

Universidad de Montevideo
Facultad de Ingeniería

Proyecto UMix



Desarrollo de un sistema de búsqueda de supervivientes de catástrofes

Informe del
Proyecto Final de Carrera
Versión 2.5

Juan Brenes
Sofía Silva

Montevideo
27 de mayo de 2010



Dedicatoria

“Me es imposible comenzar una dedicatoria sin pensar en mi familia, esas personas que siempre están ahí con sus capas invisibles haciendo con su magia que todo funcione. Con cada uno de ellos estoy eternamente agradecido, pero por su cercanía quiero resaltar a mi madre, a mi hermana, Osvaldo y a mis abuelos.

Tampoco puedo olvidar a las personas “culpables” de haber llegado a la Universidad de Montevideo que son Larry Stensas y mi abuela Adela Wittemberger, que aún ya no estando físicamente se las arregla para seguir aconsejándome.

También tengo que reconocer que no hubiera llegado hasta donde estoy sin haber contado con unos compañeros de clase que resultaron ser un grupo humano entrañable, el cual me apoyó en todo lo que me tocó vivir con ellos.

Mis amigos, esa familia elegida, fueron y serán una fuente inagotable de inspiración y por eso merecen su mención en este espacio. Agradezco cada día que me ha tocado compartir mi vida con ellos, tanto a aquellos que siempre están con su sonrisa cotidiana como a aquellos que aparecen ocasionalmente y con su varita mágica dan un toque que cambia el mundo. Sin desmerecer a nadie, quiero mencionar especialmente a María Pía Aschieri que, entre otras cosas, ha sido una amiga invaluable.

Finalmente quiero agradecer a mis compañeros del Canal 10, un grupo de trabajo el cual tuve la fortuna de un día haberme integrado y que me ha ayudado en mil y una formas con mi proyecto y mi carrera.”

Juan Brenes



“En primer lugar quiero agradecer a mi familia y amigos, quienes me acompañaron durante estos cinco años de facultad y durante toda mi vida. Ellos me apoyaron siempre y me ayudaron a salir adelante en los momentos más difíciles.

También quiero agradecer a mis compañeros, a aquellos con los que compartí todas las materias y a aquellos con los que solo compartí algunas. Todos dejaron en mí una huella y gracias a todos ellos pude aprender infinidad de cosas.

En particular quiero agradecer a mi madre, quien me apoyó emocionalmente siempre, sobre todo en los momentos más difíciles en los que parecía que ya no me quedaban fuerzas.

También quiero agradecer de forma especial a mi padre, quien se las ingenió para ser un integrante más del equipo de este proyecto, ayudándonos con su manualidad e ingenio a que el prototipo de este proyecto pudiera existir.

No puedo dejar de mencionar a mi novio, Jonathan, quien escuchó cada uno de los problemas que fueron surgiendo en este proyecto y me prestó atención aunque no supiera de qué le estaba hablando. También quiero agradecer a su padre, Mario quien entendiendo un poco más del tema aportó muchas ideas.

Por último, pero no menos importantes, quiero darles las gracias a mis compañeros de IBM, quienes participaron en este proyecto aportando información y colaborando de distintas maneras.

A todas estas personas: Muchas gracias!! Este proyecto también es de ustedes.”

Sofía Silva



Reconocimientos

El proyecto UMix no hubiera sido posible sin la ayuda de una cantidad de personas, de las cuales aquí citaremos todas aquellas a las que nuestra corta memoria nos permita recordar en este momento.

En primer lugar los asesores de proyecto, el Ing. Marcelo Abreu y el Ing. Thomas Hobbins, que brindaron su experiencia en la gestión práctica del proyecto de forma que todo resultara más simple.

En segundo lugar queremos reconocer la ayuda que nos brindó todo el personal administrativo de la facultad, que con su labor nos facilitaron las herramientas que nos permitieron cursar la carrera y llevar a buen término el proyecto. Particularmente queremos destacar al Ing. Rafael Sotelo, que por su cercanía como coordinador de Telemática es a quien tuvimos más presente durante la carrera.

También queremos mencionar en este espacio a todos los profesores que con sus conocimientos y su apoyo nos ayudaron a progresar como estudiantes y como personas.

Finalmente hay que destacar a aquellas personas que colaboraron directamente con el proyecto, entre las cuales se encuentran la Ing. Luciana Briozzo, a la que agradecemos por el préstamo de la placa BeagleBoard en la cual se basó nuestro proyecto; al Ing. Aldo Castagna, gerente del área de planificación técnica de Antel, quien nos proporcionó el receptor GPS que se utilizó en este proyecto y al Ing. Diego Duran, quien gestionó este préstamo y, por último, pero no menos importante, queremos reconocer también a aquellos que con su consejo técnico nos orientaron en el desarrollo de este proyecto, fundamentalmente a los empleados de Fablet&Bertoni Pocitos.



Sobre este documento

Este documento es el Informe Final del Proyecto de Fin de Carrera. En el mismo se incluye el marco teórico asociado a las tareas realizadas para cumplir con los objetivos del proyecto, como también se incluye una descripción del desarrollo de cada una de dichas tareas. Por otro lado, se incluye una descripción de la metodología de trabajo empleada.

Historial de cambios

Versión	Fecha	Descripción	Autor
1.0	7 de noviembre de 2009	Formato del documento	Sofía Silva
1.1	5 de diciembre de 2009	Versión inicial del documento	Sofía Silva
1.2	17 de diciembre de 2009	Se completaron los capítulos Marco Teórico y Desarrollo	Juan Brenes Sofía Silva
2.0	25 de febrero de 2010	Se agrega información teórica y práctica sobre la segunda etapa del proyecto.	Juan Brenes Sofía Silva
2.1	4 de marzo de 2010	Se realizan modificaciones a nivel de redacción y estructura.	Sofía Silva
2.2	26 de marzo de 2010	Se agrega información teórica y práctica sobre las tareas restantes.	Juan Brenes Sofía Silva
2.3	21 de abril de 2010	Se realizan correcciones a nivel de la profundidad del contenido y otras correcciones.	Sofía Silva
2.4	12 de mayo de 2010	Correcciones en referencias bibliográficas.	Sofía Silva
2.5	27 de mayo de 2010	Se agrega Abstract en inglés.	Sofía Silva



Resumen Ejecutivo

Hoy en día, en el mundo suceden catástrofes con demasiada frecuencia. Esto genera enormes cantidades de heridos cuyo rescate suele ser una tarea ardua y compleja. La búsqueda de supervivientes implica, entre otras cosas, tener suficiente personal a disposición para recorrer las áreas de catástrofe. Por otro lado, el hecho de que sean personas las que realizan la búsqueda implica un riesgo para éstas por el hecho de estar recorriendo zonas peligrosas.

En este contexto, este proyecto tiene como principal objetivo contribuir a simplificar el rescate de personas que hayan sobrevivido a catástrofes, desarrollando un sistema automatizado que realice las tareas de búsqueda de supervivientes.

Previo a este proyecto, el equipo de proyecto había realizado trabajos que le permitieron adquirir cierta experiencia en lenguajes de programación tales como Assembler, Java y C; en sistemas operativos Linux y en otras áreas relacionadas a este proyecto. Esta experiencia fue fundamental para el cumplimiento de los objetivos del proyecto UMix.

Respecto de los componentes de este proyecto, existen muchos sistemas embebidos que utilizan el kernel de Linux como base para realizar sus funciones. En particular, existen muchos proyectos de robótica implementados con dicho sistema operativo utilizando específicamente la placa BB. Por otro lado, el puerto de expansión de la placa BB ha sido frecuentemente utilizado como una interfaz de comunicación con un PIC y también existen proyectos que implementan interfaces de control de motores y manejo de sensores desde un PIC. Sin embargo, no se ha encontrado ningún proyecto que integre todos estos elementos dándoles una aplicación similar a la del proyecto UMix.

En nuestro país, Uruguay, hasta el momento no se han llevado a cabo proyectos similares al proyecto UMix. Uno de los motivos es que Uruguay no es un país en donde ocurran importantes catástrofes con frecuencia. Sin embargo, sí existen, a nivel internacional, proyectos en esta área, aunque estos son propietarios, mucho más complejos y de alto costo. Desde un principio, la idea del proyecto UMix fue mantener bajos costos y modularidad para que pueda ser mejorado por otros equipos en algún futuro.

El objetivo general de este proyecto es contribuir al desarrollo de tecnologías que faciliten la tarea de búsqueda de supervivientes en áreas de catástrofes.

Los objetivos específicos de este proyecto son los siguientes:

- Implementar funcionalidades básicas de búsqueda de personas.
- Desarrollar un módulo que permita la comunicación inalámbrica con una computadora personal utilizando protocolos de red estandarizados. Esto permitirá la fácil administración y manipulación del móvil.
- Incluir un módulo de posicionamiento que haga uso de un dispositivo GPS estándar para determinar la ubicación del móvil en todo momento y en particular cuando se detecta un superviviente.



- Diseñar el sistema de movimiento de forma tal que la plataforma física de movimiento sea fácilmente intercambiable.
- Simular el módulo de detección de vida con algún otro sensor diferente.

Durante el semestre en que se cursó la Asignatura Proyecto de Telemática también se cursó la asignatura Gestión de Proyectos, cuyo objetivo era gestionar el proyecto de fin de carrera. Para cumplir con este objetivo se realizaron diversas tareas, algunas de las cuales concluyeron con un entregable. Estas tareas incluyeron la planificación del proyecto, creando un cronograma con desglose de tareas y asignación de los integrantes a las mismas; la formulación de un documento de alcance y la redacción del Plan de Proyecto.

Para la asignación de tareas se tuvo en cuenta la modularidad del proyecto, lo cual permitió dividir las tareas de forma de poder avanzar en paralelo con los distintos módulos. Es decir, que el avance de uno de los integrantes del equipo condicionaba lo mínimo posible al avance del otro integrante.

El proyecto también requiere la realización de tareas técnicas para poder implementar los distintos módulos que comprende el sistema. Para llevar a cabo esta implementación se planificó la dedicación de cierto tiempo a la investigación teórica de los temas pertinentes, para luego pasar a la implementación del módulo y, por último, al testeo.

Para los módulos asociados al PIC se dividió la implementación y el testeo en dos etapas: primero se implementó y testeó en una simulación (utilizando el programa Proteus) y luego se armaron los circuitos físicos y se testearon. Esta última etapa, a su vez se dividió en dos: primero se armaron los circuitos sobre ProjectBoards y luego de verificado su correcto funcionamiento, se soldaron sobre placas universales de pertinax.

Como resultado de la primera etapa del proyecto, se logró un prototipo que cumplía con las funcionalidades que se especificaron al comienzo del proyecto:

- Compilación de Linux para la placa BeagleBoard (BB). Se compiló el kernel de Ubuntu para el procesador utilizado por la placa BB (OMAP3530 que se basa en el procesador ARM Cortex-A8)
- Comunicación inalámbrica entre la placa BB y un PC. Conexión de la placa BB a Internet a través de un modem inalámbrico 3G y un servidor DNS dinámico (DDNS), lo cual permite que la placa BB sea accedida desde cualquier PC conectado a Internet.
- Comunicación entre la placa BB y el PIC. Envío de directivas de movimiento desde la placa BB hacia el PIC y notificación de detección de vida y de obstáculos desde el PIC hacia la placa BB.
- Instalación de un servidor Apache con módulo para PHP en la placa BB.
- Control y monitoreo del sistema. Interfaz web que permite enviar directivas y recibir notificaciones.
- Ejecución de un programa escrito en el lenguaje C con rutinas de acceso a una base de datos SQL.



- Módulo de movimiento: Diseño de circuitos y desarrollo de rutinas para controlar un motor de continua y un motor paso a paso desde un PC.
- Simulación de sensor de vida: Diseño de un sensor infrarrojo (IR) para simular el sensor de vida. Este sensor detecta un tono de 1 kHz de frecuencia y notifica a la placa BB. La placa BB almacena esta información y notifica al usuario final.
- Detección de obstáculos: Diseño de circuitos y desarrollo de rutinas que permiten la detección de obstáculos y la notificación correspondiente a la placa BB, la cual se encarga de notificar al usuario final.

Durante la segunda etapa del proyecto se tomó la decisión de no realizar las tareas asociadas al movimiento automatizado del vehículo. Se planteó a los tutores que se estimaba que el tiempo disponible no era suficiente para implementar el módulo encargado de realizar dichas tareas y estos estuvieron de acuerdo en que este cambio no modificaba el alcance del proyecto.

Omitiendo esta parte del proyecto, los restantes objetivos fueron cumplidos. Es decir, además de los objetivos que se cumplieron en la primera etapa del proyecto, el prototipo final cumple con las siguientes funcionalidades:

- Kernel Linux OMAP: Compilación de un kernel más adaptado al procesador de la placa BB y que, además, consume menos recursos.
- Módulo de posicionamiento: Integración al sistema de un receptor GPS y desarrollo del código necesario para conocer la posición del móvil en todo momento y convertir las coordenadas del móvil en Datum WGS84 al Datum Yacaré.
- Módulo de persistencia: Creación de una base de datos SQL y desarrollo del código necesario para respaldar la información de los puntos que recorrió el móvil. Se almacenan las coordenadas de cada punto, la fecha y hora en que el móvil pasó por ese punto y un flag indicando si detectó o no un superviviente en ese punto.
- Interfaz Web: Desarrollo completo de la interfaz Web del proyecto, la cual permite ver el estado de los sensores, enviar directivas de movimiento, ver la posición actual del móvil y los puntos en los que se detectaron supervivientes y realizar configuraciones de red, entre otras funcionalidades.
- Fuente de alimentación: Diseño de la alimentación del sistema utilizando reguladores de la familia 78xx para obtener los voltajes requeridos por los distintos componentes.

En la primera etapa del proyecto se cumplieron los objetivos fijados para la entrega del primer prototipo. Sin embargo, algunas tareas importantes para el proyecto fueron pospuestas para poder dedicar más tiempo a la documentación y a los testeos necesarios para esa entrega. El equipo asumió la responsabilidad de dicha postergación y minimizó sus consecuencias reajustando los tiempos de la planificación inicial.

Los atrasos mencionados anteriormente se debieron principalmente a falta de desarrollo en el kernel de Linux utilizado, sobre todo en los temas relacionados a la comunicación inalámbrica.



Por otro lado, durante las tareas de testeo de circuitos se experimentaron dificultades debido a que en la práctica los circuitos no se comportan exactamente igual a como lo hacen en la simulación.

También se experimentaron dificultades al intentar comunicar la placa BB con el PIC como consecuencia de que los mismos manejan lógicas con distintos niveles de voltaje. Teniendo en cuenta que fue imposible encontrar en el mercado local los circuitos integrados necesarios para solucionar este problema, se decidió implementar los circuitos necesarios para convertir los niveles lógicos utilizando transistores y resistencias.

