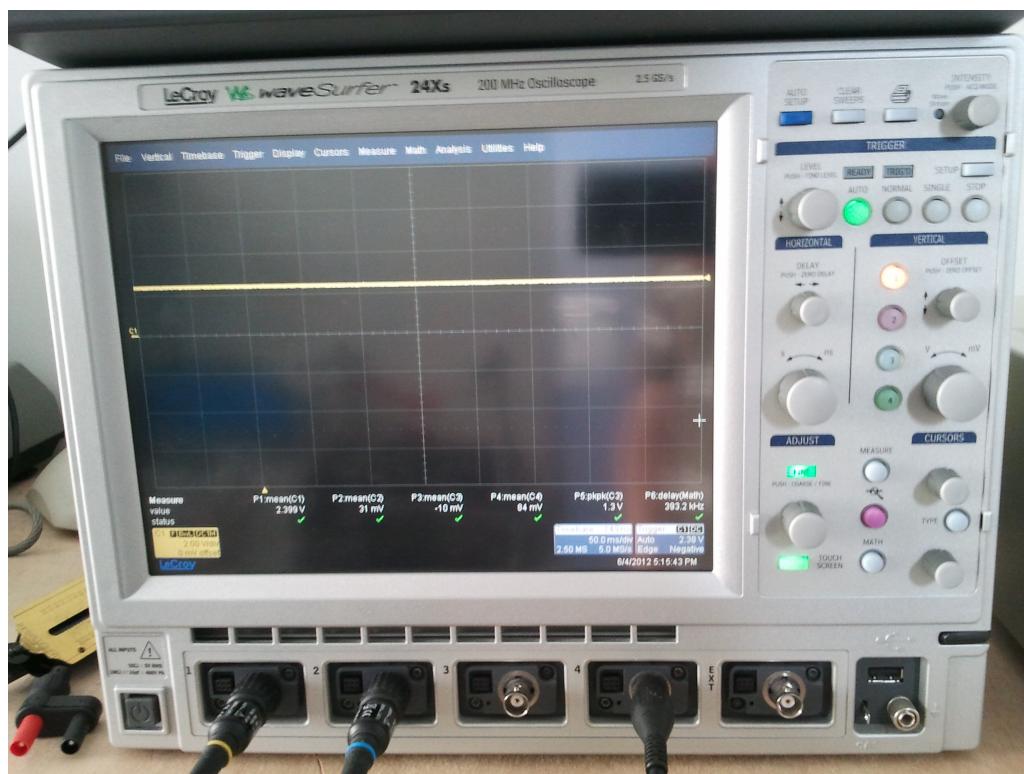


Figura 28: Oscil·loscopi

Les següents son les mesures preses al laboratori:

Tensió d'entrada (B)	Mesura FIO1 (V)
3	1,182
3,1	1,217
3,2	1,252
3,3	1,287
3,4	1,321
3,5	1,364
3,6	1,396
3,7	1,431
3,8	1,465
3,9	1,502
4	1,541
4,1	1,572
4,2	1,609
4,3	1,646
4,4	1,681

Taula 4: Mesures del port FIO1 relatives a l'increment de la tensió

Al següent gràfic mostra el pendent de les mesures obtingudes:

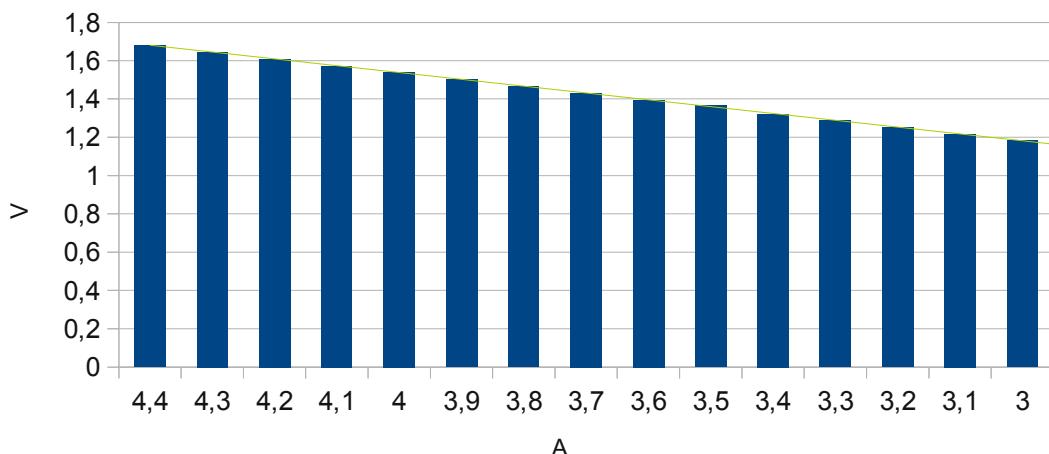


Figura 29: Pendent relatiu a les mesures del port FIO1

4.3.2 Programari

Segons el disseny explicat anteriorment per a la creació del programari que obtingues la potència instantània del terminal mòbil s'ha creat el següent codi que serà explicat part per part:

1. Establir la connexió amb el LabJack: primer de tot el que s'ha fer a l'aplicació és definir la codificació de caràcters, importar les llibreries necessàries per a la comunicació amb el LabJack (u3) i per a obtindre la marca de temps (time), i crear l'objecte u3 amb el que s'utilitzaran els diferents mètodes per a utilitzar el LabJack.

```
# -*- coding: utf-8 -*-
import u3
import sys
import time

# Nou objecte u3
d = u3.U3()
```

Codi 1: Aplicació per a la mesura de la potència instantània (part 1)

2. Configurar els ports FIO0 i FIO1: en aquest pas s'han de definir com actuaran els ports d'entrada al LabJack. Primer de tot es configuraran els dos primers ports (FIO0 i FIO1) com a entrades analògiques (mètode configIO). Després serà necessari definir la capacitat d'escriptura sobre aquests i la direcció del senyal (mètode PortDirWrite). El mètode getFeedback és utilitzat per a tractar les sortides d'una gran part d'operacions d'E/S del LabJack. Amb la configuració anterior solament queda configurar els canals de cadascun dels ports i establir que les mostres siguin el més precises possible (mètode AIN).

```
# Definició de variables (no és necessari declarar-les)
numChannels = 2 # FIO0 i FIO1
quickSample = 0 # 1 -> AnalogToDigital és + ràpid i té + soroll
longSettling = 0 # 1 -> Afegeix delay entre multiplexor i
AnalogToDigital
latestAinValues = [0] * numChannels
potencia = 0

# Definició del tipus d'entrada
fios = 0x3 # b11 (1 analog, 0 digital)
d.configIO( FIOAnalog = fios )

# Definició d'E/S i capacitat d'escriptura (0 = FALSE, 1 = TRUE,
Direction (0 = IN, 1 = OUT)
d.getFeedback(u3.PortDirWrite(Direction = [0, 0, 0], WriteMask = [0, 0, 3]))

feedbackArguments = []

for i in range(numChannels):
    feedbackArguments.append( u3.AIN(i, 31, QuickSample = quickSample,
LongSettling = longSettling ) ) # configuració de la forma d'obtindre les
dades des d'una entrada analògica
```

Codi 2: Aplicació per a la mesura de la potència instantània (part 2)

3. Obtindre les dades dels dos ports, calcular la potència i treure la mostra per pantalla: finalment queda l'execució infinita d'un bucle on s'obtinguin les mesures a partir de la configuració anterior, es realitzi la conversió de bits a decimal per a cada canal (binaryToCalibratedAnalogVoltage), es realitzi el càlcul amb les rectes de regressió calculades anteriorment, i es mostri per pantalla la potència instantànea juntament amb la seva marca de temps en milisegons.

```

while (True):
    results = d.getFeedback( feedbackArguments )
    for j in range(numChannels):
        latestAinValues[j] = d.binaryToCalibratedAnalogVoltage(results[2 + j], isLowVoltage = True, isSingleEnded = True)

    volts_terminal = latestAinValues[0]
    ampers_terminal = ( (volts_terminal + 0.0882769231) / 1.1430989011 )
    volts_bateria = ( latestAinValues[1] - 0.1100345238 ) / 0.3570357143
    potencia = volts_bateria * ampers_terminal
    print '{0:f} {1:f}'.format(time.time(), potencia)

```

Codi 3: Aplicació per a la mesura de la potència instantània (part 3)

4.4 Conclusió

Aquesta part del treball ha sigut la més desconeguda des d'un punt de partida per a mi degut a que tenia pocs coneixements d'electrònica, però amb l'ajuda dels membres d'ESAII Toni Benedito, Carlos Morata i Joan Vidós, he après a realitzar aquestes mesures i a entendre una mica més el mon de l'electrònica.

El resultat ha sigut satisfactori ja que l'entorn de mesura del consum és un entorn reduït, portable, i eficient amb l'objectiu marcat. Aquest entorn ha sigut suficient per a determinar posteriorment canvis significatius en el consum del terminal mòbil degut a certes accions.

Capítol 5. Monitorització del funcionament intern

En aquest capítol es tractarà d'explicar com es van aconseguir registrar les accions que es realitzen en última instància sobre el xip GPS. Aquesta és una part clau del projecte ja que una vegada es puguin registrar aquestes accions es podran establir relacions entre aquestes i les accions realitzades per una aplicació, arribant a tindre la certesa de quines accions executades per una aplicació interfereixen directament sobre el funcionament del xip GPS. També es podrà associar un consum energètic als patrons d'accions que es registrin, d'aquesta forma es podrà estimar l'energia consumida per una aplicació sense tindre accés al seu codi font.

5.1 Estudi previ

A aquesta part del projecte es va arribar amb els coneixements assolits a l'assignatura Projecte de Sistemes Operatius (ProSO). Aquesta assignatura es dividia en dos pràctiques: una consistia en la modificació d'un nucli per a proporcionar funcionalitats a un sistema operatiu, l'altra consistia en la creació d'un controlador carregable dinàmicament al nucli per mitjà d'un mòdul. A aquesta segona pràctica s'havia de realitzar una monitorització de certes crides a sistema per a obtindre una estadística del número de crides realitzades. A continuació s'expliquen uns sèrie de conceptes bàsics que podran ser consultats més endavant si fos necessari per a l'enteniment de les explicacions donades:

5.1.1 Crides a sistema, dispositius i descriptor del fitxer

Una crida al sistema és un mecanisme utilitzat per una aplicació per sol·licitar un servei al sistema operatiu el qual serà executat en mode privilegiat. Una de les seves utilitats és l'execució de funcions bàsiques per a cada dispositiu del sistema, com ara la seva obertura o escriptura. L'estructura d'aquestes crides és la mateixa per a cada dispositiu del sistema, solament canvién els paràmetres utilitzats. Els controladors de cada dispositiu desenvolupen aquestes crides de manera dependent del dispositiu i serà aquesta implementació la que s'executarà en última instància al realitzar la crida a sistema.

Per a l'execució de les crides a sistema sobre un dispositiu en concret s'utilitzen els dispositius de caràcters o els dispositius de blocs (d'ara en endavant dispositius虚拟). Aquests dispositius virtuals son al sistema de fitxers en forma de fitxer al directori /dev i es comuniquen

amb el processador per mitjà de bytes, en el cas dels dispositius de caràcters, o per mitjà de blocs⁷ indivisibles, en el cas dels dispositius de blocs. Cada dispositiu crea el seu dispositiu virtual al carregar-se el seu controlador i associa a aquests les seves crides a sistema. Per a treballar amb un dispositiu primerament serà necessària la seva obertura amb la crida open:

```
int open(const char *ruta_a_dispositiu_virtual, int flags, mode_t mode);
```

Aquesta crida retorna un enter conegut com file descriptor que és un índex d'una estructura pròpia del dispositiu virtual, el file descriptor table, la qual conté la informació necessària per a permetre l'obertura d'aquest dispositiu més d'una vegada al mateix procés o per permetre l'obertura en altres processos i mantindre un funcionament independent a cadascun. A partir d'aquí es podran utilitzar la resta de crides a sistema disponibles sobre el dispositiu indicant per mitjà del file descriptor que és aquesta instància la que es vol utilitzar.

En el cas concret del dispositiu GPS, s'utilitza el dispositiu de caràcters ttyO0.

5.1.2 Capa d'abstracció del maquinari (HAL)

A Android, a diferència de les actuals distribucions que utilitzen Linux com a nucli, s'executa un servei en mode usuari que és l'encarregat de descobrir, enumerar i fer d'intermediari en l'accés al maquinari d'un dispositiu. Aquest servei és conegut com a **HAL** (en anglès *Hardware Abstraction Layer*, capa d'abstracció del maquinari). HAL va sorgir degut a la complexitat que requerien els controladors actuals per a establir definicions comunes del maquinari del qual depenen capes superiors, per a oferir controladors estàndards i per facilitar la portabilitat dels controladors.

En el sistema operatiu Android, els fabricants de maquinari implementen controladors com a llibreries compartides. Aquestes llibreries son realitzades en C/C++ per a ser executades en mode usuari i poden enllaçar llavors la llibreria estàndard de C bionic. D'aquesta forma un controlador pot ser propietari al no haver-ne d'enllaçar parts del nucli per a la seva creació (recordem que bionic té llicència BSD i Linux GPL). Les llibreries creades han de ser dipositades a /vendor/lib/hw o a /system/lib/hw per a que en temps d'execució aquestes puguin ser carregades per HAL mitjançant la llibreria **libhardware**. HAL es comunicarà amb els controladors Linux per mitjà de /dev, /sys o /proc.