Es importante tener en cuenta que al momento de la planificación del proyecto no se tuvo en cuenta la gran demanda de tiempo requerida para la elaboración de documentos. Este fue otro motivo por el cual fue necesario postergar tareas.

En la segunda etapa del proyecto se encontraron dificultades al intentar establecer una comunicación con un receptor GPS con interfaz USB. Estas dificultades se debieron a errores en cierto driver utilizado por la aplicación GPS Daemon. Dicho driver dejó de tener soporte por lo que se optó por utilizar un receptor GPS con interfaz serial.

Por otro lado, se decidió convertir las coordenadas obtenidas del receptor GPS a coordenadas planas de forma de simplificar la automatización del movimiento del móvil. Esta conversión fue más compleja de lo previsto y fue necesario obtener y adaptar código de aplicaciones genéricas.

Como se mencionó en la sección anterior, no se cumplió con el objetivo asociado a implementar un módulo responsable del movimiento automatizado del vehículo. Esto se debió principalmente a la falta de experiencia por parte del equipo de proyecto en la planificación de proyectos de esta índole. Esta falta de experiencia llevó a que no se tuvieran en cuenta las dificultades que podían surgir y que llevarían a demoras excesivas.

Exceptuando el objetivo mencionado en el párrafo anterior, durante la segunda etapa del proyecto se logró cumplir de forma satisfactoria con todos los objetivos que habían quedado pendientes de la primera etapa.



Abstract

Nowadays, catastrophes happen all around the world too frequently. The result is: lots of people hurt whose rescue is usually a complex and hard task. The search for survivors implies, among other things, having enough staff available to go round catastrophes' areas. Besides, the fact of being human beings the ones who do the search implies a risk for them due to the fact of going around dangerous areas.

In this context, this project's main objective is to contribute to simplify the rescue of survivors from catastrophes, deploying an automated system which performs the tasks associated to the search for survivors.

Before starting this project, the members of the team had carried out several other projects that allowed them to acquire some experience on programming languages such as Assembler, Java and C; on Linux operating systems and on other fields related to this project. Such experience was essential for the achievement of UMix project's objectives.

As regards the components of this project, there exist lots of embedded systems which use the Linux kernel as a basis for performing their functions. In particular, there exist lots of robotics projects implemented using the above mentioned operating system on the BeagleBoard (BB). On the other hand, the expansion port of the BB has been frequently used as a communications interface with a PIC microcontroller and there also exist projects implementing motor control and sensor handling interfaces. However, it hasn't been found any project which integrates all these items together and applies them as UMix project does.

In our country, Uruguay, projects similar to UMix project haven't been carried out so far. One of the reasons is that Uruguay is not a country where important catastrophes happen frequently. However, there do exist, internationally, projects in this area, although they are much more complex and expensive. From the beginning, the focus of the UMix project was to keep costs as low as possible and be modular so that it can be improved by other teams in the not to distant future.

The general objective of this project is to contribute to the deployment of technologies which facilitate the task of searching for survivors at catastrophes' areas.

The specific objectives of this project are those which follow:

- Implement basic functionalities of people search.
- Deploy a module which allows wireless communication with a personal computer using standardized network protocols. This will allow an easy administration and control of the vehicle.
- Include a positioning module which uses a standard GPS device to determine the position of the vehicle anytime and, in particular, when a survivor is detected.
- Design the motion system so that the physical platform of motion can be easily changed.
- Simulate the life detection module with another sensor.



The main objective of Project Management, a subject of the Telematic Engineering career, was to manage the final project. To achieve this objective, different tasks were performed, some of them concluding with a deliverable document. These tasks included the planning of the project, making a schedule with a breakdown of tasks and the allocation of the members of the team to those tasks; the formulation of a document of scope and the writing of the Project Plan.

For the allocation of tasks, the modularity of the Project was taken into account, which allowed dividing the tasks up so that each member could progress with each module in parallel. In other words, the progress of one of the team members determined as less as possible the progress of the other member.

The Project also requires the performing of technical tasks to be able to implement the different modules covered by the system. To carry this implementation out, the dedication of some time to theoretically investigate the relevant topics was planned, to go on to the implementation of the module later on and, finally, to the testing.

For the modules associated to the PIC microcontroller, the implementation and the testing were divided into two stages: firstly, the modules were implemented and tested on a simulation (using an application called Proteus) and then the physical circuits were assembled and tested. This last stage was also divided into two sub-stages: firstly, the circuits were assembled on ProjectBoards and, after verifying their correct operation, they were soldered on pertinax universal boards.

As a result of the first stage of the project, a prototype was obtained. This prototype complied with the functionalities which had been specified at the beginning of the project:

- Linux compilation for the BeagleBoard (BB). The Ubuntu kernel was compiled for the processor used by the BB (OMAP3530 which is based on the ARM Cortex-A8)
- Wireless communication between the BB and a PC. Internet connection for the BB through a wireless 3G modem and a dynamic DNS, which allows the BB to be accessed from any PC connected to the Internet.
- Communication between the BB and the PIC microcontroller. Sending of movement directives from the BB to the PIC and notification of life and obstacles detection from the PIC to the BB.
- Installation of an Apache Server with a PHP module on the BB.
- Control and monitoring of the system. Web interface which allows sending directives and receiving notifications.
- Deployment of a program written in C language which includes routines to access an SQL data base.
- Motion module. Design of circuits and deployment of routines to control a DC motor and a stepper motor from a PC.
- Simulation of the “life” sensor. Design of an infrared (IR) sensor to simulate the “life” sensor. This sensor detects a 1 kHz tone and notifies the BB. The BB stores the information and notifies the end user.



- Obstacle detection. Design of circuits and deployment of routines which allow the detection of obstacles and the corresponding notification to the BB, which is in charge of notifying the end user.

During the second stage of the project, the team decided not to perform the tasks associated to the automated motion of the vehicle. The team communicated the tutors that the available time could no be enough to implement the module in charge of performing those tasks and they agreed that this change wouldn't modify the scope of the project.

Having left this part out, the remaining objectives were achieved. That is to say, apart from the achieved objectives of the first stage of the project, the final prototype complies with the functionalities mentioned below:

- Linux OMAP kernel. Compilation of a kernel more adequate to the BB processor which consumes fewer resources.
- Positioning module. Integration of a GPS receiver to the system and deployment of the necessary source code to know the position of the vehicle anytime and to convert to coordinates from the WGS84 Datum to the Yacare Datum.
- Persistency module. Creation of an SQL data base and deployment of the necessary source code to back up the information of the points gone round by the vehicle. The information stored is: the coordinates of each point, the date and time when the vehicle was there and a flag to indicate if a survivor was detected on that point or not.
- Web interface. Complete deployment of the Web interface of the project, which allows watching the status of the sensors, sending movement directives, watching the present position of the vehicle and the point where survivors where detected and configure the network, among other functionalities.
- Power supply. Design of the supply for the system using regulators of the 78xx family to obtain the voltages needed by the different components.

In the first stage of the project, all the objectives planed for the hand over of the first prototype were achieved. However, some important tasks had to be postponed to dedicate more time to the documentation and the testing required for this hand over. The team took on the responsibility of such postponement and minimized its consequences readjusting the dates stated at the initial schedule.

The delays mentioned before were mainly due to the lack of deployment of the used Linux kernel, especially on the topics related to the wireless communication.

On the other hand, during the circuit testing tasks, some difficulties were experimented due to the fact that in practice the circuits don't behave the same as they do at the simulation.

Other difficulties were also experimented when trying to communicate the BB with the PIC as a result of both of them using logics with different voltage levels. Taking into account that, in the local market it was impossible to find the integrated circuits necessary to solve



this problem, the team decided to implement the necessary circuits to convert the logic levels using transistors and resistors.

It's important to bear in mind that when the team planed the project, the big demand of time needed for the elaboration of the documents wasn't taken into account. This was another reason for postponing tasks.

In the second stage of the project, some difficulties were found when trying to establish a communication with an USB GPS receiver. These difficulties were due to errors on a driver used by the application GPS Daemon. This driver ceased to be supported so the team decided to use a GPS receiver with serial interface.

On the other hand, the team decided to convert the coordinates obtained by the GPS receiver to plane coordinates so that the automation of the vehicle motion was simplified. This conversion was much more complex than it had been anticipated and it was necessary to obtain and adapt source code of generic applications.

As it was mentioned in the previous section, the objective associated to the implementation of a module in charge of the automated motion of the vehicle wasn't achieved. This was mainly due to the lack of experience by the team on the planning of this kind of projects. This lack of experience resulted in not taking into account the difficulties that could arise and which would lead to excessive delays.

Except from the objective mentioned in the previous paragraph, during the second stage of the project, all the other pending objectives from the first stage were satisfactorily achieved.



Índice

DEDICATORIA	I
RECONOCIMIENTOS.....	III
SOBRE ESTE DOCUMENTO	IV
HISTORIAL DE CAMBIOS.....	IV
RESUMEN EJECUTIVO	V
ABSTRACT	IX
ÍNDICE.....	XIII
ÍNDICE DE FIGURAS.....	XXIII
ÍNDICE DE TABLAS.....	XXVI
1 INTRODUCCIÓN DEL PROYECTO FINAL DE CARRERA.....	1
1.1 PROYECTO UMix.....	1
1.2 OBJETIVOS.....	1
1.2.1 <i>Objetivo general del proyecto</i>	1
1.2.2 <i>Objetivos específicos del proyecto</i>	2
1.3 ANTECEDENTES	3
1.4 JUSTIFICACIÓN DE IMPACTO	4
1.5 GRUPOS DE INTERÉS	5
2 MARCO TEÓRICO DEL PROYECTO FINAL DE CARRERA.....	6
2.1 SISTEMAS EMBEBIDOS.....	6
2.2 PLACA BEAGLEBOARD.....	7
2.2.1 <i>Reseña</i>	7
2.2.2 <i>Características generales de la placa</i>	8
2.2.3 <i>Procesador</i>	8
2.2.4 <i>Memoria</i>	9
2.2.5 <i>Conecotor SD/MMC 6 en 1</i>	9
2.2.6 <i>Puerto USB-OTG</i>	10
2.2.7 <i>Puerto de Expansión</i>	11
2.2.7.1 <i>Protocolo GPIO</i>	12
2.2.8 <i>Puerto RS232</i>	13
2.2.9 <i>Alimentación</i>	13
2.3 SISTEMA OPERATIVO.....	15
2.3.1 <i>Linux</i>	15



2.3.2	<i>Kernel de Ubuntu</i>	18
2.4	COMUNICACIÓN	19
2.4.1	<i>Enlace 3G</i>	19
2.4.2	<i>Enlace Ethernet</i>	20
2.5	MICROCONTROLADOR PIC.....	21
2.5.1	<i>Arquitectura</i>	23
2.5.2	<i>CPU y ALU</i>	23
2.5.3	<i>Oscilador</i>	24
2.5.4	<i>Memoria</i>	24
2.5.4.1	Memoria de programa	24
2.5.4.2	Memoria de datos.....	26
2.5.4.2.1	Registro STATUS	28
2.5.4.2.2	Registro OPTION_REG	28
2.5.4.2.3	Registro PCON	28
2.5.4.3	Memoria de datos EEPROM	28
2.5.5	<i>Modo SLEEP</i>	28
2.5.6	<i>Interrupciones</i>	30
2.5.6.1	Interrupción externa	31
2.5.6.2	Interrupción por cambio en el Puerto B.....	31
2.5.7	<i>Periféricos</i>	33
2.5.7.1	Puertos de entrada/salida.....	33
2.5.7.2	Timer2	34
2.6	HERRAMIENTAS DE DESARROLLO	36
2.6.1	<i>Entorno de desarrollo integrado MPLAB</i>	36
2.6.2	<i>Paquete MPASM</i>	36
2.6.3	<i>Simulador Proteus</i>	37
2.6.4	<i>Programador PICSTART Plus</i>	37
2.7	CONTROL DE MOTORES'.....	38
2.7.1	<i>Motor de continua (Tracción)</i>	38
2.7.2	<i>Motor PaP (Dirección)</i>	39
2.7.3	<i>Driver L293D</i>	41
2.8	MANEJO DE SENsoRES	43
2.8.1	<i>Simulación del sensor de vida</i>	43
2.8.1.1	Micrófono.....	43
2.8.1.1.1	Micrófono electret	44
2.8.1.1.2	Alimentación del micrófono	45
2.8.1.2	Filtros activos	45
2.8.1.3	Detector de tono	48



2.8.1.4	Relé	49
2.8.1.5	Compuerta NOT	50
2.8.2	<i>Sensor infrarrojo</i>	51
2.8.2.1	Luz infrarroja	51
2.8.2.2	Emisor IR	51
2.8.2.3	Receptor IR.....	51
2.8.3	<i>Sensores de obstáculos</i>	51
2.8.3.1	Microswitches	52
2.8.3.2	Flip-Flop RS asíncrono	53
2.9	COMUNICACIÓN ENTRE EL PIC Y LA PLACA BB	55
2.10	FUENTE DE ALIMENTACIÓN	56
2.11	GLOBAL POSITIONING SYSTEM (GPS)	58
2.11.1	<i>Introducción</i>	58
2.11.2	<i>Funcionamiento</i>	58
2.11.3	<i>Fiabilidad de los datos</i>	59
2.12	DATOS NMEA	61
2.12.1	<i>Introducción</i>	61
2.12.2	<i>Sentencias NMEA</i>	61
2.12.3	<i>Interfaz de hardware</i>	62
2.13	INFORMACIÓN DE POSICIÓN	63
2.13.1	<i>Introducción</i>	63
2.13.2	<i>Tipos de Datum</i>	63
2.13.3	<i>De la esfera al plano</i>	64
2.13.3.1	Coordenadas Cartográficas	64
2.13.3.2	Tipos de proyecciones.....	64
2.13.3.2.1	Proyección Cilíndrica.....	64
2.13.3.3	Sistema Gauss-Krüger	65
2.13.3.4	Parámetros.....	65
2.13.3.4.1	Parámetros lineales	66
2.13.3.4.2	Parámetros angulares.....	66
2.14	WGS84	67
2.15	COORDENADAS GEODÉSICAS VS COORDENADAS GEOCÉNTRICAS	68
2.16	MANEJO DE DISPOSITIVOS EN LINUX	69
2.16.1	<i>Udev</i>	69
2.17	GESTIÓN DE PAQUETES EN LINUX	71
2.17.1	<i>Apt-get</i>	71
2.18	GPS DAEMON	72
2.19	SOCKETS	73



2.19.1	<i>Tipos de sockets</i>	74
2.19.2	<i>Dominios</i>	74
2.20	LENGUAJE C'	76
2.20.1	<i>Introducción</i>	76
2.20.2	<i>Manejo de arreglos</i>	76
2.20.3	<i>Asignación dinámica de memoria</i>	77
2.21	GNU COMPILER COLLECTION (GCC).....	79
2.21.1	<i>Etapas de compilación</i>	79
2.21.1.1	Pre-procesamiento.....	79
2.21.1.2	Compilación.....	79
2.21.1.3	Ensamblado.....	79
2.21.1.4	Enlazado	80
2.21.2	<i>Proceso completo</i>	80
2.21.3	<i>Otras opciones de compilación</i>	80
2.22	CONSTRUCCIÓN DE BIBLIOTECAS ESTÁTICAS'	82
2.22.1	<i>Creación de archivos</i>	82
2.22.2	<i>Compilación y enlace</i>	83
2.22.3	<i>Utilización de la biblioteca</i>	83
2.23	API C DE MySQL	85
2.23.1	<i>Tipos de datos</i>	85
2.23.2	<i>Funciones</i>	85
2.23.2.1	mysql_init()	85
2.23.2.2	mysql_real_connect().....	86
2.23.2.3	mysql_real_query()	86
2.23.2.4	mysql_store_result()	87
2.23.2.5	mysql_num_rows().....	87
2.23.2.6	mysql_affected_rows().....	87
2.23.2.7	mysql_fetch_row()	88
2.23.2.8	mysql_free_result()	88
2.23.2.9	mysql_close()	88
2.24	LENGUAJES PARA DESARROLLO WEB	89
2.24.1	<i>HTML'</i>	89
2.24.2	<i>PHP</i>	89
2.24.3	<i>Java Script</i>	90
2.25	CARGADOR DE ARRANQUE	91
2.25.1	<i>Tipos de cargadores de arranque</i>	91
2.25.1.1	Gestor de arranque de segunda etapa.....	91
2.25.1.2	Gestor de arranque Flash	91



2.25.1.3	Gestor de arranque de red.....	92
2.25.1.4	Otros tipos de secuencia de arranque.....	92
2.25.2	<i>Cargador de arranque en la placa BeagleBoard</i>	92
2.25.2.1	uBoot.....	92
2.25.2.2	Secuencia de arranque.....	92
2.25.2.3	Puerto de expansión	93
3	MARCO METODOLÓGICO DEL PROYECTO FINAL DE CARRERA.....	94
3.1	INTRODUCCIÓN A LA METODOLOGÍA	94
3.2	GESTIÓN DEL PROYECTO	94
3.2.1	<i>Planificación</i>	94
3.2.2	<i>Control de avance</i>	95
3.2.3	<i>Redacción de documentos</i>	95
3.2.3.1	Documento de Alcance	95
3.2.3.2	Plan de Proyecto	95
3.3	TAREAS TÉCNICAS.....	96
3.3.1	<i>Investigación</i>	96
3.3.2	<i>Implementación y testeo</i>	96
4	DESARROLLO DEL PROYECTO.....	98
4.1	TAREAS INICIALES	99
4.1.1	<i>Definición del tema</i>	99
4.1.2	<i>Estudio de la placa</i>	99
4.1.3	<i>Definición del formato de documentos</i>	99
4.1.4	<i>Redacción del plan de proyecto</i>	99
4.2	COMPILACIÓN E INSTALACIÓN DE ARM EABI UBUNTU 9.04	100
4.2.1	<i>Fundamento de la elección del sistema operativo</i>	100
4.2.2	<i>Conexión Serial</i>	101
4.2.3	<i>Actualización de uBoot</i>	102
4.2.4	<i>Compilación del kernel (Método 1)</i>	102
4.2.4.1	Adaptación de la PC huésped.....	103
4.2.4.2	Generación de imágenes y kernel	104
4.2.5	<i>Compilación del kernel (Método 2)</i>	105
4.2.5.1	Adaptación de la PC huésped	105
4.2.5.2	Selección de la rama del kernel.....	106
4.2.5.3	Obtención del código fuente	107
4.2.5.4	Crear y modificar una configuración	107
4.2.5.5	Compilación del kernel.....	108
4.2.6	<i>Particionado y armado de la SD</i>	108



4.2.6.1	Particionado con gparted	108
4.2.6.2	Instalación de uBoot/uImage en la partición de la SD.....	111
4.2.6.3	Instalación de la partición ext2/ext3 de la SD	111
4.2.7	<i>Configuración de uBoot</i>	112
4.3	INSTALACIÓN DE APLICACIONES Y ACTUALIZACIÓN DEL KERNEL.....	113
4.3.1	<i>Conexión de red cableada</i>	113
4.3.2	<i>Conexión cableada a Internet</i>	115
4.3.3	<i>Servidor SSH</i>	116
4.3.4	<i>Actualización del kernel</i>	117
4.3.5	<i>Servidor Apache</i>	117
4.3.6	<i>DNS Dinámico</i>	118
4.4	CONEXIÓN INALÁMBRICA A INTERNET.....	119
4.5	DISEÑO Y CONSTRUCCIÓN DEL MÓVIL.....	121
4.5.1	<i>Motor de continua</i>	121
4.5.2	<i>Motor PaP</i>	121
4.6	IMPLEMENTACIÓN DEL MÓDULO DE MOVIMIENTO.....	123
4.6.1	<i>Estudio del módulo PWM</i>	123
4.6.2	<i>Desarrollo del código</i>	123
4.6.3	<i>Testeo del módulo de movimiento</i>	129
4.7	IMPLEMENTACIÓN MÓDULO COMUNICACIÓN PIC - PLACA BB	131
4.7.1	<i>Investigación de los protocolos del puerto de expansión</i>	131
4.7.2	<i>Desarrollo protocolo de comunicación</i>	131
4.7.2.1	A nivel del PIC.....	131
4.7.2.2	A nivel de la placa BB	132
4.7.3	<i>Integración con otros módulos</i>	133
4.7.3.1	A nivel del PIC.....	133
4.7.3.2	A nivel de la placa BB	134
4.7.4	<i>Testeo protocolo comunicación</i>	135
4.7.4.1	A nivel del PIC.....	135
4.7.4.2	A nivel de la placa BB	135
4.7.4.3	PIC y placa BB	135
4.8	IMPLEMENTACIÓN MÓDULO DE DETECCIÓN DE SUPERVIVIENTES.....	138
4.8.1	<i>Estudio de integración de simulación de sensor de vida</i>	138
4.8.2	<i>Desarrollo de código de sensor de vida</i>	140
4.8.3	<i>Testeo sensor de simulación de vida</i>	140
4.8.4	<i>Sensor infrarrojo</i>	142
4.9	ADAPTACIÓN E INTEGRACIÓN DE MÓDULO DE DETECCIÓN DE OBSTÁCULOS	145
4.9.1	<i>Estudio de integración de sensores de obstáculos</i>	145



4.9.2	<i>Desarrollo del código para evasión de obstáculos y movimiento automatizado</i>	145
4.10	IMPLEMENTACIÓN DEL MÓDULO DE POSICIONAMIENTO GLOBAL	146
4.10.1	<i>Investigación de interacción con GPS</i>	146
4.10.2	<i>Obtención de la posición del receptor GPS desde un PC de pruebas</i>	147
4.10.2.1	Instalación y configuración de GPS Daemon	147
4.10.2.2	Detección del receptor GPS en el PC de pruebas	147
4.10.2.3	Ejecución de GPSD en el PC de pruebas	149
4.10.2.4	GPSD vs. GPSBabel	153
4.10.2.5	Receptor GPS serial	154
4.10.3	<i>Desarrollo del código de posicionamiento</i>	156
4.10.3.1	Obtención de coordenadas	156
4.10.3.2	Conversión de coordenadas al Datum Yacaré.....	159
4.10.4	<i>Compilación del código de posicionamiento</i>	160
4.10.5	<i>Testeo del módulo de posicionamiento en el PC de pruebas</i>	160
4.10.6	<i>Obtención de la posición del receptor GPS desde la placa BeagleBoard</i>	162
4.10.6.1	Detección del receptor GPS en la placa BB	162
4.10.6.2	Ejecución de gpsd en la placa BB.....	166
4.10.7	<i>Construcción de biblioteca estática</i>	168
4.10.8	<i>Testeo del módulo de posicionamiento en la placa BB</i>	169
4.11	IMPLEMENTACIÓN DEL MÓDULO DE PERSISTENCIA	171
4.11.1	<i>Diseño de la base de datos</i>	171
4.11.2	<i>Instalación de LAMPP en el PC de pruebas</i>	172
4.11.3	<i>Creación de la base de datos en el PC de pruebas</i>	172
4.11.4	<i>Obtención del script de creación de la base de datos</i>	173
4.11.5	<i>Creación de usuario en el PC de pruebas</i>	173
4.11.6	<i>Creación de la base de datos en el servidor definitivo</i>	174
4.11.7	<i>Creación de usuario en el servidor definitivo</i>	176
4.11.8	<i>Integración del módulo de persistencia’</i>	176
4.11.8.1	Desarrollo del código del módulo de persistencia	176
4.11.8.2	Compilación del código del módulo de persistencia	178
4.11.8.3	Testeo del funcionamiento del código en el PC de pruebas	179
4.11.8.4	Migración del módulo de persistencia a la placa BB	181
4.11.8.5	Testeo del funcionamiento del código en la placa BB.....	182
4.11.8.6	Construcción de biblioteca estática	183
4.12	DISEÑO DE LA FUENTE DE ALIMENTACIÓN.....	185
4.13	COMPILACIÓN KERNEL LINUX OMAP.....	187
4.13.1	<i>Análisis del cambio de kernel</i>	187
4.13.1.1	Justificación	187
4.13.1.2	Impacto	187



4.13.2	<i>Configuración del entorno</i>	188
4.13.2.1	Fundamento	188
4.13.2.2	Instalación de emdebian	188
4.13.3	<i>Compilación</i>	189
4.13.3.1	Obtención del código fuente.....	189
4.13.3.2	Configuración del kernel y los módulos	189
4.13.3.3	Compilación.....	190
4.13.3.4	Instalación	191
4.14	ADAPTACIÓN Y COMPILACIÓN DE UBOOT	192
4.14.1	<i>Cambios en el código de uBoot</i>	192
4.14.2	<i>Toolchain</i>	193
4.14.3	<i>Compilación de uBoot</i>	193
4.14.4	<i>Instalación de uBoot</i>	193
4.15	DESARROLLO DE LA INTERFAZ WEB	194
4.15.1	<i>Fundamentos de la elección del lenguaje</i>	194
4.15.2	<i>Desarrollo</i>	195
4.15.2.1	Adaptaciones a la plataforma - Elevación de privilegios	195
4.15.2.2	GoogleMaps	196
4.15.2.3	Diagrama de entidades	197
4.15.2.4	Diagrama de secuencia	198
4.15.2.5	Movimiento	199
4.16	INTEGRACIÓN DE LOS MÓDULOS.....	200
5	CONCLUSIONES	202
6	RECOMENDACIONES	205
7	BIBLIOGRAFÍA.....	206
8	ANEXO I – PLAN DEL PROYECTO FINAL DE CARRERA.....	209
8.1	ACTA DEL PROYECTO	209
8.1.1	<i>Nombre del Proyecto</i>	209
8.1.2	<i>Fecha de comienzo</i>	209
8.1.3	<i>Áreas de conocimientos / procesos</i>	209
8.1.4	<i>Áreas de aplicación (sector / actividad)</i>	210
8.1.5	<i>Fecha de inicio del Proyecto</i>	210
8.1.6	<i>Fecha tentativa de finalización del Proyecto</i>	210
8.2	OBJETIVOS DEL PROYECTO	210
8.2.1	<i>Objetivo General</i>	210
8.2.2	<i>Objetivos Específicos</i>	210



8.3	DESCRIPCIÓN DEL PRODUCTO	211
8.4	NECESIDAD DEL PROYECTO	211
8.5	ALCANCE.....	212
8.5.1	<i>Incluido</i>	212
8.5.2	<i>No incluido</i>	212
8.6	JUSTIFICACIÓN DEL IMPACTO (APORTE Y RESULTADOS ESPERADOS)	212
8.7	ENTREGABLES.....	212
8.8	RESTRICIONES.....	213
8.9	IDENTIFICACIÓN DE GRUPOS DE INTERÉS (DIRECTOS E INDIRECTOS).....	213
9	ANEXO II – ACTAS	214
9.1	ACTA N° 1.....	214
9.2	ACTA N° 2.....	215
9.3	ACTA N° 3.....	216
9.4	ACTA N° 4.....	217
9.5	ACTA N° 5.....	218
9.6	ACTA N° 6.....	219
10	ANEXO III – CONFIGURACIÓN DE HYPERTERMINAL	220
11	ANEXO IV – CÓDIGO DEL PIC	222
12	ANEXO V – CÓDIGO DEL MÓDULO DE POSICIONAMIENTO	232
13	ANEXO VI – PROCEDIMIENTOS EN LAMPP	242
13.1	INSTALACIÓN DE LAMPP	242
13.2	CREACIÓN DE LA BASE DE DATOS	243
13.3	OBTENCIÓN DEL SCRIPT DE CREACIÓN DE LA BASE DE DATOS	247
13.4	CREACIÓN DE USUARIO	249
14	ANEXO VII - SCRIPT DE CREACIÓN DE LA BASE DE DATOS	254
15	ANEXO VIII - CÓDIGO DEL MÓDULO DE PERSISTENCIA	255
16	ANEXO IX – CÓDIGO DE GPSSTORE	259
17	ANEXO X – CÓDIGO DE GPSCONVERT	261
18	ANEXO XI – CÓDIGO INTERFAZ WEB	262
19	ANEXO XII – DESARROLLO DE DRIVERS EN LINUX	274
19.1	INTRODUCCIÓN.....	274
19.2	MARCO TEÓRICO	274



19.2.1	<i>Kernel</i>	274
19.2.2	<i>Módulos</i>	275
19.2.2.1	Tipos de módulos	275
19.2.2.2	Módulos vs Aplicaciones	276
19.3	IMPLEMENTACIÓN.....	276
19.3.1	<i>Código</i>	276
19.3.2	<i>Compilación y ejecución</i>	277
19.4	PROBLEMAS ENCONTRADOS	277
20	ANEXO XIII - ACRÓNIMOS Y ABREVIATURAS	278