5.1.3 UART

Els dispositius avui dia contenen diferents xips i circuiteria que comuniquen cadascun dels microcontroladors interns del dispositiu amb la unitat central de processament. Un d'aquests xips és l'**UART** (en anglès *Universal Asynchronous Receiver/Transmitter*, transmissor/receptor asíncron universal). L'UART converteix les dades a transmetre entre un microcontrolador i la unitat central de processament de paral·lel a sèrie per a ser transmeses a través d'un únic cable per cada sentit, i realitza el procés invers al receptor. Aquest sistema és ideal si no es requereix una alta velocitat o un ús exhaustiu de la comunicació. La transmissió

7 Un bloc és un número de bytes indivisible.

entre els dos extrems es realitza de forma asíncrona però establint unes configuracions de comunicació necessàries per a l'enviament i recepció de les dades. Les dades abans de ser enviades son dipositades en un buffer a l'espera del seu enviament, i al ser rebudes son dipositades també en un buffer a l'espera de la seva adquisició. Un mateix xip UART pot controlar la comunicació de varis dispositius connectats a un mateix bus de dades.

5.2 Intercepció de les crides a sistema al dispositiu de caràcters GPS

La idea d'A continuació s'explicaran els conceptes necessaris per a l'enteniment de les posteriors explicacions de com s'ha realitzat la intercepció de les crides a sistema realitzades sobre el dispositiu de caràcters utilitzat per el controlador GPS.

5.2.1 Camí seguit per una funció de localització

A continuació s'explicaran els passos necessaris per a a realitzar una acció des d'una aplicació d'usuari que requereixi l'accés al maquinari. S'utilitzarà com a exemple la sol·licitud d'actualització de la localització geogràfica cada x segons.

1. Primer de tot s'inicia l'aplicació que utilitzarà el sistema GPS a Dalvik juntament amb les classes necessàries i també les llibreries nadiues que hi precisi.
2. Una de les classes necessàries serà **LocationManager** que demanarà a **binder** la referència al gestor de serveis per a demanar-li després a aquest la referència al servei de localització **LocationManagerService**. Aquest servei haurà estat prèviament registrat al gestor de serveis durant la seva creació per part del servei del sistema o **system_server**. Aquesta gestió realitzada per l'aplicació es deguda a que es necessari la creació d'un **listener** a l'aplicació per a obtindre les actualitzacions de la localització des del servidor de localització.
3. Una vegada es tingui la referència del servei de localització s'executarà l'ordre **requestLocationUpdates** de la classe LocationManager especificant als paràmetres el temps que ha de passar el sistema de localització deshabilitat i com a proveïdor del servei de localització al sistema GPS (recordem que es poden escollir també altres proveïdors). Això es traduirà en una sèrie de crides a sistema sobre binder a través del **proxy** proporcionat per la classe LocationManager.
4. Binder invoca l'execució de l'ordre demanada per l'aplicació al servei de localització a través del seu proxy i aquest a la vegada crida al proveïdor del servei de localització seleccionat a requestLocationUpdates, en aquest cas el del sistema GPS **GpsLocationProvider**.
5. El servei proveïdor del sistema GPS realitza la crida a la llibreria d'abstracció de maquinari **gps omap4.so** passant per la interfície nadiua de Java **_location_GpsLocationProvider.cpp** que proporcionarà el pas del llenguatge Java al llenguatge C/C++ de la llibreria.
6. La llibreria gps omap4.so, desenvolupada per el fabricant del xip GPS i de caràcter propietari, realitza crides a sistema sobre el dispositiu de caràcters **ttyO0**. Aquest dispositiu s'ha creat a partir del controlador estàndard **UART** de Linux ja que el xip GPS està connectat d'aquesta forma a la unitat de processament central. Les crides a

sistema, a més de les comunes, seran les característiques d'aquest sistema de transmissió.

7. La execució cada x segons sol·licitant l'actualització de la localització geogràfica es realitzarà des del servei LocationManager i s'actualitzaren les dades a l'aplicació a través del listener creat al punt 2.

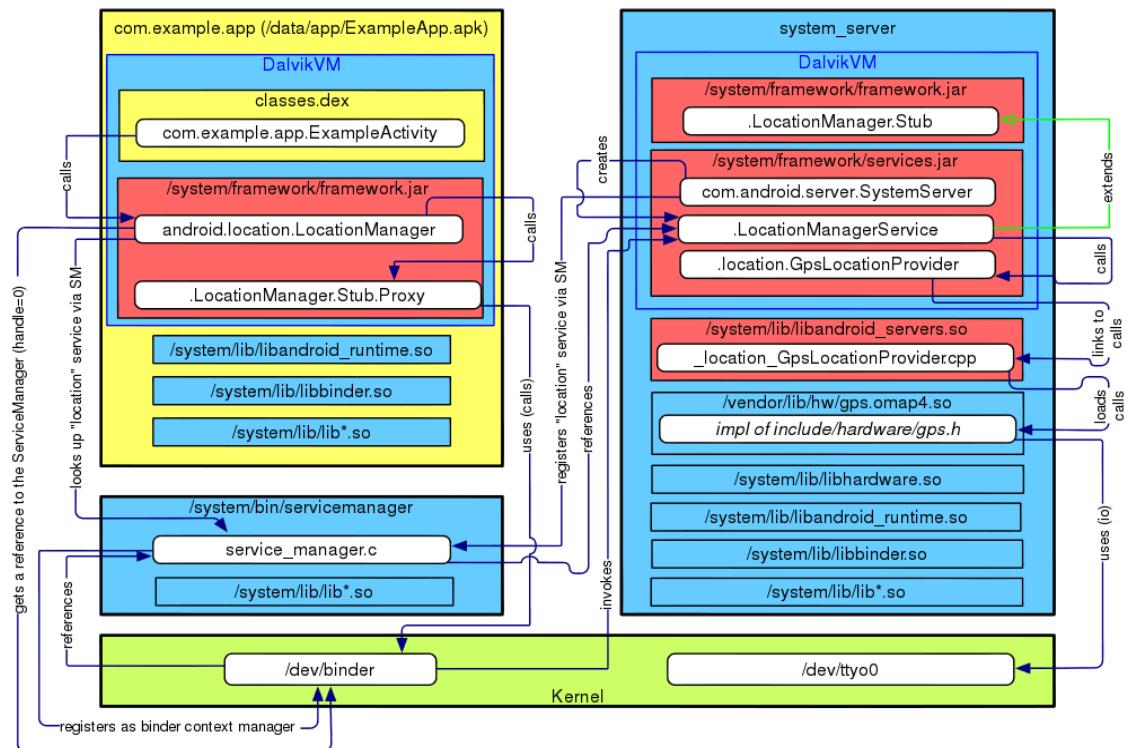


Figura 30: Flux de la localització a Android

5.2.2 Modificació del nucli Linux

Una vegada conegut el camí que recorren les accions executades des del nivell més superior d'Android fins al maquinari es va decidir la intercepció de les crides a sistema realitzades des del controlador GPS gps.omap4.so fins al dispositiu de caràcters UART ttyO0 al ser aquest l'última part del programari que es comunica directament amb el xip GPS. Per a realitzar això era necessari per una banda la utilització d'un mòdul que realitzes les accions pertinents. Els nuclis utilitzats per els diferents dispositius amb Android acostumen a tindre la opció de càrrega de mòduls deshabilitada quedant únicament la opció de configurar el nucli amb aquesta opció i construir-ho per a carregar-lo al dispositiu. A més d'aquest canvi serà necessari la exportació de dos símbols. Un símbol és una variable, una funció o qualsevol altre element definit per un nom i una direcció de memòria que poden ser utilitzats des d'un programari executat en mode privilegiat. En aquest cas concret, existeixen els següents símbols que serà necessari exportar-los afegint el pertinent codi al nucli:

- La llista tty_drivers: aquesta llista conté enllaçats els diferents dispositius sèrie del sistema com ara els dispositius USB o els dispositius UART. Serà necessari l'accés a aquesta llista des del mòdul per a poder obtindre la direcció de memòria de les crides a sistema del dispositiu de caràcters ttyO0.
- El vector de ports UART ui: aquest vector conté tots els ports UART (un per cada element gestionat) i per a cadascun les crides a sistema concretes per al funcionament d'aquest sistema de comunicació.

A continuació es mostrerà el procés realitzat per a descarregar, modificar, configurar, compilar, construir i instal·lar un nucli per al dispositiu utilitzat per aquest projecte:

5.2.2.1 Obtenció de les fonts del nucli

Primer de tot es van descarregar les fonts del nucli. Hi han diferents versions del nucli Linux disponibles per a la seva descàrrega des de la pàgina web d'Android <http://android.googlesource.com/>. Segons el fabricant de la unitat de processament central o el del dispositiu final els nuclis porten diferents configuracions i modificacions. Les disponibles actualment son common, goldfish, msm, omap, samsung i tegra. Per al dispositiu seleccionat per al projecte es va necessitar el nucli omap de Texas Instruments ja que la seva CPU és una TI OMAP 4460. Per a la seva descàrrega s'ha d'executar la següent comanda:

```
$ git clone https://android.googlesource.com/kernel/omap
```

Una vegada descarregat s'ha d'escollir una versió concreta del nucli entre les diferents que s'ofereixen. Es pot obtindre la versió concreta utilitzada per al dispositiu executant per mitjà d'adb:

```
shell@android:/ # cat /proc/cpuinfo | busybox grep "Hardware"
Hardware : Tuna
```

Després es compara la versió obtinguda amb les versions al repositori. Dintre del directori omap resultant de la descàrrega s'executa:

```
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/android-omap-3.0
  remotes/origin/android-omap-panda-3.0
  remotes/origin/android-omap-tuna-3.0
  remotes/origin/android-omap-tuna-3.0-ics-mr1
  remotes/origin/android-omap-tuna-3.0-mr0
  remotes/origin/android-omap-tuna-3.0-mr0.1
  remotes/origin/linux-omap-3.0
  remotes/origin/master
```

Es selecciona la versió a utilitzar i es realitzen els canvis pertinents sobre la versió base descarregada:

```
$ git checkout -b remotes/origin/android-omap-tuna-3.0-mr0
$ git checkout origin/android-omap-tuna-3.0-mr0
```

5.2.2.2 Modificació, compilació i construcció del nucli

Primer de tot es van haver de modificar els fitxers necessaris per a exportar la llista tty_drivers i el vector ui. Per a la primera es necessari editar el fitxer drivers/tty/tty_io.c i afegir a sota de la declaració de la llista la següent línia:

```
EXPORT_SYMBOL(tty_drivers);
```

Per al vector ui es necessari editar el fitxer drivers/serial/omap-serial.c i afegir a sota de la declaració del vector la següent línia:

```
EXPORT_SYMBOL(ui);
```

Una vegada realitzades les modificacions pertinents es va haver de construir el nucli. Per a construir el nucli primer de tot es necessari realitzar la seva configuració. Per començar s'ha de crear el fitxer .config per a la configuració del nucli:

```
$ export ARCH=arm  
$ make tuna_defconfig
```

La segona acció realitzada és una configuració predefinida per a aquest nucli. La resta de configuracions poden trobar-se a arch/arm/configs. Ara es configurarà el nucli per a que tingui actiu el suport per a mòduls. Primer de tot s'hauran d'exportar les variables d'entorn necessàries per a utilitzar el compilador creuat per a ARM:

```
$ export CCOMPILER=<ruta_a_android>/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-  
$ export CROSS_COMPILE=$CCOMPILER
```

Després es passarà a configurar el nucli amb la interfície ncurses:

```
$ make menuconfig
```

En aquest punt s'ha de revisar que estigui marcada la opció *Enable loadable module support*:

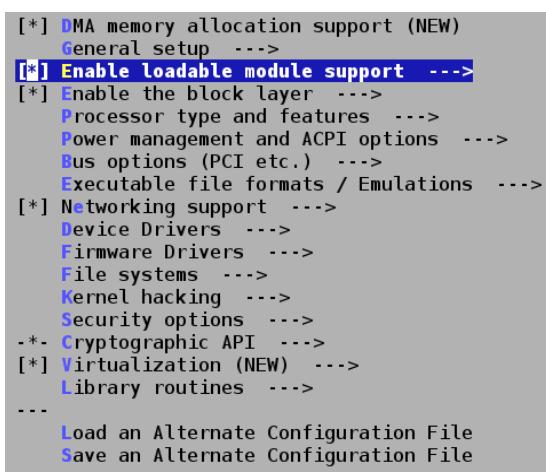


Figura 31: Menú de configuració del nucli

Ara ja es podrà construir el nucli:

```
$ make -j `grep 'processor' /proc/cpuinfo | wc -l`
```

5.2.2.3 Instal·lació del nucli

Finalment per a instal·lar el nucli, des del terminal mòbil es passarà a guardar la imatge de la partició d'arranc per seguretat i perquè és necessari per a el posterior flash d'aquesta ja que conté el nucli.

```
shell@android:/ # cat /dev/block/platform/omap/omap_hsmmc.0/by-name/boot > /sdcard/boot.img
```