Índice de Figuras

FIGURA 1.1 - ESQUEMA MODULAR	3
FIGURA 2.1 - CONEXIONADO CONECTOR SD/MMC 6 EN 1	10
FIGURA 2.2 - CONEXIONADO PUERTO USB-OTG	11
FIGURA 2.3 - CONEXIONADO PUERTO DE EXPANSIÓN	12
FIGURA 2.4 - PUERTO SERIAL RS232.....	13
FIGURA 2.5 - CONEXIONADO ALIMENTACIÓN BB.....	14
FIGURA 2.6 - DIFERENCIAS ENTRE NÚCLEOS.....	16
FIGURA 2.7 - MAPA DE MEMORIA DE LINUX.....	17
FIGURA 2.8 - DIAGRAMA DE BLOQUES PIC 16F877A.....	22
FIGURA 2.9 - MAPA DE LA MEMORIA DE PROGRAMA Y STACK	25
FIGURA 2.10 - MAPA DE BANCOS DE MEMORIA	27
FIGURA 2.11 - DIAGRAMA DE BLOQUES PINES RB7:RB4	32
FIGURA 2.12 - TÍPICO PUERTO DE ENTRADA/SALIDA.....	33
FIGURA 2.13 - DIAGRAMA DE BLOQUES TIMER2	34
FIGURA 2.14 - REGISTRO T2CON	35
FIGURA 2.15 - MOTOR DE CONTINUA CON FUENTE SIMÉTRICA.....	39
FIGURA 2.16 - MOTOR DE CONTINUA CON PUENTE H	39
FIGURA 2.17 - MOTOR PAP BIPOLAR.....	40
FIGURA 2.18 - MOTOR PAP UNIPOLAR	40
FIGURA 2.19 - ESQUEMA INTERNO L293D	41
FIGURA 2.20 - ESQUEMA MICRÓFONO ELECTRET.....	44
FIGURA 2.21 - ALIMENTACIÓN DE MICRÓFONO ELECTRET	45
FIGURA 2.22 - FILTRO PASIVO VS. FILTRO ACTIVO	46
FIGURA 2.23 - FILTRO ACTIVO PASA BANDA.....	46
FIGURA 2.24 - RESPUESTA EN FRECUENCIA DE UN FILTRO PASA BANDA	47
FIGURA 2.25 - DIAGRAMA DE BLOQUES LM567	48
FIGURA 2.26 - ECUACIONES LM567	48
FIGURA 2.27 - ANCHO DE BANDA VS AMPLITUD DE LA SEÑAL DE ENTRADA.....	49
FIGURA 2.28 - SÍMBOLO COMPUERTA NOT	50
FIGURA 2.29 - ESQUEMA MICROSWITCH	52
FIGURA 2.30 - FLIP-FLOP RS CON COMPUERTAS NOR	54
FIGURA 2.31 - TRANSISTOR COMO INTERRUPTOR	55
FIGURA 2.32 - DIAGRAMA DE BLOQUES REGULADOR DE VOLTAJE.....	56
FIGURA 2.33 - REGULACIÓN DE VOLTAJE	57



FIGURA 2.34 - PROYECCIÓN CILÍNDRICA	65
FIGURA 2.35 - COORDENADAS GEODÉSICAS Y GEOCÉTRICAS	68
FIGURA 4.1 - ESKUEMA MODULAR	98
FIGURA 4.2 - CONEXIÓN SERIAL	101
FIGURA 4.3 - CONEXIÓN ESTABLECIDA	102
FIGURA 4.4 - NÚCLEOS DISPONIBLES	106
FIGURA 4.5 – PARTICIÓN FAT32 EXISTENTE	109
FIGURA 4.6 – ELIMINAR PARTICIÓN FAT32 EXISTENTE	109
FIGURA 4.7 – CREACIÓN DE NUEVA PARTICIÓN FAT32	110
FIGURA 4.8 - CREACIÓN DE NUEVA PARTICIÓN EXT3	110
FIGURA 4.9 – OPCIÓN MANAGE FLAGS	110
FIGURA 4.10 – PARTICIÓN BOOTEABLE	111
FIGURA 4.11 - PUERTO USB	113
FIGURA 4.12 - CONEXIÓN DE RED	114
FIGURA 4.13 - ESKUEMA PIC Y DRIVERS	122
FIGURA 4.14 - DIAGRAMA DE FLUJO RUTINA MAIN	124
FIGURA 4.15 - DIAGRAMA DE FLUJO RUTINA ATENCIONINTERRUPCION	125
FIGURA 4.16 - DIAGRAMA DE FLUJO RUTINA ATENCIONSENSORES	126
FIGURA 4.17 - DIAGRAMA DE FLUJO RUTINA RECIBIRDIRECTIVA	127
FIGURA 4.18 - DIAGRAMA DE FLUJO SUBRUTINAS DIRECTIVAS	128
FIGURA 4.19 - DIAGRAMA DE FLUJO SUBRUTINAS SENSORES	128
FIGURA 4.20 - TRANSISTOR COMO INTERRUPTOR	136
FIGURA 4.21 - DIVISOR DE VOLTAJE	137
FIGURA 4.22 - CIRCUITO SENSOR DE SONIDO	139
FIGURA 4.23 - FILTRO ACTIVO DE PRIMER ORDEN	141
FIGURA 4.24 - RESPUESTA EN FRECUENCIA FILTRO ACTIVO	142
FIGURA 4.25 - SENSOR IR	143
FIGURA 4.26 - ESKUEMA EMISOR IR	144
FIGURA 4.27 - PINES DE LA SALIDA DE DATOS DEL RECEPTOR GPS	154
FIGURA 4.28 - SALIDA MINICOM	165
FIGURA 4.29 - PANTALLA PRINCIPAL PHPMyADMIN	173
FIGURA 5.1 - FUENTE DE ALIMENTACIÓN	185
FIGURA 5.2 - DIAGRAMA DE ENTIDADES	197
FIGURA 5.3 - DIAGRAMA DE SECUENCIA	198
FIGURA 5.4 - DIAGRAMA DE SECUENCIA MOVIMIENTO	199
FIGURA 9.1 - ESKUEMA MODULAR	211
FIGURA 11.1 - NUEVA CONEXIÓN HYPERTERMINAL	220



FIGURA 11.2 - ESTABLECIMIENTO DE LA CONEXIÓN SERIAL	220
FIGURA 11.3 - PROPIEDADES DE LA CONEXIÓN.....	221
FIGURA 11.4 - CONEXIÓN ESTABLECIDA.....	221
FIGURA 14.1 - LAMPP	243
FIGURA 14.2 - PANTALLA PRINCIPAL PHPMyADMIN	243
FIGURA 14.3 - NUEVA BASE DE DATOS	244
FIGURA 14.4 - BASE DE DATOS CREADA	244
FIGURA 14.5 - NUEVA TABLA	245
FIGURA 14.6 - CAMPOS DE LA TABLA	245
FIGURA 14.7 - INFORMACIÓN DE LOS CAMPOS.....	246
FIGURA 14.8 - CAMPOS AGREGADOS	247
FIGURA 14.9 - TAB EXPORTAR.....	247
FIGURA 14.10 - EXPORTAR LA BASE DE DATOS.....	248
FIGURA 14.11 - DESCARGAR ARCHIVO	248
FIGURA 14.12 - NOMBRE Y DESTINO DEL ARCHIVO	249
FIGURA 14.13 - TAB PRIVILEGIOS	249
FIGURA 14.14 - VISTA GLOBAL DE USUARIOS.....	250
FIGURA 14.15 - AGREGAR NUEVO USUARIO	250
FIGURA 14.16 - USUARIO CREADO	251
FIGURA 14.17 - PRIVILEGIOS ESPECÍFICOS.....	252
FIGURA 14.18 - PRIVILEGIOS OTORGADOS	252
FIGURA 14.19 - PRIVILEGIOS ACTUALIZADOS	253
FIGURA 20.1 – ROLES DEL KERNEL.....	274
FIGURA 20.2 - INSMOD Y RMMOD	275



Índice de Tablas

TABLA 2-1 - CARACTERÍSTICAS PLACA BB	8
TABLA 2-2 - SECUENCIA DE PASOS MOTOR PAP	40
TABLA 2-3 - TABLA DE VERDAD NOT	50
TABLA 2-4 - TABLA DE VERDAD FLIP-FLOP RS.....	53
TABLA 2-5 - TABLA DE VERDAD NOR	54
TABLA 2-6 - SEÑALES PUERTO DE EXPANSIÓN	93
TABLA 4-1 - RESUMEN DE FUNCIONES DE PINES DEL PIC.....	131
TABLA 4-2 - PINES GPIO	132
TABLA 4-3 - RESUMEN DE FUNCIONES DE PINES DE LA BB	133
TABLA 14-1 - CAMPOS DE LA TABLA PUNTOS.....	246



1 Introducción del Proyecto Final de Carrera

1.1 Proyecto UMix

El presente proyecto, “Proyecto UMix – Búsqueda de supervivientes”, consiste en la investigación necesaria para desarrollar un sistema automatizado de búsqueda de supervivientes de catástrofes y en el desarrollo de un prototipo de este sistema.

Para el desarrollo de este proyecto se utilizaron diversos dispositivos de ubicación, monitoreo y comunicación ya estandarizados. Se utiliza una placa de desarrollo BeagleBoard (BB) para la cual se compiló un sistema operativo Linux. Esta placa se integra con un sistema de comunicación inalámbrica estándar que permite que se pueda interactuar con el sistema del proyecto desde cualquier PC conectada a Internet. También se utiliza un receptor GPS de mano, el cual se conecta a la placa BB para enviarle la información de posición del móvil.

Por otro lado, se manejan microswitches como sensores que permiten detectar obstáculos; se simulan los sensores de vida con un sensor infrarrojo que detecta un tono con una frecuencia determinada y se utilizan circuitos integrados que implementan drivers para controlar los motores del vehículo.

Todo el sistema fue implementado en base a software, interfaces y plataformas libres, lo cual permite la fácil adaptación de este proyecto a otros sistemas.

1.2 Objetivos

1.2.1 Objetivo general del proyecto

El objetivo general de este proyecto es contribuir al desarrollo de tecnologías que faciliten la tarea de búsqueda de supervivientes en áreas de catástrofes. Para ello se utilizaron componentes estandarizados, de fácil adquisición y se recurrió a código open source, de forma de permitir futuros desarrollos sobre la base de este proyecto.



1.2.2 Objetivos específicos del proyecto

Para cumplir con el objetivo general de este proyecto se estableció como meta la implementación de funcionalidades básicas de búsqueda de personas, es decir, la construcción de un móvil que se desplace de forma autónoma en un área determinada e informe sobre los lugares específicos donde se encontraron posibles supervivientes. Para esto es necesario:

- Cargar un sistema operativo Linux sobre una placa de desarrollo BeagleBoard (BB). Se compiló el kernel Ubuntu adaptándolo a la placa BB.
- Desarrollar la lógica para el control de los motores y para la interacción con los sensores en un micro controlador PIC 16F877A.
- Utilizar circuitos integrados para controlar los sensores y los motores.
- Implementar sensores de obstáculos y sensor infrarrojo.
- Utilizar un motor de continua y un motor paso a paso para la tracción y la dirección del móvil respectivamente.
- Desarrollar un módulo en la placa BB que permita la comunicación inalámbrica con una computadora personal utilizando protocolos de red estandarizados. Esto permitirá la fácil administración y manipulación del móvil. Para esto se utilizó un modem inalámbrico Huawei E160. Para conectar este dispositivo a la placa BB fue necesario agregar un hub USB con alimentación independiente y puerto maestro.
- Incluir un módulo de posicionamiento que haga uso de un dispositivo GPS estándar para determinar la ubicación del móvil en todo momento y en particular cuando se detecta un superviviente.
- Diseñar el sistema de movimiento de forma tal que la plataforma física de movimiento sea fácilmente intercambiable.
- Simular el módulo de detección de vida con un sensor infrarrojo.



En la Figura 1.1 se presenta un esquema del proyecto, mostrándose los módulos del mismo, donde residen y cómo interactúan entre sí.

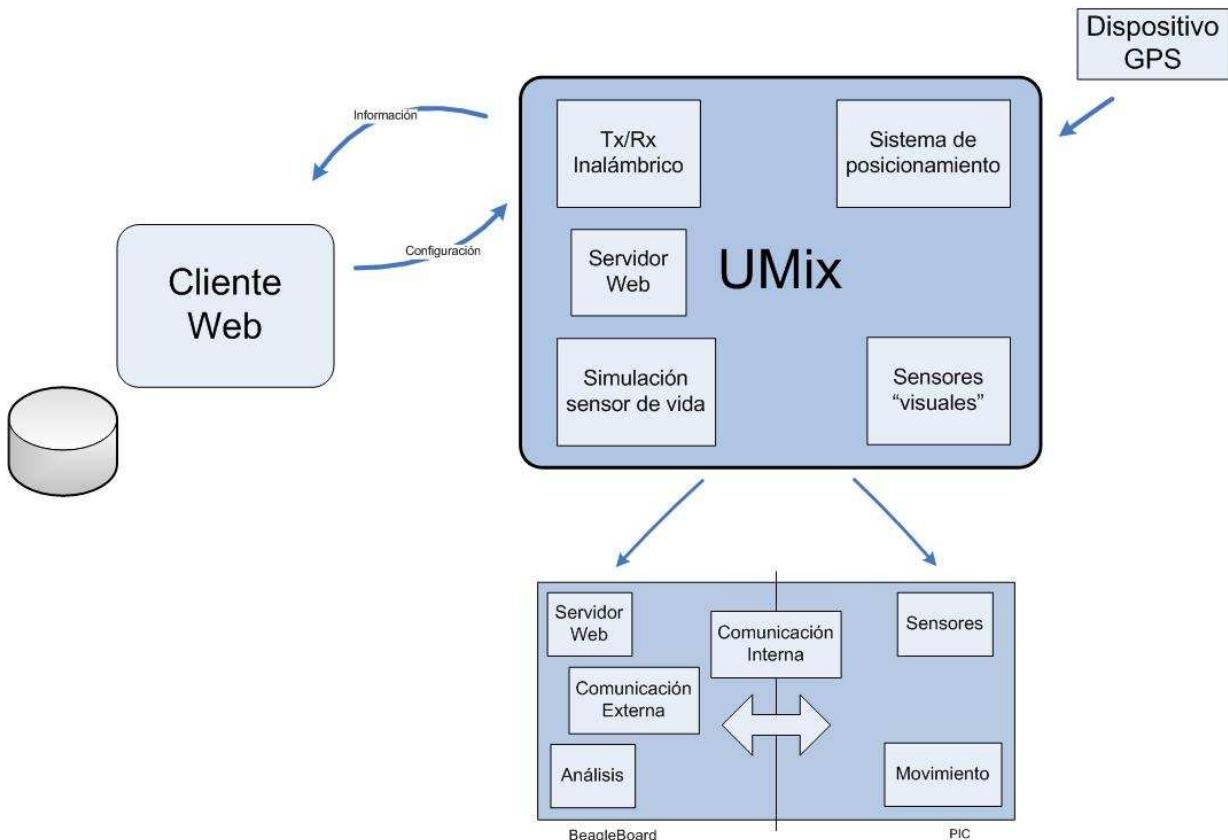


Figura 1.1 - Esquema modular

1.3 Antecedentes

Aplican como antecedentes del proyecto UMix, los trabajos realizados por el equipo de proyecto en distintas asignaturas de la carrera, tales como Programación, Sistemas Operativos y Laboratorio de Telemática, entre otros. Estos trabajos permitieron al equipo adquirir cierta experiencia en lenguajes de programación tales como Assembler, Java y C; en sistemas operativos Linux y en otras áreas relacionadas a este proyecto. Esta experiencia fue fundamental para el cumplimiento de los objetivos del proyecto UMix.

Respecto de los componentes de este proyecto, se investigó en busca de otros proyectos que hicieran uso de componentes similares o que tuvieran aplicaciones similares a las del proyecto UMix. Se tomó conocimiento de que existen muchos sistemas embebidos que



utilizan el kernel de Linux como base para realizar sus funciones. En particular, existen muchos proyectos de robótica implementados con dicho sistema operativo utilizando específicamente la placa BB.

Por otro lado, el puerto de expansión de la placa BB ha sido frecuentemente utilizado como una interfaz de comunicación con un PIC y también existen proyectos que implementan interfaces de control de motores y manejo de sensores desde un PIC.

Sin embargo, no se ha encontrado ningún proyecto que integre todos estos elementos dándoles una aplicación similar a la del proyecto UMix.

En nuestro país, Uruguay, hasta el momento no se han llevado a cabo proyectos similares al proyecto UMix. Uno de los motivos es que Uruguay no es un país en donde ocurran importantes catástrofes con frecuencia. Sin embargo, sí existen, a nivel internacional, proyectos en esta área, aunque estos son propietarios, mucho más complejos y de alto costo. Desde un principio, la idea del proyecto UMix fue mantener bajos costos y modularidad para que pueda ser mejorado por otros equipos en algún futuro.

1.4 Justificación de impacto

UMix es un proyecto que no pretende competir con sistemas automatizados de búsqueda que utilicen alta tecnología, sino que su objetivo es brindar una solución económica, sencilla y modular que pueda adaptarse a distintas realidades. Además pretende ser la base para futuros proyectos de investigación y desarrollo que se lleven a cabo en la misma área de conocimientos.

Si bien muchos de los componentes que se utilizan en la implementación de UMix ya existen, no existe ningún proyecto hasta el momento que las vincule y las integre en una solución de estas características.



Se espera que el valor agregado por la integración de todas las tecnologías y las herramientas desarrolladas permitan que el proyecto UMix tenga un gran impacto y aceptación en el mercado.

1.5 Grupos de interés

- Entes Gubernamentales que participen en la búsqueda de supervivientes
- ONGs u otras organizaciones que participen en la búsqueda de supervivientes
- Fuerzas armadas, ejércitos
- Grupos sociales que habiten en zonas de accidentes o catástrofes naturales frecuentes
- Empresas dedicadas a la venta de este tipo de tecnología
- Investigadores particulares de robótica y desarrolladores de sistemas automatizados
- Desarrolladores del kernel de Linux



2 Marco Teórico del Proyecto Final de Carrera

En este capítulo se establece un marco teórico como referencia para comprender las tareas desarrolladas como parte del proyecto y se detallan los conceptos claves utilizados. Solo se detallan los conceptos utilizados directamente para el desarrollo del proyecto.

2.1 Sistemas embebidos

Un sistema embebido es una computadora de poca capacidad, destinada a realizar una tarea específica en tiempo real. Estos sistemas deben su nombre a que, por lo general, forman la parte integral del hardware de un sistema más grande.

Su diseño suele ser muy flexible, de forma que se adapta a la gran mayoría de los requerimientos de los potenciales usuarios.

Estos sistemas son controlados por un procesador central que típicamente es implementado en base a micro controladores o procesadores digitales de señal.

La limitada función que realizan estos sistemas hace que se pueda reducir enormemente su consumo de energía y el espacio físico que ocupan.

En el caso del proyecto UMix, se eligió utilizar un sistema de este tipo de forma de facilitar la lógica relacionada con el manejo de protocolos de red, manejo de dispositivos GPS y el algoritmo de movimiento del móvil.



2.2 Placa BeagleBoardⁱ

Para el desarrollo del proyecto UMix se decidió utilizar la placa BeagleBoard (BB), en primer lugar por la gran potencialidad que brindan sus componentes y, en segundo lugar, debido a la posibilidad que tenía el equipo de proyecto de adquirir la misma. En esta sección se explican brevemente las características de la placa BB que fueron utilizadas en el proyecto.

2.2.1 Reseña

La placa BB es una plataforma basada en el procesador OMAP3530 (que a su vez se basa en el procesador ARM Cortex-A8) y diseñada específicamente para cumplir las especificaciones relacionadas con la comunidad “Open Source”.

Su implementación se realizó teniendo como objetivo disponer de una plataforma con una combinación mínima de herramientas que permitan al usuario experimentar el poder del OMAP3530, sin pretender en ningún momento obtener una plataforma de desarrollo completa. Muchas de las capacidades del OMAP3530 no son accesibles desde la placa BB. De todas formas, el diseño de la placa incluye diversas interfaces de comunicación que hacen que la misma sea ampliamente extensible, permitiendo así el desarrollo de diversas herramientas y aplicaciones.



2.2.2 Características generales de la placa

La Tabla 2-1 muestra las características de la placa:

Feature		
Processor	OMAP3530 ES3.0	
POP Memory	Micron	
	2Gb NAND (256MB)	2Gb MDDR SDRAM (256MB)
PMIC TPS65950	Power Regulators	
	Audio CODEC	
	Reset	
	USB OTG PHY	
Debug Support	14-pin JTAG	GPIO Pins
	UART	LEDs
PCB	3.1" x 3.0" (78.74 x 76.2mm)	6 layers
Indicators	Power	2-User
	PMU	
HS USB 2.0 OTG Port	Mini AB USB connector	
	TPS65950 I/F	
	MiniAB	
HS USB Host Port (Rev C2/3 Only)	Single USB HS Port	Up to 500ma Power
Audio Connectors	3.5mm	3.5mm
	L+R out	L+R Stereo In
SD/MMC Connector	6 in 1 SD/MMC/SDIO	4/8 bit support, Dual voltage
User Interface	1-User defined button	Reset Button
Video	DVI-D	S-Video
Power Connector	USB Power	DC Power
Expansion Connector (Not Populated)	Power (5V & 1.8V)	UART
	McBSP	McSPI
	I2C	GPIO
	MMC	PWM
2 LCD Connectors (Rev C2/3 Only)	Access to all of the LCD control signals plus I2C	3.3V, 5V, 1.8V

Tabla 2-1 - Características placa BB

2.2.3 Procesadorⁱⁱ

La placa BB incluye un procesador OMAP3530 versión ES3.0, empaquetado en un POP (Package On Package) de .4mm. Esta técnica de empaquetamiento consiste en situar las memorias (NAND y SDRAM) de la placa sobre el procesador, haciendo que el mismo no sea visible para el usuario.



2.2.4 Memoria

Como se mencionó anteriormente, la placa BB posee únicamente dos tipos de memorias: 2Gb NAND x 16 (256 MB) y 2Gb MDDR SDRAM x32 (256 MB @166 MHz).

Las memorias NAND son un tipo de memoria Flash de bajo costo y gran capacidad. Las memorias Flash se caracterizan por mantener los datos aún sin disponer de una fuente de energía. En particular, las memorias NAND se diferencian de las otras memorias Flash por su implementación en base a la compuerta lógica a la que debe su nombre.

La Mobile Double Data Rate (MDDR) SDRAM es un tipo de memoria de doble transferencia de datos pensada para dispositivos móviles. El objetivo de esta tecnología es ahorrar espacio físico dentro de la placa y mejorar el consumo total de energía en comparación con la versión normal de este tipo de memoria.

Al ser una memoria RAM, esta memoria depende de una fuente de energía para mantener la información que guarda.

Como se mencionó anteriormente, los dos tipos de memoria citados son los únicos que vienen nativamente con la placa BB, sin embargo, es posible ampliar la memoria NAND de la placa mediante la utilización de una tarjeta SD ó mediante algún disco USB conectado mediante un hub USB (con alimentación independiente) al puerto USB-OTG de la placa BB. Cabe destacar que la viabilidad de esta última opción depende del sistema operativo que se utilice.

2.2.5 Conector SD/MMC 6 en 1

Una de las interfaces de expansión que dispone la BB es el conector SD/MMC 6 en 1, el cual soporta el estándar MMC4.0 (MMC+) y puede ser utilizado para bootear la placa BB desde tarjetas MMC ó SD. Todas las tarjetas de 4 u 8 bits funcionan para esta interface, sin embargo, únicamente las de 4 bits se pueden utilizar para bootear.



La Figura 2.1 describe el conexionado de la interface.

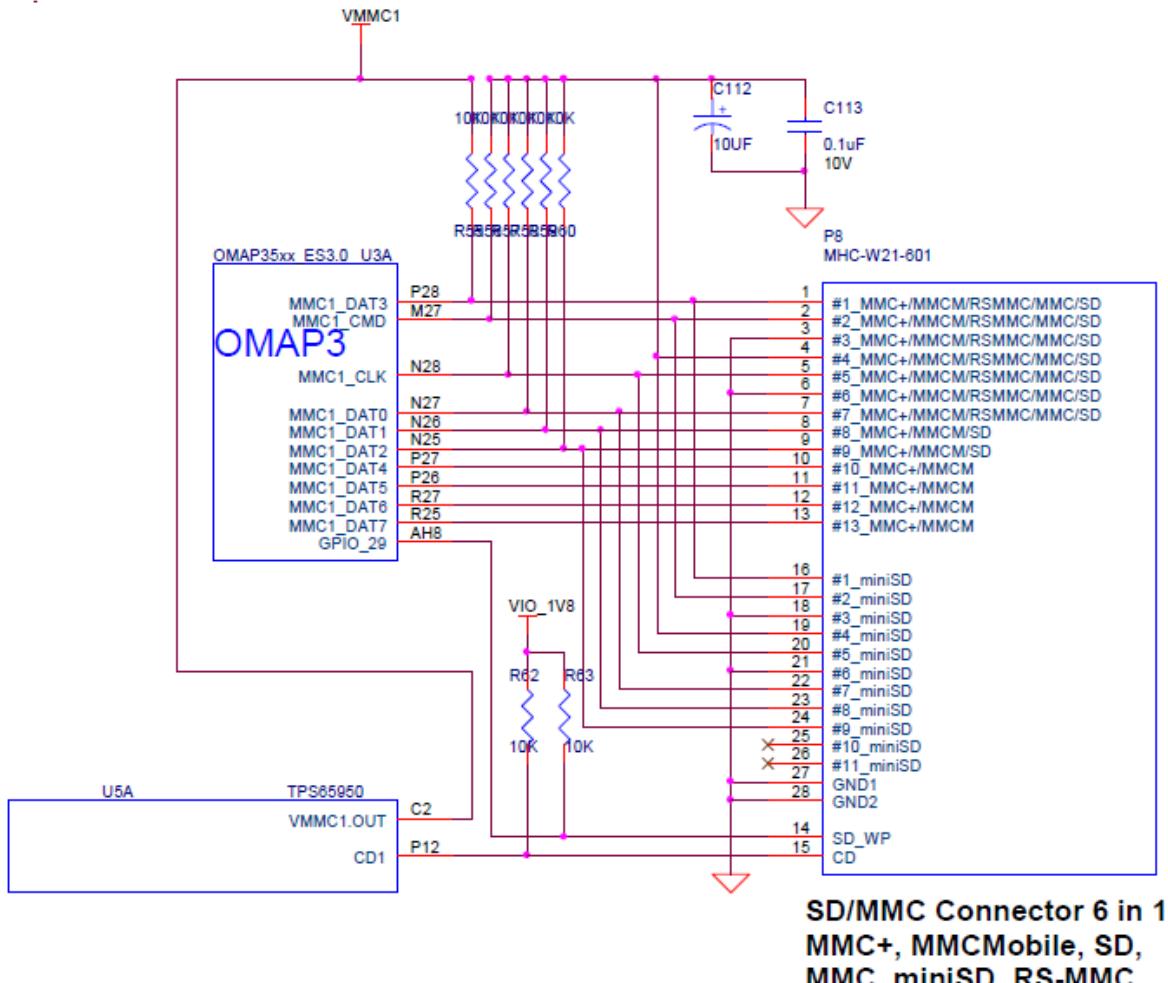


Figura 2.1 - Conexionado conector SD/MMC 6 en 1

2.2.6 Puerto USB-OTGⁱⁱⁱ

Este puerto es la fuente primaria de alimentación y transferencia de datos prevista para la placa BB, derivando la energía de la PC o equipo donde se conecte.

El Universal Serial Bus (bus universal en serie) o Conducto Universal en Serie (CUS), abreviado comúnmente USB, es un puerto que permite conectar periféricos a una computadora.

El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades plug-and-play,



permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos, o donde se necesita una latencia baja, los buses PCI o PCIe son la opción más adecuada.

La extensión USB-On-The-Go (USB-OTG) añadida al estándar USB, permite que un equipo pueda alternativamente actuar como servidor o como dispositivo. Esta nueva funcionalidad fue específicamente diseñada para equipos como la BB, cuyo rol en la comunicación USB varía continuamente dependiendo de la aplicación.

La Figura 2.2 muestra el conexionado del puerto USB-OTG.

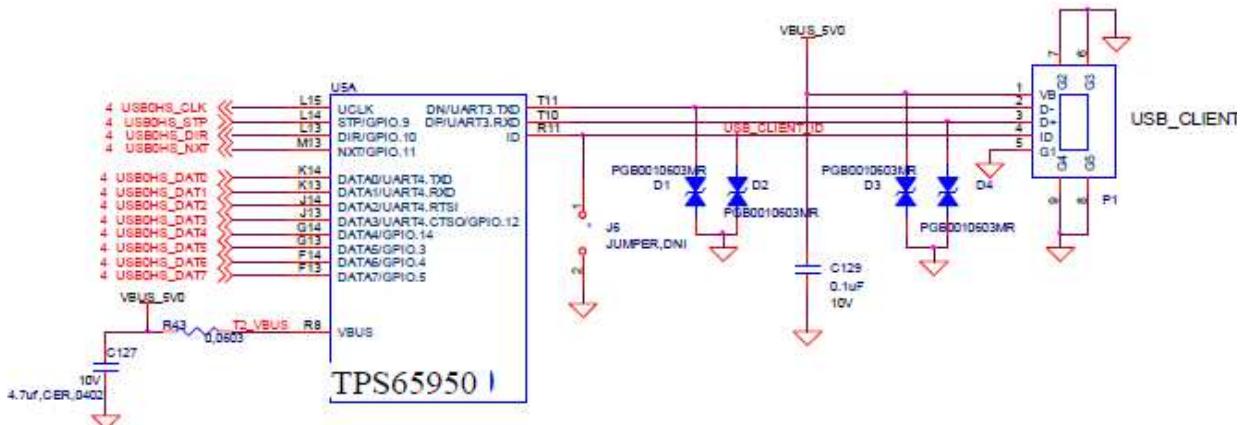


Figura 2.2 - Conexionado puerto USB-OTG

2.2.7 Puerto de Expansión

La placa BB posee un puerto de expansión diseñado para hacer aún más adaptable la aplicación de la placa. Dicho puerto está conectado con diversos pines de entrada del procesador pudiendo multiplexar varias señales.