Una vegada copiada a l'ordinador es desempaquetarà aquesta:

```
$ unpack-bootimg.pl boot.img
```

Des del mateix directori s'haurà d'empaquetar la imatge d'arranc amb el nou nucli:

```
$ repack-bootimg.pl <ruta_a_kernel>/arch/arm/boot/zImage boot.img-ramdisk imatge-boot.img
```

Després de copiar la nova imatge al terminal mòbil s'haurà de flashejar la partició d'arranc amb aquesta:

```
shell@android:/ # cat /dev/zero > /dev/block/platform/omap/omap_hsmmc.0/by-name/boot
write: No space left on device
shell@android:/ # cat imatge-boot.img > /dev/block/platform/omap/omap_hsmmc.0/by-name/boot
```

A continuació es reiniciarà el terminal mòbil per a que carregui la nova imatge amb el nou nucli. Es podrà comprovar que s'han exportat els símbols amb la comanda:

```
# cat /proc/kallsyms | busybox grep <nom_del_símbol>
```

5.2.3 Segrest de les crides a sistema

Per a la realització d'aquesta comesa era necessari la creació d'un mòdul que intercanvies la direcció de memòria a la que apuntaven els punters de cada crida per la direcció de memòria de funcions locals del mòdul. Una vegada es realitzes una crida al sistema en comptes d'utilitzar-se la crida original s'utilitzaria la funció local del mòdul, aquesta imprimiria al registre del nucli el nom de la crida utilitzada juntament amb el valor dels seus paràmetres per després executar la crida original amb els paràmetres pertinents. A continuació es mostrerà cada fragment del codi d'aquest mòdul fet en C amb la seva explicació:

Primer de tot s'han de fer les declaracions de les funcions que son necessàries per a la utilització de funcions, estructures, etc:

```
1 #include <plat/omap-serial.h>
```

Codi 4: Mòdul (part 1)

Després s'han de definir a través de macros predefinides característiques d'aquest mòdul com el seu nom, el seu creador, la seva llicència.

```
3 MODULE_LICENSE("GPL");  
4 MODULE_AUTHOR("Alberto Burgos Plaza");  
5 MODULE_DESCRIPTION("Mòdul per a la intercepció de les crides a  
sistema realitzades sobre el dispositiu de caràcters tty00 i  
el port UART 0");
```

Codi 5: Mòdul (part 2)

A continuació es definiran els símbols exportats com a externs per a indicar al nucli que aquests son elements no locals.

```
7 extern struct list_head tty_drivers;  
8 extern struct uart_omap_port *ui[OMAP_MAX_HSUART_PORTS];
```

Codi 6: Mòdul (part 3)

Tot seguit s'han de definir les variables on s'emmagatzemaran les direccions de memòria originals de les crides a sistema segregades. Aquestes direccions originals es desen ja que si es vol descarregar el mòdul del nucli serà necessari retornar les direccions originals a les crides a sistema. Primer es realitza per a les crides a sistema sobre el dispositiu tty00 i després per les crides a sistema sobre el port UART.

```
10 asmlinkage int (*install_saved)(struct tty_driver *, struct  
tty_struct *);  
11 asmlinkage void (*remove_saved)(struct tty_driver *, struct  
tty_struct *);  
12 asmlinkage int (*open_saved)(struct tty_struct *, struct file  
*);  
13 asmlinkage void (*close_saved)(struct tty_struct *, struct file  
*);  
14 asmlinkage void (*shutdown_saved)(struct tty_struct *);
```

```

15 asmlinkage void (*cleanup_saved) (struct tty_struct *);
16 asmlinkage int  (*write_saved) (struct tty_struct *, const
        unsigned char *, int);
17 asmlinkage int  (*put_char_saved) (struct tty_struct *, unsigned
        char);
18 asmlinkage void (*flush_chars_saved) (struct tty_struct *);
19 asmlinkage int  (*write_room_saved) (struct tty_struct *);
20 asmlinkage int  (*chars_in_buffer_saved) (struct tty_struct *);
21 asmlinkage int  (*ioctl_saved) (struct tty_struct *, unsigned
        int, unsigned long);
22 asmlinkage long (*compat_ioctl_saved) (struct tty_struct *,
        unsigned int, unsigned long);
23 asmlinkage void (*set_termios_saved) (struct tty_struct *, struct
        ktermios *);
24 asmlinkage void (*throttle_saved) (struct tty_struct *);
25 asmlinkage void (*unthrottle_saved) (struct tty_struct *);
26 asmlinkage void (*stop_saved) (struct tty_struct *);
27 asmlinkage void (*start_saved) (struct tty_struct *);
28 asmlinkage void (*hangup_saved) (struct tty_struct *);
29 asmlinkage int  (*break_ctl_saved) (struct tty_struct *, int);
30 asmlinkage void (*flush_buffer_saved) (struct tty_struct *);
31 asmlinkage void (*set_ldisc_saved) (struct tty_struct *);
32 asmlinkage void (*wait_until_sent_saved) (struct tty_struct *,
        int);
33 asmlinkage void (*send_xchar_saved) (struct tty_struct *, char);
34 asmlinkage int  (*tiocmget_saved) (struct tty_struct *);

```

```

35 asmlinkage int (*tiocmset_saved) (struct tty_struct *, unsigned
    int, unsigned int);

36 asmlinkage int (*resize_saved) (struct tty_struct *, struct
    winsize *);

37 asmlinkage int (*set_termiox_saved) (struct tty_struct *, struct
    termiox *);

38 asmlinkage int (*get_icount_saved) (struct tty_struct *, struct
    serial_icounter_struct *);

39

40 asmlinkage unsigned int (*tx_empty_saved) (struct uart_port
    *);

41 asmlinkage void (*set_mctrl_saved) (struct uart_port
    *, unsigned int);

42 asmlinkage unsigned int (*get_mctrl_saved) (struct uart_port
    *);

43 asmlinkage void (*stop_tx_saved) (struct uart_port *);

44 asmlinkage void (*start_tx_saved) (struct uart_port
    *);

45 asmlinkage void (*stop_rx_saved) (struct uart_port *);

46 asmlinkage void (*enable_ms_saved) (struct uart_port
    *);

47 asmlinkage void (*pm_saved) (struct uart_port *,
    unsigned int, unsigned int);

48 asmlinkage void (*wake_peer_saved) (struct uart_port
    *);

49 asmlinkage void (*release_port_saved) (struct
    uart_port *);

50 asmlinkage int (*request_port_saved) (struct
    uart_port *);

51 asmlinkage void (*config_port_saved) (struct uart_port
    *, int);

```

```
52 asmlinkage int (*verify_port_saved)(struct uart_port
*, struct serial_struct *);
```

Codi 7: Mòdul (part 4)

A continuació es realitza la implementació de les funcions locals que faran d'intermediàries durant l'execució d'una crida a sistema. Hi han funcions que proporcionen més informació degut a que, com es comprovarà en el següent capítol, son aquestes les úniques que s'utilitzen realment durant l'execució de les funcions de localització.

```
54 asmlinkage int install_local(struct tty_driver *driver, struct
tty_struct *tty) {

55     printk(KERN_ALERT "Crida install a GPS.\n");

56     return install_saved(driver, tty);

57 }

58

59 asmlinkage void remove_local(struct tty_driver *driver, struct
tty_struct *tty) {

60     printk(KERN_ALERT "Crida remove a GPS.\n");

61     return remove_saved(driver, tty);

62 }

63

64 asmlinkage int open_local(struct tty_struct * tty, struct file
* filp) {

65     printk(KERN_ALERT "Crida open a GPS.\n");

66     return open_saved(tty, filp);

67 }

68

69 asmlinkage void close_local(struct tty_struct * tty, struct file
* filp) {

70     printk(KERN_ALERT "Crida close a GPS.\n");
```

```

71     return close_saved(tty, filp);

72 }

73

74 asmlinkage void shutdown_local(struct tty_struct *tty) {

75     printk(KERN_ALERT "Crida shutdown a GPS.\n");

76     shutdown_saved(tty);

77 }

78

79 asmlinkage void cleanup_local(struct tty_struct *tty) {

80     printk(KERN_ALERT "Crida cleanup a GPS.\n");

81     cleanup_saved(tty);

82 }

83

84 asmlinkage int write_local(struct tty_struct * tty, const
unsigned char *buf, int count) {

85     printk(KERN_ALERT "write(%s, %d)\n", buf, count);

86     return write_saved(tty, buf, count);

87 }

88

89 asmlinkage int put_char_local(struct tty_struct *tty, unsigned
char ch) {

90     printk(KERN_ALERT "Crida put_char a GPS.\n");

91     return put_char_saved(tty, ch);

92 }

93

```

```
94 asmlinkage void flush_chars_local(struct tty_struct *tty) {  
95     printk(KERN_ALERT "Crida flush_chars a GPS.\n");  
96     return flush_chars_saved(tty);  
97 }  
98  
99 asmlinkage int write_room_local(struct tty_struct *tty) {  
100    printk(KERN_ALERT "write_room()\n");  
101    return write_room_saved(tty);  
102 }  
103  
104 asmlinkage int chars_in_buffer_local(struct tty_struct *tty) {  
105    printk(KERN_ALERT "chars_in_buffer()\n");  
106    return chars_in_buffer_saved(tty);  
107 }  
108  
109 asmlinkage int ioctl_local(struct tty_struct *tty, unsigned int cmd, unsigned long arg) {  
110    printk(KERN_ALERT "ioctl(%d, %lu)\n", cmd, arg);  
111    return ioctl_saved(tty, cmd, arg);  
112 }  
113  
114 asmlinkage long compat_ioctl_local(struct tty_struct *tty, unsigned int cmd, unsigned long arg) {  
115    printk(KERN_ALERT "Crida compat_ioctl a GPS.\n");  
116    return compat_ioctl_saved(tty, cmd, arg);
```

```
117 }

118

119 asmlinkage void set_termios_local(struct tty_struct *tty, struct
ktermios * old) {

120     printk(KERN_ALERT "Crida set_termios a GPS.\n");

121     return set_termios_saved(tty, old);

122 }

123

124 asmlinkage void throttle_local(struct tty_struct * tty) {

125     printk(KERN_ALERT "Crida throttle a GPS.\n");

126     return throttle_saved(tty);

127 }

128

129 asmlinkage void unthrottle_local(struct tty_struct * tty) {

130     printk(KERN_ALERT "Crida unthrottle a GPS.\n");

131     return unthrottle_saved(tty);

132 }

133

134 asmlinkage void stop_local(struct tty_struct *tty) {

135     printk(KERN_ALERT "Crida stop a GPS.\n");

136     return stop_saved(tty);

137 }

138

139 asmlinkage void start_local(struct tty_struct *tty) {
```

```
140     printk(KERN_ALERT "Crida start a GPS.\n");  
  
141     return start_saved(tty);  
  
142 }  
  
143  
  
144 asmlinkage void hangup_local(struct tty_struct *tty) {  
  
145     printk(KERN_ALERT "Crida hangup a GPS.\n");  
  
146     return hangup_saved(tty);  
  
147 }  
  
148  
  
149 asmlinkage int break_ctl_local(struct tty_struct *tty, int  
state) {  
  
150     printk(KERN_ALERT "Crida break_ctl a GPS.\n");  
  
151     return break_ctl_saved(tty, state);  
  
152 }  
  
153  
  
154 asmlinkage void flush_buffer_local(struct tty_struct *tty) {  
  
155     printk(KERN_ALERT "flush_buffer()\n");  
  
156     return flush_buffer_saved(tty);  
  
157 }  
  
158  
  
159 asmlinkage void set_ldisc_local(struct tty_struct *tty) {  
  
160     printk(KERN_ALERT "Crida set_ldisc a GPS.\n");  
  
161     return set_ldisc_saved(tty);  
  
162 }
```

```

163

164 asmlinkage void wait_until_sent_local(struct tty_struct *tty,
   int timeout) {

165     printk(KERN_ALERT "Crida wait_until_sent a GPS.\n");

166     return wait_until_sent_saved(tty, timeout);

167 }

168

169 asmlinkage void send_xchar_local(struct tty_struct *tty, char
ch) {

170     printk(KERN_ALERT "Crida send_xchar a GPS.\n");

171     return send_xchar_saved(tty, ch);

172 }

173

174 asmlinkage int tiocmget_local(struct tty_struct *tty) {

175     printk(KERN_ALERT "Crida tiocmget a GPS.\n");

176     return tiocmget_saved(tty);

177 }

178

179 asmlinkage int tiocmset_local(struct tty_struct *tty, unsigned
int set, unsigned int clear) {

180     printk(KERN_ALERT "Crida tiocmset a GPS.\n");

181     return tiocmset_saved(tty, set, clear);

182 }

183

184 asmlinkage int resize_local(struct tty_struct *tty, struct
winsize *ws) {

```

```
185     printk(KERN_ALERT "Crida resize a GPS.\n");  
  
186     return resize_saved(tty, ws);  
  
187 }  
  
188  
  
189 asmlinkage int set_termiox_local(struct tty_struct *tty, struct  
termiox *tnew) {  
  
190     printk(KERN_ALERT "Crida set_termiox a GPS.\n");  
  
191     return set_termiox_saved(tty, tnew);  
  
192 }  
  
193  
  
194 asmlinkage int get_icount_local(struct tty_struct *tty, struct  
serial_icounter_struct *icount) {  
  
195     printk(KERN_ALERT "Crida get_icount a GPS.\n");  
  
196     return get_icount_saved(tty, icount);  
  
197 }  
  
198  
  
199 asmlinkage unsigned int tx_empty_local(struct uart_port *up) {  
  
200     printk(KERN_ALERT "tx_empty()\n");  
  
201     return tx_empty_saved(up);  
  
202 }  
  
203  
  
204 asmlinkage void set_mctrl_local(struct uart_port *up, unsigned  
int mctrl) {  
  
205     printk(KERN_ALERT "Crida set_mctrl a GPS.\n");  
  
206     return set_mctrl_saved(up, mctrl);
```

```

207 }

208

209 asmlinkage unsigned int get_mctrl_local(struct uart_port *up) {
210     printk(KERN_ALERT "Crida get_mctrl a GPS.\n");
211     return get_mctrl_saved(up);
212 }

213

214 asmlinkage void stop_tx_local(struct uart_port *up) {
215     printk(KERN_ALERT "Crida stop_tx a GPS.\n");
216     return stop_tx_saved(up);
217 }

218

219 asmlinkage void start_tx_local(struct uart_port *up) {
220     printk(KERN_ALERT "start_tx()\n");
221     return start_tx_saved(up);
222 }

223

224 asmlinkage void stop_rx_local(struct uart_port *up) {
225     printk(KERN_ALERT "stop_rx()\n");
226     return stop_rx_saved(up);
227 }

228

229 asmlinkage void enable_ms_local(struct uart_port *up) {
230     printk(KERN_ALERT "Crida enable_ms a GPS.\n");

```

```
231     return enable_ms_saved(up);  
  
232 }  
  
233  
  
234 asmlinkage void pm_local(struct uart_port *up, unsigned int state, unsigned int oldstate) {  
  
235     printk(KERN_ALERT "pm(%d, %d)\n", state, oldstate);  
  
236     return pm_saved(up, state, oldstate);  
  
237 }  
  
238  
  
239 asmlinkage void wake_peer_local(struct uart_port *up) {  
  
240     printk(KERN_ALERT "wake_peer()\n");  
  
241     return wake_peer_saved(up);  
  
242 }  
  
243  
  
244 asmlinkage void release_port_local(struct uart_port *up) {  
  
245     printk(KERN_ALERT "Crida release_port a GPS.\n");  
  
246     return release_port_saved(up);  
  
247 }  
  
248  
  
249 asmlinkage int request_port_local(struct uart_port *up) {  
  
250     printk(KERN_ALERT "Crida request_port a GPS.\n");  
  
251     return request_port_saved(up);  
  
252 }  
  
253
```

```

254 asmlinkage void config_port_local(struct uart_port *up, int op)
{
255     printk(KERN_ALERT "Crida config_port a GPS.\n");
256     return config_port_saved(up, op);
257 }
258
259 asmlinkage int verify_port_local(struct uart_port *up, struct
260 serial_struct *ss) {
260     printk(KERN_ALERT "Crida verify_port a GPS.\n");
261     return verify_port_saved(up, ss);
262 }
```

Codi 8: Mòdul (part 5)

Es necessitaran dos punters per a la resta del mòdul. Un és el punter a tty_driver que serà utilitzat per avançar a la llista tty_drivers en la cerca del dispositiu ttyO0. L'altre és un punter utilitzat per a enganyar al compilador ja que aquest està configurat per a detectar canvis a zones de memòria constants

```

264 unsigned long *trampa;
265 struct tty_driver *p = NULL;
```

Codi 9: Mòdul (part 6)

A continuació es declara la funció d'inici de mòdul. Un mòdul té una funció que s'executarà al carregar-se i una altre que s'executarà al descarregar-se.

```

267 static int __init module_start(void)
268 {
```

Codi 10: Mòdul (part 7)

Tot seguit es cercarà el dispositiu ttyO a la llista tty_drivers. Per a la cerca s'utilitza el patró ttyO. El primer dispositiu que es trobarà serà el dispositiu amb el nom ttyO0 el qual tindrà enllaçats la resta de nodes ttyO*.

```

269     char *name = "tty0";
270
271     list_for_each_entry(p, &tty_drivers, tty_drivers) {
272         if (strncmp(name, p->name, strlen(name)) == 0) {
273             break;
274         }
275     }

```

Codi 11: Mòdul (part 8)

Una vegada obtinguda la posició de memòria del dispositiu tty00 es procedirà a emmagatzemar les posicions de memòria originals a les que apunten els punters de cada crida al sistema.

```

277 // Copia de les direccions originals (tty00)
278 write_saved = p->ops->write;
279 put_char_saved = p->ops->put_char;
280 flush_chars_saved = p->ops->flush_chars;
281 write_room_saved = p->ops->write_room;
282 chars_in_buffer_saved = p->ops->chars_in_buffer;
283 ioctl_saved = p->ops->ioctl;
284 compat_ioctl_saved = p->ops->compat_ioctl;
285 set_termios_saved = p->ops->set_termios;
286 throttle_saved = p->ops->throttle;
287 unthrottle_saved = p->ops->unthrottle;
288 stop_saved = p->ops->stop;
289 start_saved = p->ops->start;

```

```

290     hangup_saved = p->ops->hangup;
291     break_ctl_saved = p->ops->break_ctl;
292     flush_buffer_saved = p->ops->flush_buffer;
293     set_ldisc_saved = p->ops->set_ldisc;
294     wait_until_sent_saved = p->ops->wait_until_sent;
295     send_xchar_saved = p->ops->send_xchar;
296     tiocmget_saved = p->ops->tiocmget;
297     tiocmset_saved = p->ops->tiocmset;
298     resize_saved = p->ops->resize;
299     set_termiox_saved = p->ops->set_termiox;
300     get_icount_saved = p->ops->get_icount;

```

Codi 12: Mòdul (part 9)

També es farà el mateix amb els punters a crides a sistema del port UART. El port on és el dispositiu GPS és el 0.

```

302     // Copia de les direccions originals (UART port 0)
303     tx_empty_saved = ui[0]->port.ops->tx_empty;
304     set_mctrl_saved = ui[0]->port.ops->set_mctrl;
305     get_mctrl_saved = ui[0]->port.ops->get_mctrl;
306     stop_tx_saved = ui[0]->port.ops->stop_tx;
307     start_tx_saved = ui[0]->port.ops->start_tx;
308     stop_rx_saved = ui[0]->port.ops->stop_rx;
309     enable_ms_saved = ui[0]->port.ops->enable_ms;
310     pm_saved = ui[0]->port.ops->pm;

```

```

311     wake_peer_saved = ui[0]->port.ops->wake_peer;
312     release_port_saved = ui[0]->port.ops->release_port;
313     request_port_saved = ui[0]->port.ops->request_port;
314     config_port_saved = ui[0]->port.ops->config_port;
315     verify_port_saved = ui[0]->port.ops->verify_port;

```

Codi 13: Mòdul (part 10)

Després es pot fer l'intercanvi del valor dels punters a crides a sistema per les funcions locals utilitzant el punter trampa per a enganyar al compilador.

```

317 // Canvi de les direccions (tty00)
318 trampa = (unsigned long *)&(p->ops->write);
319 *trampa = (unsigned long)&write_local;
320 trampa = (unsigned long *)&(p->ops->put_char);
321 *trampa = (unsigned long)&put_char_local;
322 trampa = (unsigned long *)&(p->ops->flush_chars);
323 *trampa = (unsigned long)&flush_chars_local;
324 trampa = (unsigned long *)&(p->ops->write_room);
325 *trampa = (unsigned long)&write_room_local;
326 trampa = (unsigned long *)&(p->ops->chars_in_buffer);
327 *trampa = (unsigned long)&chars_in_buffer_local;
328 trampa = (unsigned long *)&(p->ops->ioctl);
329 *trampa = (unsigned long)&ioctl_local;
330 trampa = (unsigned long *)&(p->ops->compat_ioctl);
331 *trampa = (unsigned long)&compat_ioctl_local;

```

```

332     trampa = (unsigned long *)&(p->ops->set_termios);
333     *trampa = (unsigned long)&set_termios_local;
334     trampa = (unsigned long *)&(p->ops->throttle);
335     *trampa = (unsigned long)&throttle_local;
336     trampa = (unsigned long *)&(p->ops->unthrottle);
337     *trampa = (unsigned long)&unthrottle_local;
338     trampa = (unsigned long *)&(p->ops->stop);
339     *trampa = (unsigned long)&stop_local;
340     trampa = (unsigned long *)&(p->ops->start);
341     *trampa = (unsigned long)&start_local;
342     trampa = (unsigned long *)&(p->ops->hangup);
343     *trampa = (unsigned long)&hangup_local;
344     trampa = (unsigned long *)&(p->ops->break_ctl);
345     *trampa = (unsigned long)&break_ctl_local;
346     trampa = (unsigned long *)&(p->ops->flush_buffer);
347     *trampa = (unsigned long)&flush_buffer_local;
348     trampa = (unsigned long *)&(p->ops->set_ldisc);
349     *trampa = (unsigned long)&set_ldisc_local;
350     trampa = (unsigned long *)&(p->ops->wait_until_sent);
351     *trampa = (unsigned long)&wait_until_sent_local;
352     trampa = (unsigned long *)&(p->ops->send_xchar);
353     *trampa = (unsigned long)&send_xchar_local;
354     trampa = (unsigned long *)&(p->ops->tiocmget);
355     *trampa = (unsigned long)&tiocmget_local;

```

```

356     trampa = (unsigned long *) &(p->ops->tiocmset);
357     *trampa = (unsigned long) &tiocmset_local;
358     trampa = (unsigned long *) &(p->ops->resize);
359     *trampa = (unsigned long) &resize_local;
360     trampa = (unsigned long *) &(p->ops->set_termiox);
361     *trampa = (unsigned long) &set_termiox_local;
362     trampa = (unsigned long *) &(p->ops->get_icount);
363     *trampa = (unsigned long) &get_icount_local;
364
365 // Canvi de les direccions (UART port 0)
366     trampa = (unsigned long *) &(ui[0]->port.ops->tx_empty);
367     *trampa = (unsigned long) &tx_empty_local;
368     trampa = (unsigned long *) &(ui[0]->port.ops->set_mctrl);
369     *trampa = (unsigned long) &set_mctrl_local;
370     trampa = (unsigned long *) &(ui[0]->port.ops->get_mctrl);
371     *trampa = (unsigned long) &get_mctrl_local;
372     trampa = (unsigned long *) &(ui[0]->port.ops->stop_tx);
373     *trampa = (unsigned long) &stop_tx_local;
374     trampa = (unsigned long *) &(ui[0]->port.ops->start_tx);
375     *trampa = (unsigned long) &start_tx_local;
376     trampa = (unsigned long *) &(ui[0]->port.ops->stop_rx);
377     *trampa = (unsigned long) &stop_rx_local;
378     trampa = (unsigned long *) &(ui[0]->port.ops->enable_ms);
379     *trampa = (unsigned long) &enable_ms_local;

```

```

380     trampa = (unsigned long *)(&(ui[0]->port.ops->pm));
381     *trampa = (unsigned long)&pm_local;
382     trampa = (unsigned long *)(&(ui[0]->port.ops->wake_peer));
383     *trampa = (unsigned long)&wake_peer_local;
384     trampa = (unsigned long *)(&(ui[0]->port.ops->release_port));
385     *trampa = (unsigned long)&release_port_local;
386     trampa = (unsigned long *)(&(ui[0]->port.ops->request_port));
387     *trampa = (unsigned long)&request_port_local;
388     trampa = (unsigned long *)(&(ui[0]->port.ops->config_port));
389     *trampa = (unsigned long)&config_port_local;
390     trampa = (unsigned long *)(&(ui[0]->port.ops->verify_port));
391     *trampa = (unsigned long)&verify_port_local;

```

Codi 14: Mòdul (part 11)

Es tanca la funció amb el retorn d'un enter 0.

```

392
393     return 0;
394 }

```

Codi 15: Mòdul (part 12)

Ara es definirà la funció de descàrrega del mòdul per a invertir els canvis realitzats a la funció de càrrega. Els punters a crides al sistema del dispositiu ttyO0 i del port UART 0 seran restablerts per les originals emmagatzemades a l'inici de la funció de càrrega del mòdul.

```

396 static void __exit module_end(void)
397 {
398     // Restauració de les direccions originals (tty00)
399     trampa = (unsigned long *)&(p->ops->write);
400     *trampa = (unsigned long)write_saved;
401     trampa = (unsigned long *)&(p->ops->put_char);
402     *trampa = (unsigned long)put_char_saved;
403     trampa = (unsigned long *)&(p->ops->flush_chars);
404     *trampa = (unsigned long)flush_chars_saved;
405     trampa = (unsigned long *)&(p->ops->write_room);
406     *trampa = (unsigned long)write_room_saved;
407     trampa = (unsigned long *)&(p->ops->chars_in_buffer);
408     *trampa = (unsigned long)chars_in_buffer_saved;
409     trampa = (unsigned long *)&(p->ops->iioctl);
410     *trampa = (unsigned long)iioctl_saved;
411     trampa = (unsigned long *)&(p->ops->compat_ioctl);
412     *trampa = (unsigned long)compat_ioctl_saved;
413     trampa = (unsigned long *)&(p->ops->set_termios);
414     *trampa = (unsigned long)set_termios_saved;
415     trampa = (unsigned long *)&(p->ops->throttle);
416     *trampa = (unsigned long)throttle_saved;
417     trampa = (unsigned long *)&(p->ops->unthrottle);
418     *trampa = (unsigned long)unthrottle_saved;
419     trampa = (unsigned long *)&(p->ops->stop);