La Figura 2.3 muestra el conexionado del puerto de expansión.

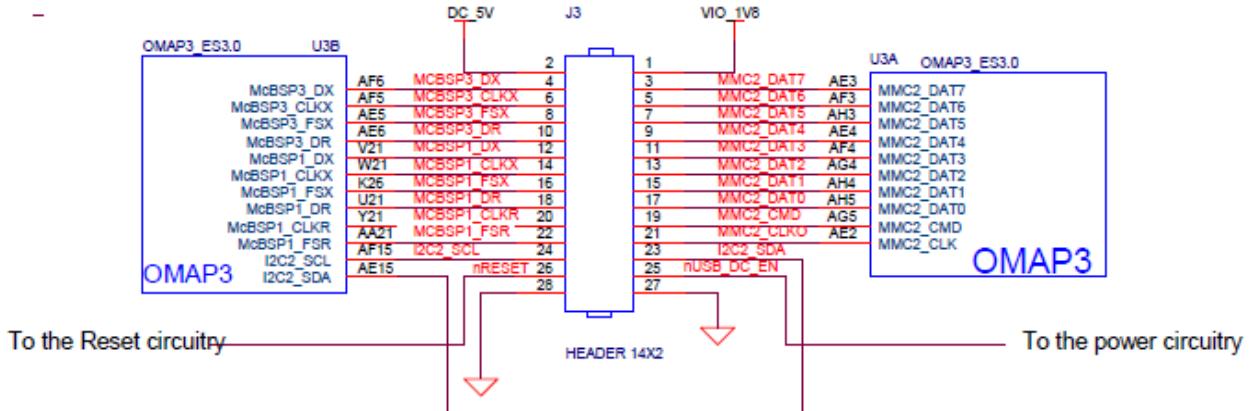


Figura 2.3 - Conexionado puerto de expansión

Existen varios protocolos soportados nativamente en dicho puerto, entre ellos los siguientes:

- General Port Input Output (GPIO)
- I2C
- SPI

Luego de investigar sobre estos protocolos, por la simpleza de su uso y la gran documentación existente, se decidió utilizar el protocolo GPIO.

2.2.7.1 Protocolo GPIO

General Purpose Input/Output es un protocolo que permite establecer una interfaz con periféricos y dispositivos externos. Esta interfaz puede actuar como una entrada (para leer señales digitales de otras partes del circuito), o como una salida, para controlar otros dispositivos.

Las interfaces GPIO suelen estar organizadas en grupos, típicamente de 8 pines (un puerto GPIO), que usualmente se pueden configurar como entradas o como salidas de forma independiente.

En algunos casos, las interfaces GPIO se pueden configurar para producir interrupciones a la CPU y para ser capaces de utilizar el acceso directo a memoria para mover grandes cantidades de datos de forma eficiente hacia el dispositivo o desde el dispositivo.



2.2.8 Puerto RS232

En la implementación de la BB se previó la incorporación de un puerto serial que permitiera la ejecución de comandos mediante una consola. Dicho puerto utiliza el estándar TIA/EIA RS 232 y está conectado directamente a la UART3 del OMAP3530 de la forma que se muestra en la Figura 2.4.

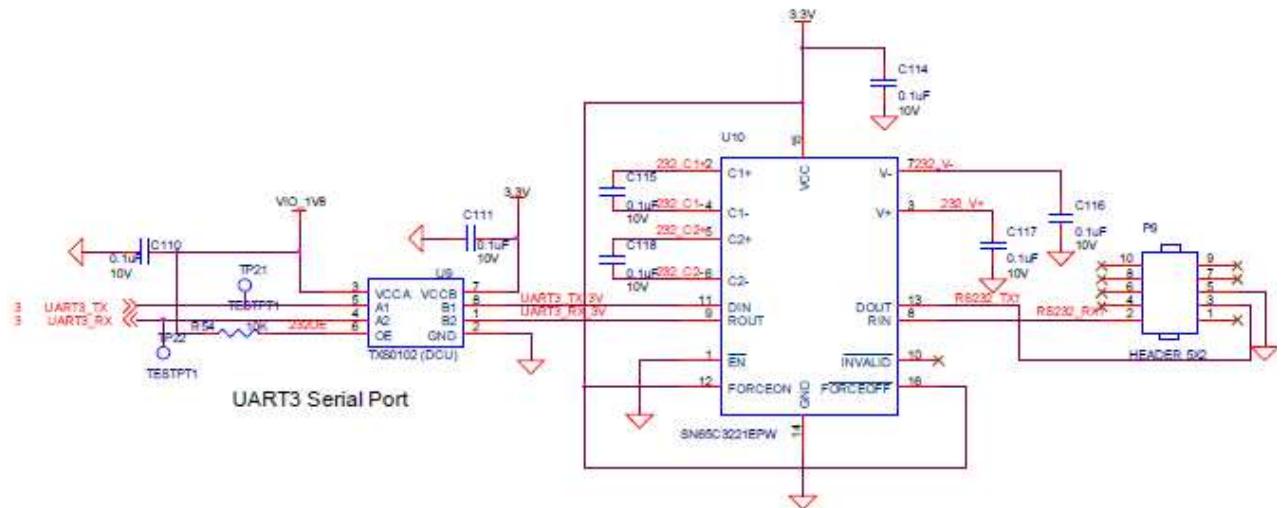


Figura 2.4 - Puerto Serial RS232

Este puerto se utiliza para ejecutar comandos que permitan realizar la configuración inicial del procesador para que arranque de la SD automáticamente, y para instalar luego un servidor SSH en el sistema operativo de forma de poder utilizar directamente un intérprete de comandos a través de una red TCP/IP.

2.2.9 Alimentación

La BB posee dos fuentes de alimentación disponibles. La primera es la correspondiente al puerto USB-OTG, en donde la placa toma como alimentación los 5 V que provee el PC ó el HUB-USB en donde se haya conectado. La segunda fuente posible es conectando directamente la BB a una fuente de alimentación independiente, como puede ser un transformador a red eléctrica, y obtener de ahí los 5 V necesarios.

En algunas aplicaciones que necesitan un gran consumo de corriente para su correcto funcionamiento, es posible utilizar un conector en con derivación en el puerto USB-OTG



para obtener más potencia ó utilizar un pin del puerto de expansión que permite ingresar otros 5 V adicionales de energía.

La Figura 2.5 muestra el conexionado de la alimentación de la BB.

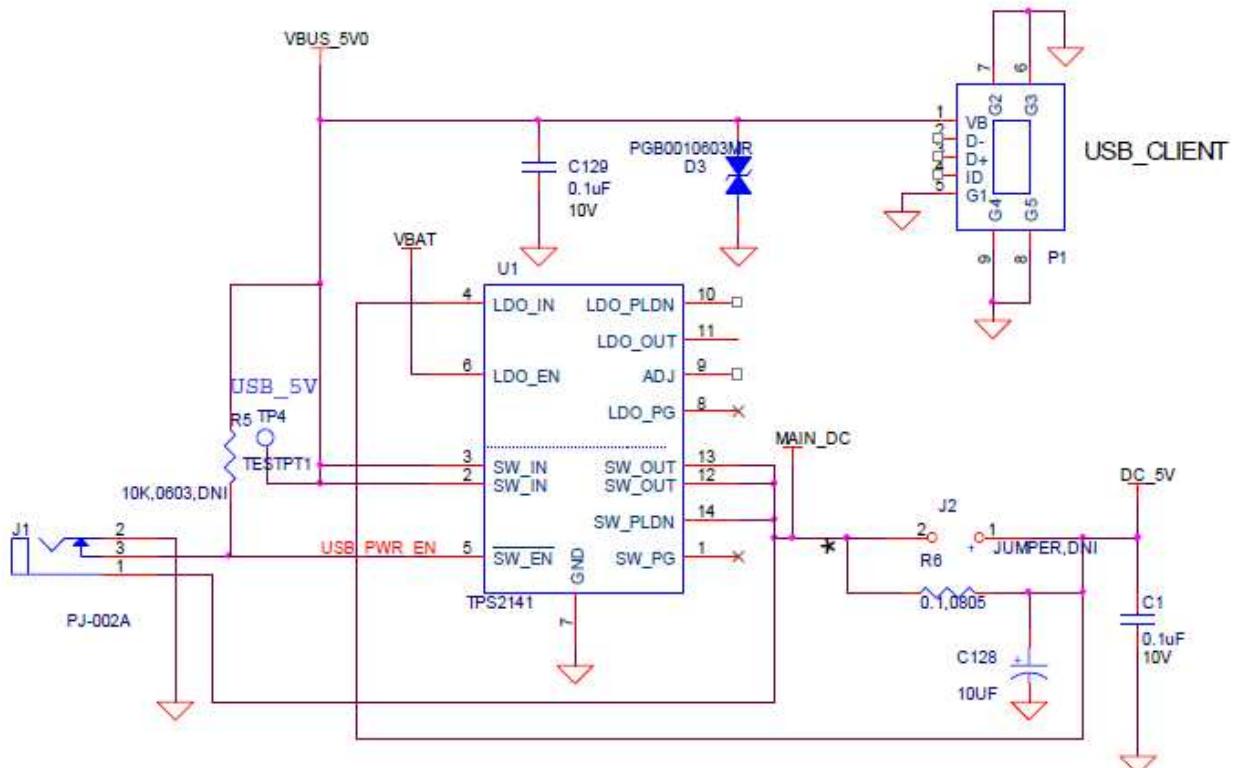


Figura 2.5 - Conexión alimentación BB



2.3 Sistema operativo

Se decidió utilizar la distribución Ubuntu del kernel de Linux para implementar UMix. Los factores que influyeron en esta decisión se explican en el capítulo Desarrollo del proyecto de este documento. A continuación se hace una reseña sobre el sistema operativo Linux y sobre el kernel de Ubuntu.

2.3.1 Linux

A pesar de que el término “Linux” refiere únicamente al kernel implementado por Linus Torvalds, por lo general se utiliza este término para referirse a todos los sistemas operativos que hacen uso de dicho kernel.

Originalmente diseñado como proyecto para la Universidad de Helsinki y basado en el kernel Minix, propuesto por Andrew Tannenbaum, Linux ganó gran popularidad tras haber sido adoptado por el proyecto GNU como reemplazo de Hurd.

En cuanto a su arquitectura, Linux es un kernel de tipo monolítico. Esto significa que todo el sistema operativo se ejecuta dentro del espacio de memoria asignado al kernel y en modalidad de supervisor. A diferencia de otros tipos de kernel, los núcleos monolíticos definen una interfaz de alto nivel sobre el hardware del sistema, que permite la manipulación del mismo a través de llamadas al sistema.



La Figura 2.6 ilustra las diferencias entre los tipos más comunes de núcleos.

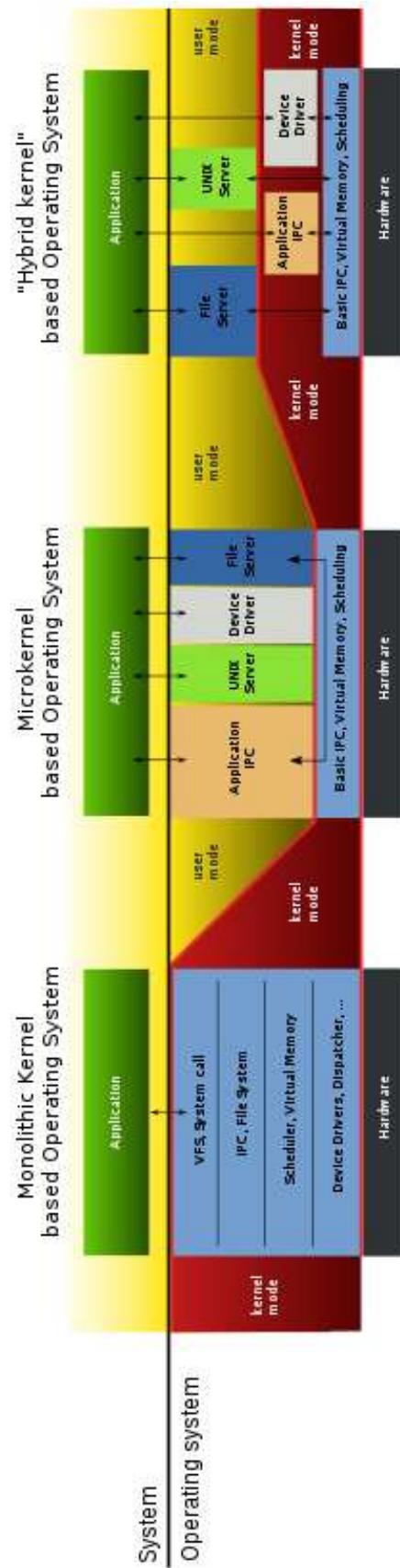


Figura 2.6 - Diferencias entre núcleos



La Figura 2.7 ilustra el mapa de memoria de Linux.