```

```

420     *trampa = (unsigned long)stop_saved;
421     trampa = (unsigned long *)&(p->ops->start);
422     *trampa = (unsigned long)start_saved;
423     trampa = (unsigned long *)&(p->ops->hangup);
424     *trampa = (unsigned long)hangup_saved;
425     trampa = (unsigned long *)&(p->ops->break_ctl);
426     *trampa = (unsigned long)break_ctl_saved;
427     trampa = (unsigned long *)&(p->ops->flush_buffer);
428     *trampa = (unsigned long)flush_buffer_saved;
429     trampa = (unsigned long *)&(p->ops->set_ldisc);
430     *trampa = (unsigned long)set_ldisc_saved;
431     trampa = (unsigned long *)&(p->ops->wait_until_sent);
432     *trampa = (unsigned long)wait_until_sent_saved;
433     trampa = (unsigned long *)&(p->ops->send_xchar);
434     *trampa = (unsigned long)send_xchar_saved;
435     trampa = (unsigned long *)&(p->ops->tiocmget);
436     *trampa = (unsigned long)tiocmget_saved;
437     trampa = (unsigned long *)&(p->ops->tiocmset);
438     *trampa = (unsigned long)tiocmset_saved;
439     trampa = (unsigned long *)&(p->ops->resize);
440     *trampa = (unsigned long)resize_saved;
441     trampa = (unsigned long *)&(p->ops->set_termiox);
442     *trampa = (unsigned long)set_termiox_saved;
443     trampa = (unsigned long *)&(p->ops->get_icount);

```

```

444 *trampa = (unsigned long)get_icount_saved;
445
446 // Restauració de les direccions originals (UART port 0)
447 trampa = (unsigned long *)&(ui[0]->port.ops->tx_empty);
448 *trampa = (unsigned long)tx_empty_saved;
449 trampa = (unsigned long *)&(ui[0]->port.ops->set_mctrl);
450 *trampa = (unsigned long)set_mctrl_saved;
451 trampa = (unsigned long *)&(ui[0]->port.ops->get_mctrl);
452 *trampa = (unsigned long)get_mctrl_saved;
453 trampa = (unsigned long *)&(ui[0]->port.ops->stop_tx);
454 *trampa = (unsigned long)stop_tx_saved;
455 trampa = (unsigned long *)&(ui[0]->port.ops->start_tx);
456 *trampa = (unsigned long)start_tx_saved;
457 trampa = (unsigned long *)&(ui[0]->port.ops->stop_rx);
458 *trampa = (unsigned long)stop_rx_saved;
459 trampa = (unsigned long *)&(ui[0]->port.ops->enable_ms);
460 *trampa = (unsigned long)enable_ms_saved;
461 trampa = (unsigned long *)&(ui[0]->port.ops->pm);
462 *trampa = (unsigned long)pm_saved;
463 trampa = (unsigned long *)&(ui[0]->port.ops->wake_peer);
464 *trampa = (unsigned long)wake_peer_saved;
465 trampa = (unsigned long *)&(ui[0]->port.ops->release_port);
466 *trampa = (unsigned long)release_port_saved;
467 trampa = (unsigned long *)&(ui[0]->port.ops->request_port);

```

```

468     *trampa = (unsigned long)request_port_saved;
469     trampa = (unsigned long *)&(ui[0]->port.ops->config_port);
470     *trampa = (unsigned long)config_port_saved;
471     trampa = (unsigned long *)&(ui[0]->port.ops->verify_port);
472     *trampa = (unsigned long)verify_port_saved;
473 }
```

Codi 16: Mòdul (part 13)

Finalment es defineixen quines son les funcions de càrrega i descàrrega del mòdul amb les funcions `module_init` i `module_exit` respectivament.

```

475 module_init(module_start);
476 module_exit(module_end);
```

Codi 17: Mòdul (part 14)

5.2.3.1 Compilació i construcció

Una vegada implementat el mòdul s'ha de crear un **Makefile** per a la seva construcció. Els fragments d'aquest fitxer seran explicats a continuació:

Primer de tot s'han de definir les variables necessàries per a la construcció del mòdul. Entre aquestes es troben:

- **obj-m**: el seu valor serà el fitxer de sortida. Aquest valor ha de ser el mateix que el del fitxer font del mòdul a compilar, en aquest cas **monitoritzacio.c**.
- **KDIR**: el seu valor és la ruta al nucli construït anteriorment per a la utilització de les llibreries incloses al mòdul.
- **COMPILER** i **CROSS_COMPILE**: el seu valor és el mateix, la ruta al compilador creuat per a ARM que es troba en aquest cas al codi font d'Android.

```

1 obj-m := monitoritzacio.o

2 KDIR := ~/src/android_kernel omap

3 PWD := $(shell pwd)

4 COMPILER := ~/src/android/prebuilt/linux-x86/toolchain/arm-eabi-
   4.4.3/bin/arm-eabi-

5 CROSS_COMPILE := $(COMPILER)

```

Codi 18: Makefile del mòdul (part 1)

Tot seguit es troben els diferents mètodes que es podran executar amb la comanda **make**. S'ha de definir almenys el cas en el que s'executa make sense paràmetres amb el mètode default. En aquest cas també s'especifica la neteja dels elements creats per la construcció del mòdul amb el mètode clean:

```

7 default:

8 $(MAKE) -C $(KDIR) ARCH=arm SUBDIRS=$(PWD) M=$(PWD)
      CROSS_COMPILE=$(COMPILER) modules

9 clean:

10 $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean

```

Codi 19: Makefile del mòdul (part 2)

Una vegada implementat el Makefile es podrà realitzar la seva construcció executant la comanda make. Aquesta generarà un fitxer anomenat **monitoritzacio.ko** que serà el mòdul a carregar. S'ha de tindre en compte que el fitxer Makefile ha de ser-hi al mateix directori que el fitxer font del mòdul:

```
$ make
```

Després, si es volen eliminar els fitxers generats per la construcció del mòdul s'haurà d'executar la comanda make amb el paràmetre **clean**:

```
$ make clean
```

5.2.4 Inserció del mòdul i obtenció dels resultats

Una vegada finalitzat el mòdul s'afegeix aquest al dispositiu i es carrega per a obtindre les crides a sistema realitzades:

```
$ adb push monitoritzacio.ko /data/local/tmp/  
$ adb shell  
shell@android:/ # insmod /data/local/tmp/monitoritzacio.ko
```

Després es poden comprovar les crides registrades llegit el fitxer /proc/kmsg:

```
shell@android:/ # cat /proc/kmsg
```

En acabar, es pot descarregar el mòdul per a tornar a deixar el nucli com estava abans de carregar aquest:

```
shell@android:/ # rmmod /data/local/tmp/monitoritzacio.ko
```

5.3 Conclusions

Aquesta part del projecte ha sigut bastant difícil d'aconseguir ja que es van cometre errors a l'hora de no tindre en compte que el xip GPS estava connectat a un bus de dades controlat per el xip UART. També es va tindre el problema de modificar memòria aliena al mòdul ja que el compilador no acceptava aquests canvis, per això es van provar diverses solucions.

Finalment el resultat és satisfactori ja que permet identificar correctament quines son les crides a sistema utilitzades en cada moment i gràcies a això més endavant es van poder relacionar amb funcions de nivells superiors.

Capítol 6. Obtenció de les dades e interpretació

Arribat a aquest punt ja es compta amb la possibilitat d'interceptar les crides a sistema realitzades sobre el dispositiu de caràcters ttyO0 i sobre el port UART 0. Aquests dos conjunts de crides son les accions realitzades sobre al maquinari més properes a aquest. El que quedarà per fer a partir d'aquest capítol serà relacionar aquestes crides a sistema amb funcions de l'API d'Android. És per això que cal l'elaboració d'una aplicació d'usuari que utilitzi aquestes.

Després de tindre la relació de crides a sistema amb funcions de l'API d'Android serà necessari seleccionar els patrons que puguin ser identificats com a funcions executades al nivell de l'API i obtindre el consum d'aquests. Una vegada obtingut el consum es tractaran els resultats per a poder extreure conclusions.

6.1 Relació entre funcions de l'API i les crides a sistema interceptades

Tal i com s'ha comentat a la introducció d'aquest capítol és necessari la creació d'una aplicació que utilitzi les funcions de l'API d'Android per a generar crides a sistema i poder relacionar unes amb les altres. En concret, les funcions a relacionar seran les utilitzades per les classes del paquet **android.location**. El paquet android.location proveeix l'accés als serveis de localització del sistema amb diferents interfícies i classes:

- Interfícies
 - **GpsStatus.Listener**: utilitzat per a rebre notificacions quan l'estat del GPS canvia. Aquest listener genera esdeveniments quan el GPS s'inicia, s'atura, quan es rep la primera localització o periòdicament per a enviar la informació de l'estat dels satèl·lits.
 - **GpsStatus.NmeaListener**: utilitzat per a rebre sentències NMEA des del GPS. En el cas d'aquest projecte aquest listener no ha funcionat possiblement degut a que no està implementat aquest mètode al controlador.
 - **LocationListener**: utilitzat per a rebre notificacions cada vegada que la localització canvia. Per a que aquest listener generi esdeveniments serà necessari l'execució del mètode requestLocationUpdates.
- Classes

- **Address:** aquesta classe implementa mètodes per a representar adreces amb l'estructura que posa Google a disposició amb les seves API's o per a obtindre dades concretes d'una adreça. Aquesta classe únicament utilitza connectivitat a la xarxa per a descarregar la informació necessària així que el seu ús no és necessari per a l'objectiu d'aquest projecte.
- **Criteria:** aquesta classe permet establir criteris per a la utilització dels diferents serveis de localització que ofereix el sistema operatiu. Poden establir-se criteris com ara el consum energètic o la precisió. En el cas d'aquest projecte únicament s'utilitzarà el servei proveïdor del sistema GPS així que no s'utilitzaran criteris per a l'ús d'un o altre proveïdor de localització.
- **Geocoder:** aquesta classe serveix per transformar una frase a unes coordenades GPS o per realitzar el procés invers. Com el cas de la classe Address únicament s'utilitza la connectivitat a la xarxa i per tant no s'utilitzarà.
- **GpsSatellite:** aquesta classe serveix per a la obtenció dels paràmetres de l'estat de cada satèl·lit amb la interfície GpsStatus.Listener.
- **GpsStatus:** aquesta classe desenvolupa mètodes complementaris per a la classe anterior com ara la obtenció d'un vector iterable amb cadascun dels satèl·lits dels quals s'ha obtingut el senyal.
- **Location:** aquesta classe és necessària per al tractament de les localitzacions obtingudes des de la interfície LocationListener. Es poden obtindre dades com ara l'altitud, la longitud, la latitud, la velocitat, o es poden utilitzar mètodes més complexos com ara el càcul de la distància entre la posició actual i una altre.
- **LocationManager:** aquesta classe permet l'accés als serveis de localització geogràfica. Aquests serveis permeten la obtenció de localitzacions periòdiques o puntuals quan es compleixen unes condicions fixades.
- **LocationProvider:** aquesta classe permet obtindre informació dels diferents proveïdors de localització utilitzada per la classe Criteria.

Després d'haver-se provat tots els mètodes al mateix temps que es comprovaven les crides a sistema interceptades es va descobrir que únicament el mètode requestLocationUpdates provoca l'execució d'aquestes. Realment no va ser un fet inesperat ja que, tenint en compte el camí de dades seguit per una funció de l'API fins al maquinari, i sabent que realitza cada mètode, es podia augurar amb antelació aquest resultat.

Tenint en compte aquestes proves es va desenvolupar una aplicació definitiva per a realitzar la relació entre els esdeveniments que succeeixen a l'aplicació i les crides a sistema interceptades.

6.1.1 Creació d'aplicació d'usuari

Des de l'entorn de desenvolupament Eclipse i utilitzant el conjunt d'eines de desenvolupament d'Android va crear-se la següent aplicació. A continuació s'explicarà com es va crear el projecte i els fitxers que van ser necessaris modificar.

6.1.1.1 Creació d'un nou projecte

Per a fer una aplicació en Android des d'Eclipse primer de tot és necessari crear un nou projecte. Per a fer això s'anirà al menú “File” i es farà clic a “new project”.

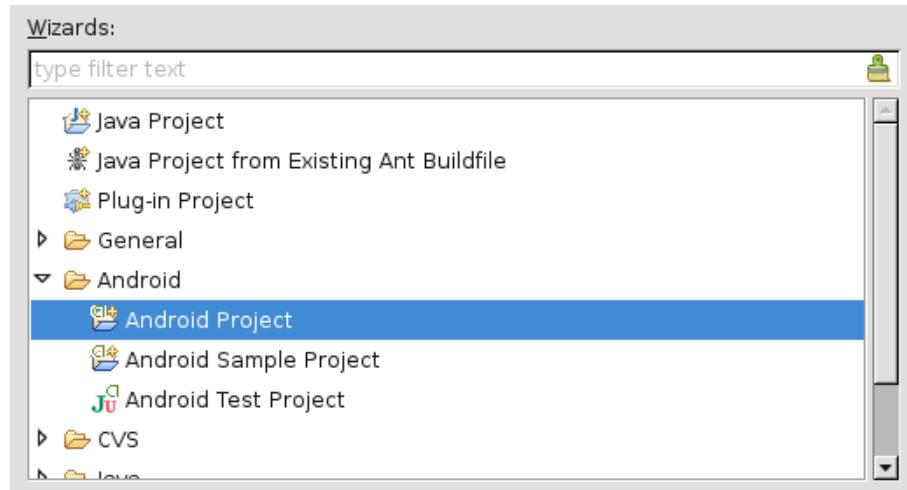


Figura 32: Creació d'un nou projecte, selecció d'un assistent

Després es seleccionarà el tipus de projecte. En aquest cas s'escolllirà “Android project” ja que el que interessa és un projecte buit. Es continuarà al següent pas fent clic a “Next”.

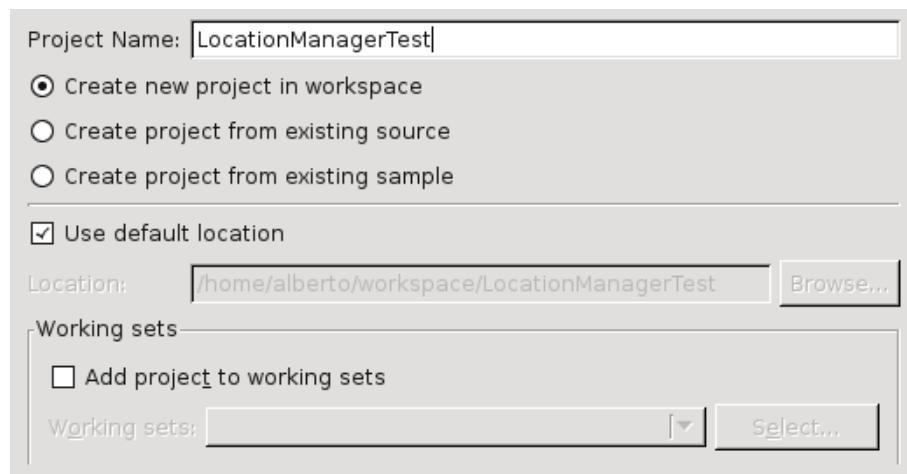


Figura 33: Creació d'un nou projecte, selecció de nom i tipus

A continuació serà necessari indicar-ne quin és el conjunt d'elements de desenvolupament amb el que es vol treballar. S'ha de tindre en compte per a quina versió del sistema operatiu Android es vol realitzar l'aplicació ja que a cada versió de l'API apareixen noves funcionalitats o canvis. En aquest cas es tracta d'un Android ICS 4.0.1 així que es seleccionarà la versió 14. Tot seguit es farà clic a “Next” per a continuar.

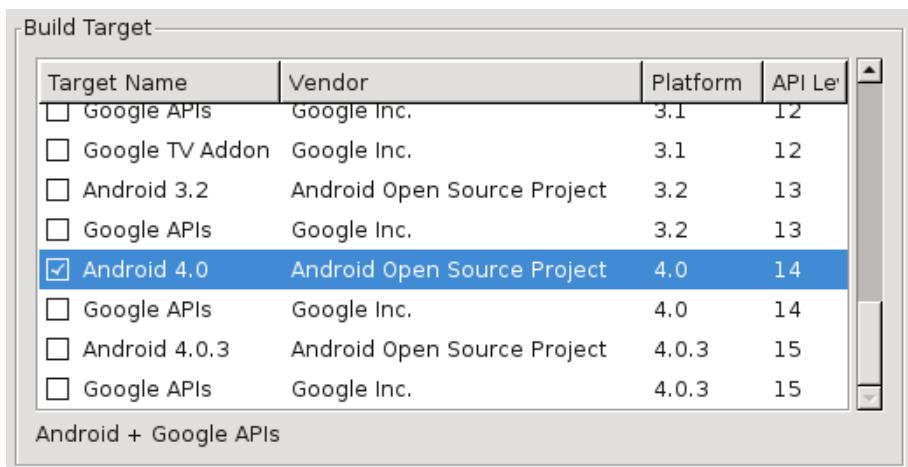


Figura 34: Creació d'un nou projecte, selecció d'SDK

A la següent finestra s'hauran d'introduir una sèrie de paràmetres com el nom de l'aplicació, el nom del paquet i, si es vol crear una activitat, el nom d'aquesta. En aquest cas l'aplicació comptarà amb una activitat per a visualitzar els esdeveniments i poder comparar-los amb les traces de crides a sistema interceptades.

Application Name:	LocationManagerTest
Package Name:	location.manager.test
<input checked="" type="checkbox"/> Create Activity:	LocationManagerTestActivity
Minimum SDK:	14
<input type="checkbox"/> Create a Test Project	
Test Project Name:	LocationManagerTestTest
Test Application:	LocationManagerTestTest
Test Package:	location.manager.test.test

Figura 35: Creació d'un nou projecte, configuració

Una vegada seguits aquests passos es farà clic a “Finish” i es tindrà una estructura de directoris com la següent:

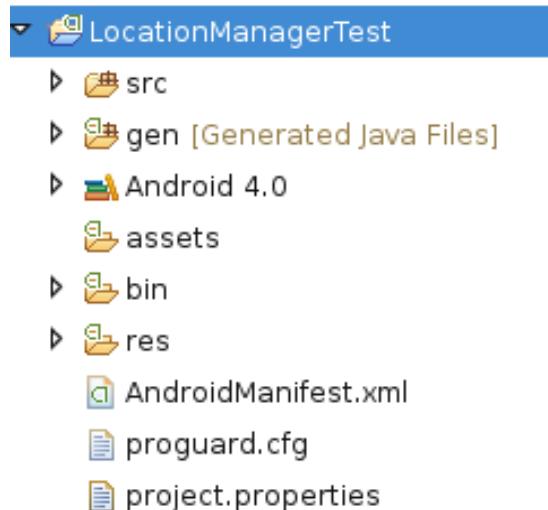


Figura 36: Creació d'un projecte, estructura

6.1.1.2 Edició de permisos

Una vegada creat el projecte s'hauran de definir els permisos necessaris que s'han d'atorgar a l'aplicació per a la utilització dels serveis del sistema. En aquest cas serà necessari atorgar el permís per al proveïdor del servei de localització GPS, anomenat **ACCESS_FINE_LOCATION**. Els permisos hauran de ser acceptats per l'usuari a l'hora d'instal·lar l'aplicació. Aquest operació es realitza modificant el fitxer **AndroidManifest.xml** situat a l'arrel del projecte. També pot realitzar-se gràficament.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest
3     xmlns:android="http://schemas.android.com/apk/res/android"
4
5     package="location.manager.test"
6
7     android:versionCode="1"
8
9     android:versionName="1.0" >
10
11
12     <uses-sdk android:minSdkVersion="14" />
13
14     <uses-permission
15         android:name="android.permission.ACCESS_FINE_LOCATION"/>
16
17
18
19
```

```

10    <application>
11        android:icon="@drawable/ic_launcher"
12        android:label="@string/app_name" >
13
14            <activity>
15                android:name=".LocationManagerTestActivity"
16                android:label="@string/app_name" >
17
18                    <intent-filter>
19                        <action
19                         android:name="android.intent.action.MAIN" />
20
21                    </intent-filter>
22            </activity>
23
24    </application>

```

Codi 20: Edició de permisos, AndroidManifest.xml

6.1.1.3 Edició de capes gràfiques

A continuació s'haurà de definir un mínim la capa gràfica que utilitzarà l'activitat. Per a fer això es modificarà el fitxer **main.xml** que es troba a **res/layout**. En aquest es definirà la forma vertical de l'aplicació i s'afegiran tres quadres de text per a introduir la informació que sigui necessari mostrar per pantalla. Cada quadre tindrà un identificador que serà utilitzat des de l'activitat. Aquesta operació pot fer-se gràficament.

```

1 <?xml version="1.0" encoding="utf-8"?>

```

```
2 <LinearLayout  
3     xmlns:android="http://schemas.android.com/apk/res/android"  
4     android:layout_width="fill_parent"  
5     android:layout_height="fill_parent"  
6     android:orientation="vertical" >  
7  
8     <TextView  
9         android:id="@+id/texto1"  
10        android:layout_width="fill_parent"  
11        android:layout_height="wrap_content"  
12        android:text="" />  
13  
14     <TextView  
15         android:id="@+id/texto2"  
16         android:layout_width="fill_parent"  
17         android:layout_height="wrap_content"  
18         android:text="" />  
19  
20     <TextView  
21         android:id="@+id/texto3"  
22         android:layout_width="fill_parent"  
23         android:layout_height="wrap_content"  
24         android:text="" />
```

```
25 </LinearLayout>
```

Codi 21: Edició de capes gràfiques, res/layout/main.xml

6.1.1.4 Edició d'activitat

Finalment s'haurà d'editar l'activitat creada durant la creació del projecte. En aquesta activitat es mostraran més mètodes dels estrictament necessaris per donar una visió global de la forma de treballar amb aquests. Principalment el que es realitza en aquesta activitat és la obtenció d'actualitzacions de localització geogràfica cada 10 segons i la obtenció dels canvis a l'estat del proveïdor GPS. El fitxer a editar és **LocationManagerTestActivity.java** i es troba a **src/location.manager.test**. A continuació es mostra per fragments la implementació de l'activitat juntament amb les explicacions pertinents:

Primer de tot es defineix el paquet que identifica a l'aplicació i s'importen les classes necessàries que seran carregades a Dalvik juntament amb l'aplicació.

```
1 package gps.test;  
2  
3 import java.util.Iterator;  
4 import java.lang.String;  
5 import android.app.Activity;  
6 import android.content.Context;  
7 import android.location.GpsSatellite;  
8 import android.location.GpsStatus;  
9 import android.location.GpsStatus.Listener;  
10 import android.location.Location;  
11 import android.location.LocationListener;  
12 import android.location.LocationManager;  
13 import android.os.Bundle;  
14 import android.widget.TextView;
```

Codi 22: Edició d'activitat, importació de classes

Seguidament es declara l'activitat i el mètode executat quan es carrega aquesta per primera vegada, `onCreate()`. Llavors serà necessària la obtenció de la referència del servei LocationManager. També es carrega la capa gràfica amb els seus tres quadres de text.

```

16 public class GPSTest401Activity extends Activity {
17
18     // Definició del mètode realitzat al executar l'activitat per
19     // primera vegada
20
21     @Override
22
23     public void onCreate(Bundle savedInstanceState) {
24
25         final TextView textView1 = (TextView)
26             findViewById(R.id.texto1);
27
28         final TextView textView2 = (TextView)
29             findViewById(R.id.texto2);
30
31         final TextView textView3 = (TextView)
32             findViewById(R.id.texto3);
33
34         // Obtenció de referència del servei LocationManager
35
36         final LocationManager locationManager =
37             (LocationManager)
38             getSystemService(Context.LOCATION_SERVICE);

```

Codi 23: Edició d'activitat, funció onCreate i obtenció de referència del servei LM

A continuació es defineix el listener utilitzat per a quan hi hagin esdeveniments provinents del servei LocationManager. A dintre s'han de definir obligatòriament tots els esdeveniments

disponibles: `onLocationChanged`, per a ser executat quan canvia la localització geogràfica, i els esdeveniments utilitzats quan s'utilitza més d'un proveïdor de localització geogràfica, `onStatusChanged`, `onProviderEnabled`, i `onProviderDisabled`. Aquests tres últims esdeveniments no s'utilitzaran encara que és necessari definir-los. Quan s'actualitza la localització geogràfica s'executen una sèrie de mètodes per a filtrar la informació obtinguda i mostrar-la per pantalla al quadre de text 1.