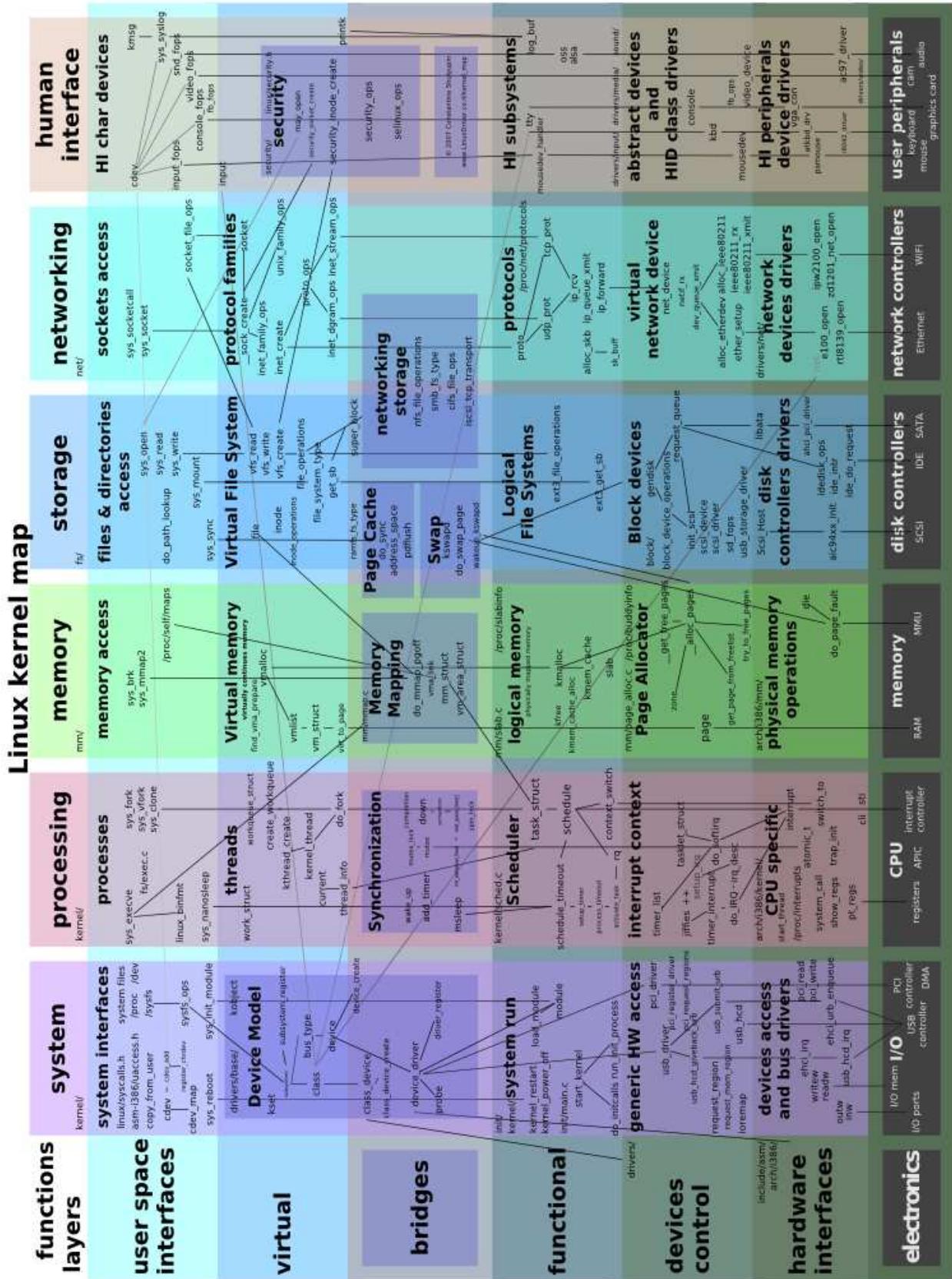


Figura 2.7 - Mapa de memoria de Linux



Una de las grandes ventajas de Linux con respecto a otros núcleos, es que su implementación se hizo en C, el cual es un lenguaje de programación de alto nivel. Esto hace, en primer lugar, que el código sea fácilmente interpretable por cualquier programador y, en segundo lugar, que sea transportable hacia casi cualquier sistema sin mayores dificultades.

2.3.2 Kernel de Ubuntu

Si bien el kernel de Linux está estandarizado y mantenido al día de hoy por Linus Torvalds, muchas comunidades independientes decidieron basarse en dicho kernel e incorporarle algunos cambios y aplicaciones, de forma de obtener sistemas operativos completos y funcionales para una gran variedad de sistemas.

Cada sistema operativo liberado por dichas comunidades se conoce como “distribución de Linux”, como son, por ejemplo: Ubuntu, Suse, Debian, RedHat, Mandriva, Slackware, etc. Debido a las libertades que ofrece la licencia GPL, bajo la cual esta licenciado Linux, cualquier persona puede registrar su propia distribución de Linux, sin necesidad de realizar ningún desembolso económico.

En particular una distribución que se ha hecho muy popular en los últimos tiempos es Ubuntu. Esta distribución tiene la ventaja de basarse en otra distribución denominada Debian, reconocida por su estabilidad, por tener una gran comunidad de soporte y desarrollo y por lanzar actualizaciones rápidamente.



2.4 Comunicación

El objetivo del proyecto UMix requiere que el móvil pueda comunicarse fácilmente con diversos dispositivos que se encuentran a gran distancia. Es por este motivo, que a la hora de su diseño se optó por incorporar al sistema la posibilidad de manejar conexiones de red estandarizadas, que permitan el pasaje de paquetes TCP/IP.

En concreto, se puso énfasis en desarrollar compatibilidad con los módems que ofrecen conexiones inalámbricas 3G, ya que esta tecnología está ampliamente difundida y tiene la ventaja de que, disponiendo de una radio base móvil, en relativamente poco tiempo se podría disponer de cobertura 3G en el área de la catástrofe. Esta solución resulta más eficiente que depender de la instalación de una red inalámbrica en el área de interés.

2.4.1 Enlace 3G

El nombre 3G ó IMT-2000 se refiere a una familia de estándares para comunicaciones móviles definidos por la ITU (International Telecommunication Unit). Los servicios provistos por estos estándares incluyen telefonía, video-llamadas y transferencia de datos dentro de entornos móviles.

Las grandes ventajas que ofrece esta tecnología con respecto a sus predecesoras son las siguientes: mayor velocidad de transferencia de datos, mejor aprovechamiento espectral y nuevas funcionalidades orientadas a la confidencialidad y seguridad de las comunicaciones.

Dentro del proyecto se incluyó un módem 3G conectado a un HUB-USB que, a su vez, se conectó al puerto USB-OTG de la BB mediante su puerto maestro. El modelo de dicho módem se escogió de forma que fuera compatible con los drivers incorporados en el kernel de Ubuntu utilizado. De esta forma, y luego de configurar adecuadamente el software de conexión, se dispuso de una conexión inalámbrica mediante la cual acceder a las distintas interfaces de usuario y al servidor SSH que provee UMix.



2.4.2 Enlace Ethernet

Ethernet ó IEEE 802.3 es un estándar para redes de área local que utiliza el protocolo de acceso al medio CSMA/CD. Dentro de dicho estándar se definen los métodos de cableado, la señalización eléctrica y la forma de empaquetar la información.

Este medio de comunicación fue incorporado con el objetivo de facilitar la transferencia de información entre la BB y un PC durante el transcurso de la implementación del sistema. De cualquier forma, esta interface se dejará habilitada de modo que se pueda acceder al móvil prácticamente desde cualquier PC.



2.5 Microcontrolador PIC^{iv}

Para el desarrollo de la lógica de control de motores y sensores se utilizó un micro controlador PIC 16F877A. Este micro controlador es un dispositivo de alta performance con (Unidad Central de Procesamiento) CPU RISC (Reduced Instruction Set Computer), la cual consta de 35 instrucciones de una palabra que se ejecutan en un único ciclo de instrucción.

El PIC 16F877A cuenta con varios periféricos. Los periféricos son lo que diferencia un microcontrolador de un microprocesador. Estos facilitan la interfaz con el exterior del PIC y tareas internas tales como mantener distintas bases de tiempo. Algunos de los periféricos incluidos en este PIC son: timers, módulos de captura, comparación y modulación por ancho de pulso (PWM), puerto serial síncrono (SSP), receptor-transmisor síncrono asíncrono universal (USART), puerto paralelo esclavo (PSP), conversor analógico/digital (A/D), entre otros.

De todos los periféricos disponibles, en este proyecto solo se hizo uso del Timer2 y de los puertos de entrada/salida. Por este motivo, en la sección *Periféricos* de este informe solo se explicarán estos módulos.



En la Figura 2.8 se muestra el diagrama de bloques del PIC utilizado.

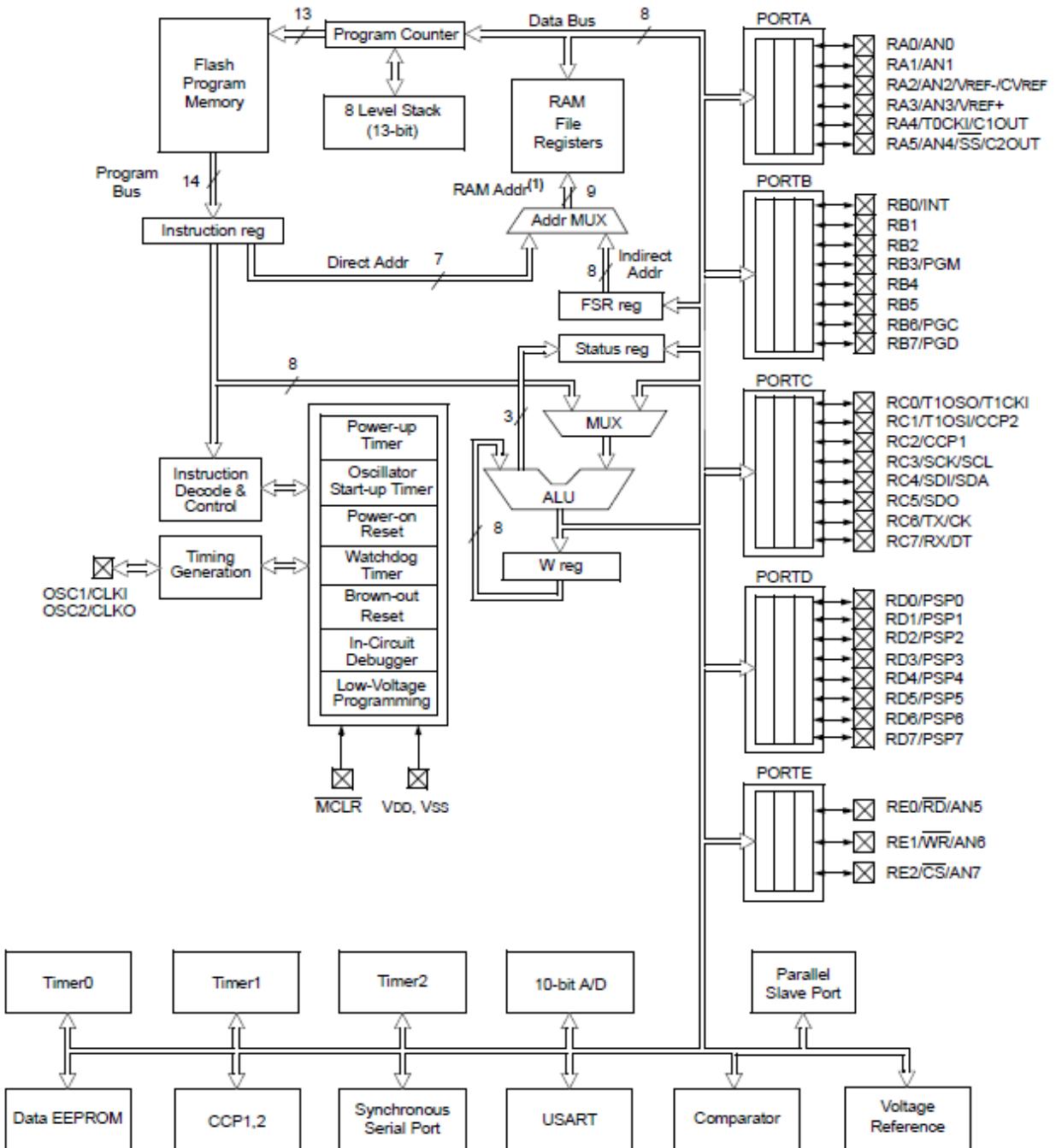


Figura 2.8 - Diagrama de bloques PIC 16F877A



2.5.1 Arquitectura

El PIC 16F877A tiene una arquitectura Harvard, por lo que la memoria de programa y la memoria de datos están separadas y éstas son accedidas desde buses separados. Esto mejora el ancho de banda y permite que se ejecute una instrucción mientras se carga la siguiente (pipeline de instrucciones).

Otras características de la arquitectura de este PIC son las siguientes:

- Dispone de instrucciones de palabra larga e instrucciones de palabra sencilla
- Ejecuta las instrucciones en un solo ciclo de máquina
- Tiene un set de instrucciones reducido lo cual permite que sea aprendido fácilmente
- La memoria de registro de archivos o datos puede ser accedida directa o indirectamente

Su programación es sencilla pero eficiente, ya que las instrucciones son simétricas, es decir que es posible realizar cualquier operación sobre cualquier registro usando cualquier modo de direccionamiento. Solo se utilizan dos instrucciones no orientadas a registros que son SLEEP (coloca al dispositivo en el modo de menor consumo) y CLRWDT (verifica que el chip esté operando correctamente evitando que el Watchdog Timer (WDT) se desborde y resetee el dispositivo).

2.5.2 CPU y ALU

La Unidad Central de Procesamiento (CPU) es responsable de usar la información que está en la memoria de programa (instrucciones) para controlar la operación del dispositivo. Muchas de estas instrucciones operan sobre la memoria de datos. Para operar sobre la memoria de datos es necesaria la Unidad Lógica Aritmética (ALU).

Además de realizar operaciones aritméticas y lógicas, la ALU controla los bits de estado (que se encuentran en el registro STATUS). El resultado de algunas instrucciones fuerzan los bits de estado a un valor que depende del estado del resultado.



2.5.3 Oscilador

El circuito interno del oscilador se usa para generar el reloj del PIC, el cual es necesario para que dicho dispositivo ejecute las instrucciones y para que funcionen los periféricos. Cuatro períodos del reloj del dispositivo generan un ciclo del reloj interno de instrucciones (T_{CY}).

Existen ocho modos en los que puede trabajar el oscilador. Para seleccionar el modo del oscilador se utilizan los bits de configuración del dispositivo FOSC2, FOSC1 y FOSC0. Los diferentes modos permitirán ahorrar costos del sistema o ahorrar energía.

En este proyecto se utiliza el oscilador en modo High Speed para lo cual se conecta un cristal de 20 MHz en los pines OSC1 y OSC2. Se tomó la decisión de utilizar este modo debido a que se requería alta velocidad en la lógica de control de motores y sensores.

2.5.4 Memoria

El PIC 16F877A tiene tres bloques de memoria: memoria de programa, memoria de datos y memoria de datos EEPROM.

2.5.4.1 Memoria de programa

La memoria de programa es una memoria Flash que puede ser borrada eléctricamente, es decir que puede ser borrada y reprogramada sin ser removida del circuito.

Para especificar la dirección de la instrucción a cargar para ser ejecutada se utiliza un contador de programa (PC) de 13 bits, es decir que es capaz de direccionar un espacio de memoria de programa de una palabra de 8K por 14 bits. El ancho del bus de la memoria de programa es 14 bits. Debido a que todas las instrucciones son de una palabra, un dispositivo con una memoria de programa de 8K por 14 bits tiene espacio para 8K de instrucciones.



En la posición 0000h se encuentra el vector de Reset y en la posición 0004h se encuentra el vector de interrupción. Es decir que al resetearse el PIC el PC apuntará a la dirección 0000h y al ocurrir una interrupción el PC apuntará a la dirección 0004h.

Se dispone de un stack de 8 niveles de profundidad por 13 bits de ancho, el cual permite que ocurra una combinación de hasta 8 llamadas de programa e interrupciones. El stack contiene las direcciones de retorno.

En la Figura 2.9 se muestra el mapa de la memoria de programa y el stack.