```
32      // Definició de listener per a LocationManager
33
34      LocationListener locationListener = new
35          LocationListener() {
36
37          // Definició de mètode per a l'actuació en una
38          // actualització de la posició
39
40          public void onLocationChanged(Location location) {
41
42              // Afegit un punt de referència per a calcular
43              // després la distància i el rumb
44
45              Location montserrat = new Location
46                  (LocationManager.GPS_PROVIDER);
47
48              montserrat.setLatitude(41.599334);
49
50              montserrat.setLongitude(1.814461);
51
52
53              // Obtenció de latitud, longitud i marca de temps
54              // UNIX epoch
55
56              String strLocProp = "Latitude: " +
57                  location.getLatitude() + "\n" +
58
59                      "Longitude: " +
60                  location.getLongitude() + "\n" +
61
62                      "Time: " +
63                  location.getTime() + "\n";
64
65
66
```

```

47          // Obtenció d'altitud si es té
48
49          if (location.hasAltitude()) {
50
51              strLocProp += "Altitude: " +
52                  location.getAltitude() + "\n";
53
54          }
55
56
57          // Càcul de la distància actual a Montserrat i el
58          // rumb des del nord geogràfic
59
60          strLocProp += "Distance to ms: " +
61              location.distanceTo(montserrat) + "\n";
62
63          strLocProp += "Bearing to ms: " +
64              location.bearingTo(montserrat) + "\n";
65
66          // mostratge de les dades al quadre de text 1

```

```

67         textView1.setText(strLocProp);

68

69     }

70

71     // Definició de mètode per a l'actuació al canviar d'un
    proveïdor a un altre

72     public void onStatusChanged(String provider, int
    status, Bundle extras) {

73

74     }

75

76     // Definició de mètode per a mostrar el canvi del
    proveïdor a habilitat

77     public void onProviderEnabled(String provider) {

78         textView1.setText("Provider enabled.");
    }

79

80

81     // Definició de mètode per a mostrar el canvi del
    proveïdor a deshabilitat

82     public void onProviderDisabled(String provider) {

83         textView1.setText("Provider disabled.");
    }

84

85

86     };

```

Codi 24: Edició d'activitat, listener LocationManager i esdeveniments

Després es realitzarà el mateix amb el listener per al servei GpsLocationProvider amb l'esdeveniment onGpsStatusChanged, per a l'execució quan canvien les dades d'estat del proveïdor de servei GPS. Dintre d'aquest esdeveniment hi han a la vegada quatre esdeveniments diferents: GPS_EVENT_SATELLITE_STATUS, que s'executarà quan es canviïn les dades relatives a l'estat de cada satèl·lit i es mostraran aquests al quadre de text 2, GPS_EVENT_FIRST_FIX, que proporcionarà el temps emprat per a la obtenció de l'actualització de la posició geogràfica i la seva mostra al quadre de text 3, GPS_EVENT_STARTED, que s'executarà quan s'iniciai el servei GPS, i GPS_EVENT_STOPPED quan el servei GPS passi a un estat deshabilitat. Aquests dos últims mostraran al quadre de text 3 si el servei s'ha executat o si s'ha aturat.

```

88         // Definició de listener per a GpsStatus

89         Listener listener = new GpsStatus.Listener() {

90             // Definició de mètode per a l'actuació en una
               actualització del estat del GPS

91             public void onGpsStatusChanged(int event) {

92                 // Obtenció de les dades d'estat del sistema GPS

93                 GpsStatus gpsStatus =
locationManager.getGpsStatus(null);

94                 if(gpsStatus != null) {

95                     switch (event) {

96                         // Actualització de l'estat dels
                           satèl·lits

97                         case GpsStatus.GPS_EVENT_SATELLITE_STATUS:

98                             Iterable<GpsSatellite>satellites =
gpsStatus.getSatellites();

99                             Iterator<GpsSatellite>sat =
satellites.iterator();

100                            int i = 0;

101                            String strGpsStats = "";

102                            // Obtenció de les dades per a cada
                           satèl·lit captat

```

```

103                         while (sat.hasNext()) {
104
105                             GpsSatellite satellite =
106                             sat.next();
107
108                             strGpsStats += (i++) + ":" +
109                             satellite.getPrn() + "," +
110                             satellite.getSnR() + "," +
111                             satellite.getAzimuth() + "," +
112                             satellite.getElevation();
113
114                             // Mostra dels satèl·lits al
115                             quadre de text 2
116
117                             textView2.setText(strGpsStats);
118
119                             break;
120
121                             // Obtenció del temps emprat per a
122                             l'actualització de la posició
123
124                             case GpsStatus.GPS_EVENT_FIRST_FIX:
125
126                                 textView3.setText("Time to fix: " +
127                                 gpsStatus.getTimeToFirstFix());
128
129                                 break;

```

```
123          // Canvi d'estat al servei GPS a  
124          habilitat  
125  
126          case GpsStatus.GPS_EVENT_STARTED:  
127              textView3.setText("GPS enabled.");  
128          // Canvi d'estat al servei GPS a  
129          desabilitat  
130          case GpsStatus.GPS_EVENT_STOPPED:  
131              textView3.setText("GPS disabled.");  
132          break;  
133      }  
134  }  
135 }  
136 };
```

Codi 25: Edició d'activitat, listener GpsLocationProvider i esdeveniments

Finalment queden per definir els mètodes per a que cada servei executi les accions pertinents i retorni dades i esdeveniments a l'activitat a través del listener.

```

138      // Registre del listener amb LocationManager per a rebre
           actualitzacions periòdiques de la posició cada 10s
139
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
    10000, 0, locationListener);
140      // Registre del listener amb GpsStatus per a rebre
           actualitzacions periòdiques de l'estat dels satèl·lits
141
locationManager.addGpsStatusListener(listener);
142 }
143 }
```

6.1.2 Establiment de relacions

Una vegada provats els diferents mètodes amb l'aplicació, independentzant les accions d'aquests per a la comprovació de les traces de crides a sistema generades, s'ha pogut observar que l'únic mètode que realitza crides a sistema és **requestLocationUpdates**. Durant l'execució d'aquest mètode s'originen els següents patrons de crides a sistema:

- **Comprovació de buffers:** hi ha una continua execució d'una parella de crides: **chars_in_buffer** i **write_room**. La primera rutina retorna el nombre de caràcters que hi son al buffer de lectura per a la seva obtenció. La segona torna el nombre de caràcters que el controlador UART acceptarà per a la cua d'escriptura. Aquest número canviarà quan els buffers de sortida es buidin o si es controla el flux de sortida.
- **Escriptura al port:** execució de les crides a sistema **write**, **wake_peer** i **start_tx** que serveixen per a escriure al buffer de sortida, per arrencar el sistema d'enviament de caràcters i per a realitzar l'enviament respectivament.
- **Suspensió i continuació del servei:** al mètode requestLocationUpdates es pot definir un temps de suspensió del sistema GPS amb l'excusa de l'estalvi d'energia. Durant la fase de parada d'aquest s'executen les crides a sistema **stop_rx** i **tx_empty** les quals comproven si des del xip GPS s'envien dades o si s'estan transmetent dades cap a aquest. Si no s'està produint cap transmissió de dades es procedeix a la suspensió del xip amb la comanda **pm** i els paràmetres 3, com a estat futur en suspensió, i 0 com a l'estat actual en funcionament a canviar. Després passen els segons definits al mètode de l'API i comença el procés d'arrencada amb la crida a sistema pm però amb els valors a la inversa. A continuació s'executen les crides a sistema **ioctl TCGETS**, **TCSETS** i **TCFLSH** les quals sol·liciten obtindre la configuració del port sèrie, configurar el port sèrie i realitzar el flush dels buffers d'entrada i de sortida. Aquestes crides seran executades un parell de cops més en el mateix ordre durant el temps de funcionament del GPS. Conseqüentment a les crides ioctl apareixen les crides equivalents per al port UART.

6.2 Establiment del consum de crides a sistema

6.2.1 Realització de proves

Després d'obtindre la relació entre el mètode requestLocationUpdates i les crides a sistema realitzades durant la suspensió i continuació del servei es tractarà de mesurar el consum energètic a la fase de funcionament i a la fase de suspensió del sistema GPS. També es comprovarà quina repercussió té el temps indicat al mètode de l'API, la cobertura del sistema, o la validesa de l'efemèride. Les proves a realitzar son les següents:

1. Amb cobertura:
 - 1.1 Terminal amb el xip GPS aturat.
 - 1.2 Terminal amb el xip GPS encès.
 - 1.3 Transició del xip GPS d'aturat a encès.
 - 1.4 Transició del xip GPS d'encès a aturat.
 - 1.5 Execució de l'aplicació fins a la obtenció del primer fix.
 - 1.6 Execució de l'aplicació focalitzant la mesura en els canvis d'estats cada x segons.
 - 1.6.1 Freqüència 0 segons (no espera).
 - 1.6.2 Freqüència 5 segons.
 - 1.6.3 Freqüència 10 segons.
 - 1.6.4 Freqüència 20 segons.
 - 1.6.5 Freqüència 40 segons.
 - 1.6.6 Freqüència 80 segons.
2. Sense cobertura:
 - 2.1 Terminal amb el xip GPS aturat.
 - 2.2 Terminal amb el xip GPS encès.
 - 2.3 Transició del xip GPS d'aturat a encès.
 - 2.4 Transició del xip GPS d'encès a aturat.
 - 2.5 Execució de l'aplicació fins a la obtenció del primer fix (comparació en el temps amb 1.5).
 - 2.6 Execució de l'aplicació focalitzant la mesura en els canvis d'estats cada x segons.
 - 2.6.1 Freqüència 0 segons (no espera).
 - 2.6.2 Freqüència 5 segons.
 - 2.6.3 Freqüència 10 segons.
 - 2.6.4 Freqüència 20 segons.
 - 2.6.5 Freqüència 40 segons.
 - 2.6.6 Freqüència 80 segons.
3. Durant l'execució amb cobertura, pas a no cobertura.

Totes les proves han sigut realitzades amb les mateixes condicions:

- Temperatura ambient d'entre 22 i 26 graus.
- Les proves realitzades amb cobertura a l'exterior sense cap tipus d'obstacle al voltant, i les proves realitzades sense cobertura a l'interior d'un garatge.
- També s'han realitzat les modificacions necessàries al programari per a mostrar la mateixa marca de temps UNIX epoch i s'han sincronitzat amb els mateixos servidors de temps tant el terminal mòbil com el ordinador on s'obtenien les mesures de potència.
- Bateria al 100% de càrrega i en descàrrega.
- Supressió de tots els serveis possibles corrent al sistema des de gestor d'aplicacions d'Android.
- Nivell de brillantor al màxim.
- Temps per a passar a inactivitat de 10 minuts.
- Temps de prova de 5 minuts a les proves amb diferents freqüències i del temps necessària a la resta.
- Tres mostres presses per cada prova per a la obtenció de valors mitjans.

6.2.2 Tractament de resultats

Una vegada obtinguts els resultats es passen aquests a un full de càlcul per al seu processament. A continuació es mostraran els resultats rellevants en el consum energètic:

6.2.2.1 Impacte del temps d'inhabilitació sobre el consum energètic

Primerament es mostra com afecta el paràmetre del temps de deshabilitació del sistema GPS sobre el consum energètic. Com a primer cas es té el valor de temps a 0 amb el que s'obtindran les actualitzacions de la localització geogràfica tan aviat com sigui possible i el sistema no passarà a un estat inactiu:

En aquest cas es pot observar un consum regular a partir de la primera localització obtinguda on no hi han variacions significatives. El càlcul del consum mig i de la variància serà:

$$\text{Consum mig} = 1,278 \text{ watts} * \text{hora}$$

$$\text{Variància} = 0,0371 (\text{watts} * \text{hora})^2$$

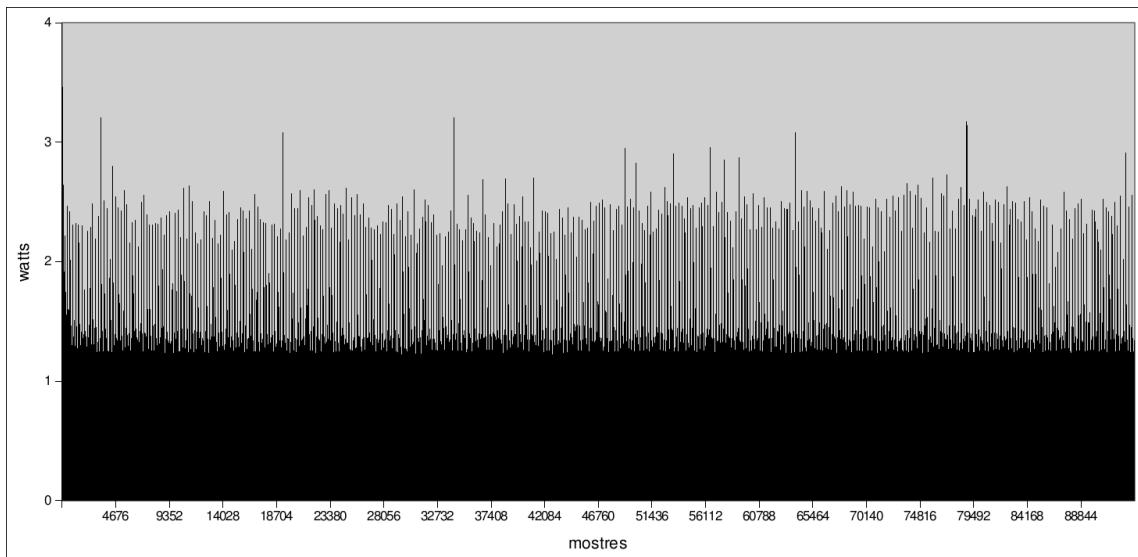


Figura 37: Gràfic d'obtenció de localització sense deshabilitament

Com a segon cas es té el valor de temps a 5 segons amb el que s'obtindran les actualitzacions de la localització geogràfica en el temps que duri aquesta obtenció i després el dispositiu quedarà aturat durant 5 segons:

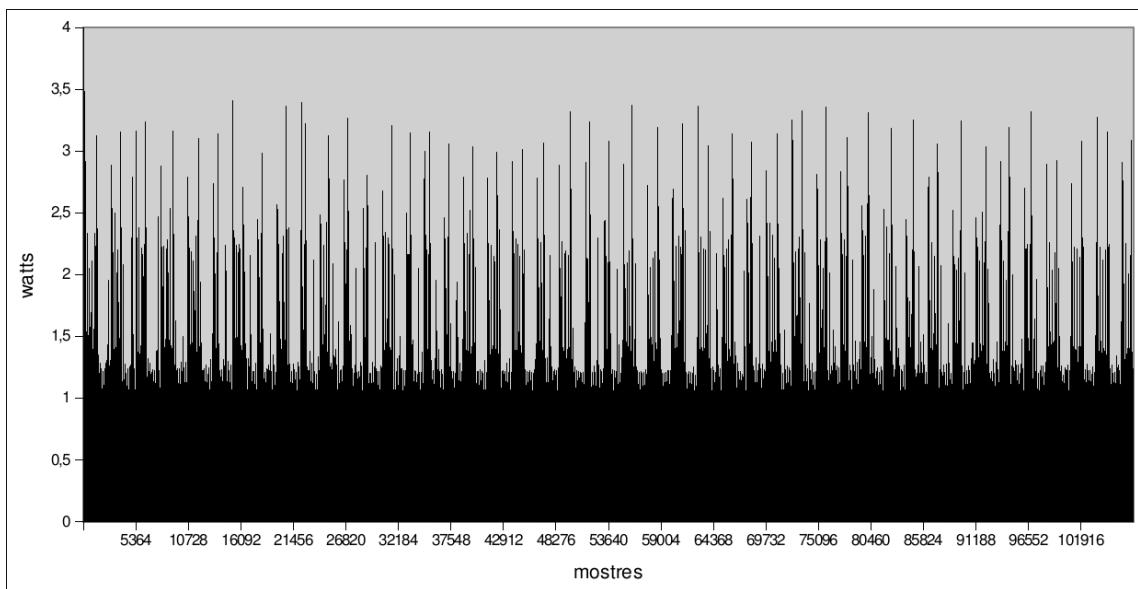


Figura 38: Gràfic d'obtenció de localització amb 5s de deshabilitament

En aquest cas es pot observar un consum variable segons l'estat en el que es troba el servei a partir de la primera localització obtinguda. El càlcul del consum mig i de la variància serà:

$$\text{Consum mig} = 1,227 \text{ watts * hora}$$

$$\text{Variància} = 0,0895 (\text{watts * hora})^2$$

<i>Tipus</i>	<i>5 segons deshabilitat</i>	<i>10 segons deshabilitat</i>
Global	1,227 watts * hora	1,147 watts * hora
Habilitat	1,39 watts * hora	1,216 watts * hora
Deshabilitat	1,151 watts * hora	1,017 watts * hora

Taula 5: Comparació de consums energètics amb 5s i 10s de deshabilitació del servei

Com es pot comprovar, entre el cas anterior i el cas actual ja hi ha hagut una millora en el consum energètic. De totes formes, encara que el consum ha disminuït, no és un temps suficient de deshabilitació com per a obtindre un consum deshabilitat i habilitat estable. A les mostres amb temps de deshabilitació igual o superior a 10 segons s'obtenen uns resultats similars tant al temps en el que el sistema està actiu com en el temps que el sistema està deshabilitat. A la següent taula es comparen els consums mitjans globals i parcials de les mostres amb 5 segons i amb 10 segons del sistema deshabilitat:

En quant al cas de no tindre cobertura GPS s'ha trobat que el consum és el mateix sense aquesta en qualsevol moment de l'aplicació. A continuació es mostren el gràfic, el consum mig i la variància creats quan no hi ha cobertura GPS:

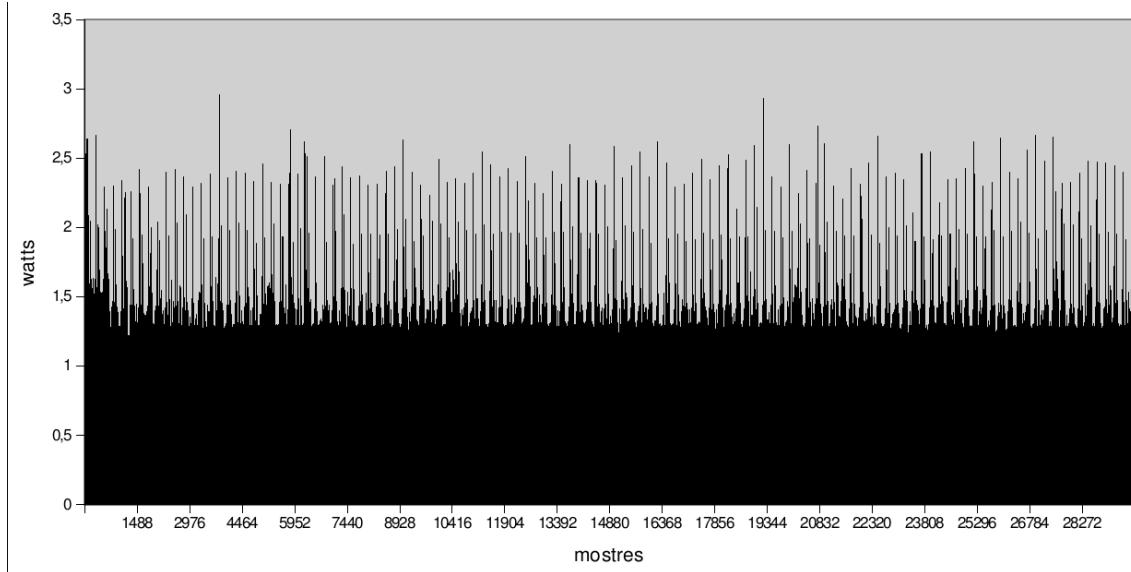


Figura 39: Gràfic de consum sense cobertura GPS

En aquest cas es pot observar un consum variable segons l'estat en el que es troba el servei a partir de la primera localització obtinguda. El càlcul del consum mig i de la variància serà:

$$\text{Consum mig} = 1,3577 \text{ watts * hora}$$

$$\text{Variància} = 0,0362 (\text{watts * hora})^2$$

Com a últim cas rellevant es té el consum energètic generat quan s'obté la primera actualització de la localització. Al tindre una efemèride vàlida es disminueix l'impacte al consum ja que es triguen aproximadament 5 segons de mitja en obtindre la primera actualització, en canvi si l'efemèride no és vàlida es necessita actualitzar i es triga un temps molt més ampli dependent del grau de cobertura que es tingui. A més, el consum mig per a obtindre l'efemèride és major al consum mig quan aquesta és vàlida. A continuació es comparen els dos casos:

<i>Tipus</i>	<i>Efemèride vàlida</i>	<i>Efemèride no vàlida</i>
Consum mig	1,273 watts * hora	1,587 watts * hora
Temps mig	5 segons	60 segons
Consum generat al primer “fix”	1,768 mW	26,45 mW

Taula 6: Comparació de consum generat a conseqüència de l'estat de l'efemèride

6.3 Conclusions

De totes les proves realitzades s'extreu que:

1. El consum amb el xip GPS encès o activat és el mateix o no s'arriba a apreciar. Fins que una aplicació no l'utilitza no realitza cap tipus de consum addicional.
2. El fet d'activar o desactivar el GPS a través de la interfície d'usuari no repercuteix al consum, és únicament un petit pic quasi instantani.
3. A l'hora d'obtindre el primer fix és important tenir en compte quan temps ha passat des de l'última utilització del sistema GPS. Si ha passat un temps suficient com per a que l'efemèride no sigui vàlida (aproximadament 2 hores) serà necessari obtindre-la de nou arribant a trigar el procés aproximadament 50 segons amb un bon nivell de cobertura, en cas contrari la obtenció del primer fix sol trigar uns 5 segons.
4. En quant al mètode requestLocationUpdates és rellevant que, a l'hora de fixar el temps d'inactivitat que passa entre fix i fix i tenint en compte el consum sense utilitzar aquest temps d'inabilitació (tan aviat com sigui possible), la tècnica comença a ser efectiva quan es deixen aproximadament 5 segons de temps per a l'estat desactivat.
5. Tal i com s'explica a les especificacions del xip GPS aquest funciona amb un cicle de 3 estats diferents amb el mode de funcionament Trickle Power Mode i es poden apreciar si observem el consum que es té al principi del cicle amb el Track Mode. Si aquest estat s'allarga, a conseqüència de la falta de cobertura, consumirà molt més el xip GPS en promig. Es pot arribar a aïllar el temps de CPU Mode fixant-ho i definint un consum mig.

Capítol 7. Conclusions

Amb aquest projecte he pogut donar el primer pas de la meva carrera cap als sistemes operatius obtenint coneixements més sòlids de Linux i aprenent el funcionament d'Android. Ha sigut determinant en la meva decisió de continuar estudiant en aquests camps en un futur pròxim i en la idea de treballar com a la recerca dels últims avenços tecnològics d'aquestes àrees.

Els requisits formatius que es pretenien amb aquest projecte han sigut complerts en cadascuna de les àrees en les que s'ha treballat i ha resultat satisfactori des del punt de vista personal.

Degut a que es va començar aquest projecte amb pocs coneixements en les matèries amb les que es treballa no es va definir formalment l'objectiu sinó que van haver de passar unes setmanes per a plasmar aquest com a rumb a prendre per a la finalització del projecte. Finalment es va fixar un objectiu assequible per al temps del que es disposava i amb aspiracions de futur per a la continuació d'aquest projecte per hom o per altra persona.

L'objectiu final de mesurar el consum d'una aplicació Android que utilitzés el sistema GPS s'ha complert satisfactoriament obtenint un procediment i un sistema capaç de ser exportat a qualsevol altre element del sistema amb el que es vulgui treballar.

7.1 Treball futur

Durant la realització del projecte ha sorgit la idea de realitzar en un futur una aplicació que, tenint les dades de consum energètic emmagatzemades per a un dispositiu en concret i havent-hi establert la relació amb uns patrons de crides al sistema, mostri el consum immediat utilitzat per el xip GPS i la quantitat de bateria utilitzada des del començament de la seva monitorització.

Capítulo 8. Bibliografía

[http://en.wikipedia.org/wiki/Code division multiple access](http://en.wikipedia.org/wiki/Code_division_multiple_access)
http://alumni.cs.ucr.edu/~saha/stuff/cdma_gps.htm
[http://en.wikipedia.org/wiki/Chip %28CDMA%29](http://en.wikipedia.org/wiki/Chip_%28CDMA%29)
http://en.wikipedia.org/wiki/GPS_signals
http://xn--gnuscultura-dbb.eu/compartits/cefire-on dara/4_funcionament_del_sistema_gps.html
<http://www.kowoma.de/en/gps/index.htm>
<http://developer.android.com/reference/android/location/package-summary.html>
<http://es.wikipedia.org/wiki/Wardriving>
<http://www.csr.com/products/25/sirfstariv-gsd4t>
<http://es.scribd.com/l4rrakin/d/4533340-Low-Power-Mode-Application-Note-GPSMS1-GPSMS1E-and-GPSPS1E-GPS-G1X00003>
http://en.wikipedia.org/wiki/C_standard_library
[http://en.wikipedia.org/wiki/Dalvik %28software%29](http://en.wikipedia.org/wiki/Dalvik_%28software%29)
http://en.wikipedia.org/wiki/Register_machine
http://ca.wikipedia.org/wiki/Programari_intermediari
http://es.wikipedia.org/wiki/N%C3%BAcleo_Linux
<http://es.wikipedia.org/wiki/Android>
<http://developer.android.com/guide/basics/what-is-android.html>
<http://developer.android.com/guide/topics/fundamentals.html>
<http://developer.android.com/guide/topics/providers/content-providers.html>
<http://developer.android.com/guide/topics/resources/index.html>
<http://developer.android.com/reference/android/app/Activity.html>
http://es.wikipedia.org/wiki/Compilador_cruzado
http://en.wikipedia.org/wiki/Java_Native_Interface
<http://developer.android.com/sdk/ndk/overview.html>
http://es.wikipedia.org/wiki/Efecto_Hall
http://es.wikipedia.org/wiki/Sensor_de_efecto_Hall
http://es.wikipedia.org/wiki/Circuito_RC
<http://www.ti.com/product/ina169>
<http://ebixio.com/blog/2011/01/03/the-android-ipc-system/>
http://openhealth.morfeo-project.org/wiki/index.php/Android_Manager_Service

<http://tldp.org/LDP/tlk/ipc/ipc.html>
<http://es.wikipedia.org/wiki/RPC>
<http://www.slideshare.net/opersys/android-internals-7623360>
http://elinux.org/Android_Kernel_Features
http://elinux.org/Android_Binder
http://en.wikipedia.org/wiki/General_Purpose_Input/Output
http://www.kandroid.org/online-pdk/guide/power_management.html
<http://marakana.com/static/courseware/android/internals/index.html>
http://en.wikipedia.org/wiki/HAL_%28software%29
http://www.360doc.com/content/12/0117/10/3700464_179849603.shtml
<http://www.linuxjournal.com/article/6331>
<http://source.android.com/source/building-kernels.html>
<http://www.mjmwired.net/kernel/Documentation/serial/driver>
<http://www.mjmwired.net/kernel/Documentation/tty.txt>
http://en.wikibooks.org/wiki/Serial_Programming/termio
http://linux.die.net/man/4/tty_ioctl