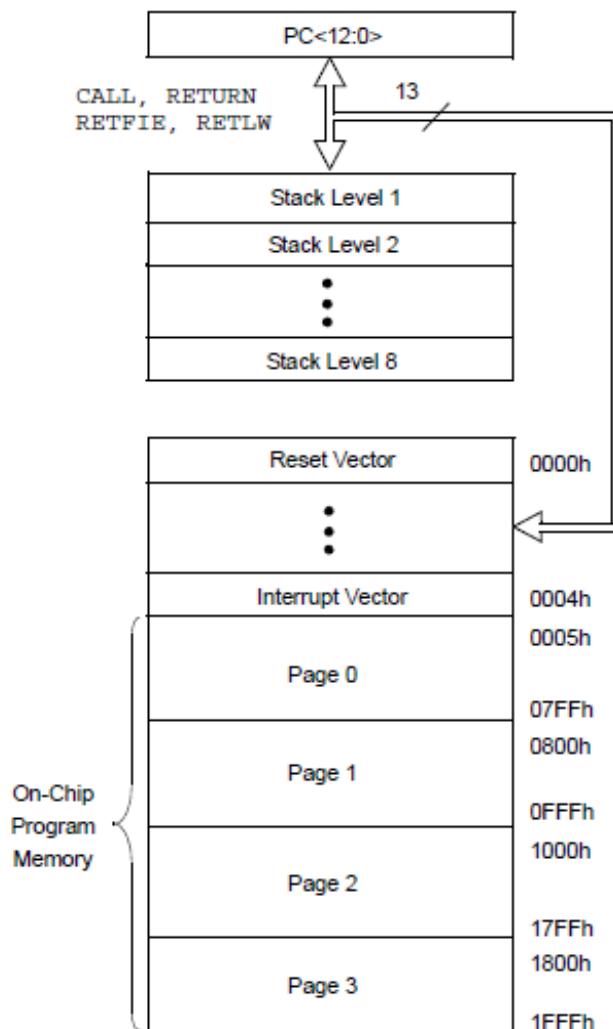


Figura 2.9 - Mapa de la memoria de programa y stack



2.5.4.2 Memoria de datos

La memoria de datos está particionada en cuatro bancos los cuales contienen los registros de propósito general (GPRs) y los registros de función especial (SFRs). Los registros de propósito general son el área general para almacenamiento de datos, mientras que los registros de función especial son registros utilizados por la CPU y por los módulos periféricos para controlar la operación deseada del dispositivo.

Toda la memoria de datos puede ser accedida tanto directa como indirectamente. El direccionamiento directo requiere el uso de los bits RP1 (bit 6 del registro STATUS) y RP0 (bit 5 del registro STATUS). El direccionamiento indirecto requiere el uso del registro de selección de archivo (FSR). El direccionamiento indirecto utiliza el bit puntero de registro indirecto (IRP) del registro STATUS para los accesos a las áreas Banco0/Banco1 o Banco2/Banco3 de la memoria de datos.

Cada banco tiene 128 bytes. Las ubicaciones más bajas de cada banco están reservadas para los registros de función especial. Luego están los registros de propósito general, implementados como RAM estática.



En la Figura 2.10 se muestra el mapa de los bancos.

File Address	File Address	File Address	File Address
Indirect addr. ⁽¹⁾	00h	Indirect addr. ⁽¹⁾	100h
TMR0	01h	OPTION_REG	101h
PCL	02h	PCL	102h
STATUS	03h	STATUS	103h
FSR	04h	FSR	104h
PORTA	05h	TRISA	105h
PORTB	06h	TRISB	106h
PORTC	07h	TRISC	107h
PORTD ⁽¹⁾	08h	TRISD ⁽¹⁾	108h
PORTE ⁽¹⁾	09h	TRISE ⁽¹⁾	109h
PCLATH	0Ah	PCLATH	10Ah
INTCON	0Bh	INTCON	10Bh
PIR1	0Ch	PIE1	10Ch
PIR2	0Dh	PIE2	10Dh
TMR1L	0Eh	PCON	10Eh
TMR1H	0Fh		10Fh
T1CON	10h		110h
TMR2	11h	SSPCON2	111h
T2CON	12h	PR2	112h
SSPBUF	13h	SSPADD	113h
SSPCON	14h	SSPSTAT	114h
CCPR1L	15h		115h
CCPR1H	16h		116h
CCP1CON	17h		117h
RCSTA	18h	TXSTA	118h
TXREG	19h	SPBRG	119h
RCREG	1Ah		11Ah
CCPR2L	1Bh		11Bh
CCPR2H	1Ch	CMCON	11Ch
CCP2CON	1Dh	CVRCON	11Dh
ADRESH	1Eh	ADRESL	11Eh
ADCOND	1Fh	ADCON1	11Fh
General Purpose Register	20h	A0h	120h
96 Bytes			
Bank 0	7Fh	General Purpose Register	General Purpose Register
		80 Bytes	80 Bytes
		accesses 70h-7Fh	accesses 70h-7Fh
		EFh	16Fh
		F0h	170h
		FFh	17Fh
			General Purpose Register
			80 Bytes
			accesses 70h - 7Fh
			1EFh
			1F0h
			1FFh
			Bank 3

- Unimplemented data memory locations, read as '0'.
* Not a physical register.

- Note 1: These registers are not implemented on the PIC16F876A.
2: These registers are reserved; maintain these registers clear.

Figura 2.10 - Mapa de bancos de memoria



2.5.4.2.1 Registro STATUS

Este registro contiene el estado aritmético de la ALU, el estado de RESET y los bits de selección de banco para la memoria de datos. Debido a que la selección del banco de la memoria de datos es controlada por este registro, es necesario que esté presente en cada banco.

2.5.4.2.2 Registro OPTION_REG

El registro OPTION_REG es un registro que se puede leer y escribir y que contiene varios bits de control para configurar el pre-scaler del Timer0/WDT, la interrupción externa INT, el Timer0 y las pull-ups del puerto B.

2.5.4.2.3 Registro PCON

El registro Control de Poder (PCON) contiene flags, los cuales, junto con los bits TO y PD, permiten que el usuario pueda diferenciar entre los resets del dispositivo.

2.5.4.3 Memoria de datos EEPROM

La memoria de datos EEPROM se puede leer y escribir durante operación normal. Esta memoria no se mapea directamente en el espacio del archivo de registro. En cambio, es direccionada indirectamente a través de los SFRs.

Se utilizan cuatro SFRs para leer y escribir esta memoria: **EECON1** (contiene los bits de control), **EECON2** (es el registro utilizado para iniciar la lectura/escritura), **EEDATA** (mantiene el dato de 8 bits a ser leído/escrito) y **EEADR** (mantiene la dirección de la ubicación de la EEPROM que está siendo accedida).

2.5.5 Modo SLEEP

El modo SLEEP es un modo en el que el dispositivo es puesto en su estado de menor consumo de corriente. Al entrar en modo SLEEP se apaga el oscilador del dispositivo, por



lo que no avanza el reloj del sistema. Se ingresa al modo SLEEP ejecutando la instrucción **sleep**.

Si está habilitado, el WDT es puesto en “0” pero sigue corriendo, el bit PD del registro **STATUS** también es puesto en “0”, el bit TO es seteado y el driver del oscilador es apagado. Los puertos de entrada/salida mantienen el estado que tenían antes de que se ejecutara la instrucción **sleep**.

Para menor consumo de corriente en este estado, todos los pines de entrada/salida deberían estar a V_{DD} o a V_{SS} , sin ninguna circuitería externa drenando corriente del pin de entrada/salida, y los módulos que se especifica que tienen una corriente diferencial en modo sleep deberían ser deshabilitados.

El dispositivo se puede despertar del modo SLEEP a través de uno de los siguientes eventos:

- Cualquier reset del dispositivo
- WDT (si está habilitado)
- Cualquier módulo periférico que pueda setear su flag de interrupción mientras el dispositivo se encuentra en modo SLEEP, tales como:
 - Pin de interrupción externa (RB0)
 - Pines de cambio en el puerto
 - Comparadores
 - A/D
 - Timer1
 - LCD
 - SSP
 - Captura



El primer evento (reset) va a resetear el dispositivo al despertarse. Sin embargo, los otros dos (WDT y periféricos) van a despertar al dispositivo y luego continuar con la ejecución del programa. Se pueden usar los bits TO y PD del registro **STATUS** para determinar la causa del reset del dispositivo. El bit PD, el cual es seteado al encenderse el dispositivo, se pone en “0” cuando se invoca la instrucción SLEEP. El bit TO se pone en “0” si el dispositivo se despertó por el WDT.

Cuando se está ejecutando la instrucción SLEEP, la siguiente instrucción es pre-cargada. Para que el dispositivo se despierte a través de un evento de interrupción, el correspondiente bit de habilitación de interrupción debe ser seteado. El dispositivo se va a despertar sin importar el estado del bit GIE. Si el bit GIE está en “0”, el dispositivo continuará la ejecución en la instrucción siguiente a la instrucción SLEEP. Si el bit GIE está en “1”, el dispositivo ejecuta la instrucción siguiente a la instrucción SLEEP y luego va al vector de interrupción (0004h). En los casos en que la ejecución de la instrucción siguiente a la instrucción SLEEP no es deseada, el usuario debería colocar un NOP (no operation) luego de la instrucción SLEEP.

2.5.6 Interrupciones

Existen quince fuentes distintas que pueden provocar una interrupción. Como mínimo se utiliza un registro para el control y el estado de las interrupciones. Éste es el registro **INTCON**, el cual contiene flags de interrupciones, bits para habilitar interrupciones y el bit para habilitar la interrupción global (GIE). Este último bit, si está seteado, habilita todas las interrupciones no enmascaradas o, si está en cero, deshabilita todas las interrupciones.

Por otro lado, existen registros para habilitar las interrupciones de los periféricos y registros para mantener los flags de interrupción. Estos registros son: **PIE1**, **PIR1**, **PIE2** y **PIR2**.

La instrucción “regresar de la interrupción”, RETFIE, hace que la ejecución salga de la rutina de interrupción y setea el bit GIE, lo cual permite que se ejecute cualquier interrupción pendiente.



Cuando ocurre una interrupción, se pone en 0 el bit GIE para deshabilitar cualquier otra interrupción, la dirección de retorno se inserta en el stack y se carga el PC con 0004h. Una vez en la rutina de atención de interrupción, se puede determinar la fuente de la interrupción testeando los flags de interrupción. Por lo general, estos flags deben ser puestos en 0 por software antes de volver a habilitar la interrupción global para evitar interrupciones recursivas.

En este proyecto se utilizan las fuentes de interrupción: Interrupción del Pin INT (Interrupción externa) e Interrupción por cambio en el Puerto B (pines RB7:RB4) por lo que solo estas fuentes de interrupción serán descritas con mayor detalle.

2.5.6.1 Interrupción externa

La interrupción externa en el pin INT es disparada por un flanco, ya sea el flanco ascendente si está seteado el bit INTEDG (bit 6 del registro OPTION_REG), o el flanco descendente si el bit INTEDG está en 0.

Cuando aparece un flanco válido en el pin INT, se setea el flag INTF (bit 1 del registro INTCON). Esta interrupción se puede habilitar/deshabilitar poniendo en 1 o en 0 el bit INTE (bit 4 del registro INTCON). El bit INTF debe ponerse en 0 por software en la rutina de atención de interrupción antes de volver a habilitar esta interrupción.

La interrupción INT puede despertar al procesador del modo SLEEP, si el bit INTE fue seteado antes de entrar en modo SLEEP. El estado del bit GIE decide si el procesador va o no al vector de interrupción al despertarse.

2.5.6.2 Interrupción por cambio en el Puerto B

Cuatro pines del Puerto B, los pines RB7:RB4, tienen una función de interrupción por cambio. Solo los pines configurados como entradas pueden provocar que ocurra esta interrupción.



Los pines de entrada de RB7:RB4 son comparados con el valor que se almacenó la última vez que se leyó el Puerto B. Si alguno de estos pines no coincide con su valor anterior, se pondrá en “1” el flag RBIF (bit 0 del registro INTCON) lo cual generará una interrupción por cambio del Puerto B.

Esta interrupción puede despertar al dispositivo del modo SLEEP. Para limpiar la interrupción, el usuario puede leer o escribir en el Puerto B o bien poner en “0” el flag RBIF.

En la Figura 2.11 se muestra un diagrama de bloques de los pines RB7:RB4.

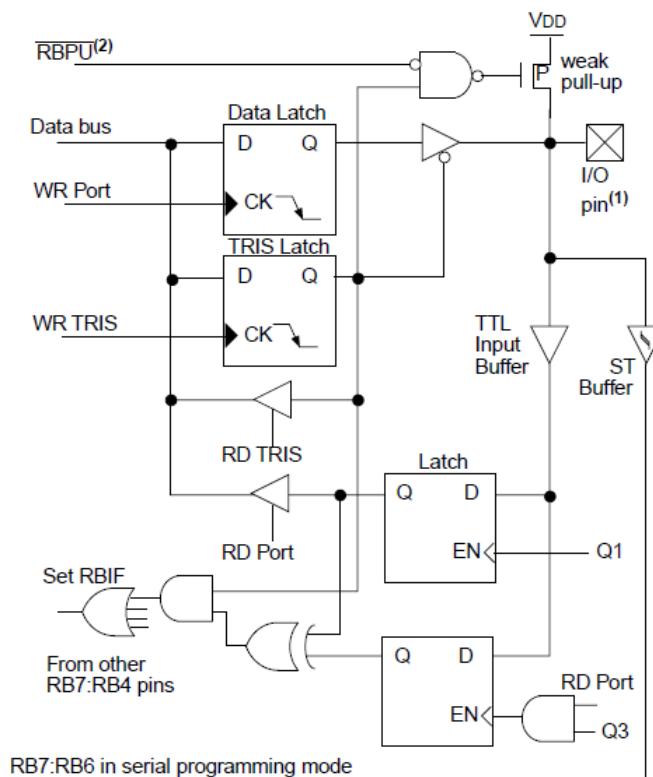


Figura 2.11 - Diagrama de bloques pines RB7:RB4



2.5.7 Periféricos

2.5.7.1 Puertos de entrada/salida

Los pines de entrada/salida de propósito general pueden considerarse los periféricos más simples. Permiten al PIC monitorear y controlar otros dispositivos. Para agregar flexibilidad y funcionalidad al PIC, algunos pines están multiplexados con una o varias funciones alternativas.

Para la mayoría de los puertos, la dirección de los pines de entrada/salida es controlada por el registro de dirección de datos, llamado registro TRIS. El bit x del registro TRIS controla la dirección del bit x del puerto correspondiente. Un “1” en el bit TRIS hace que el bit correspondiente sea una entrada, mientras que un “0” hace que el bit correspondiente sea una salida.

La Figura 2.12 muestra un esquema de un típico puerto de entrada/salida.

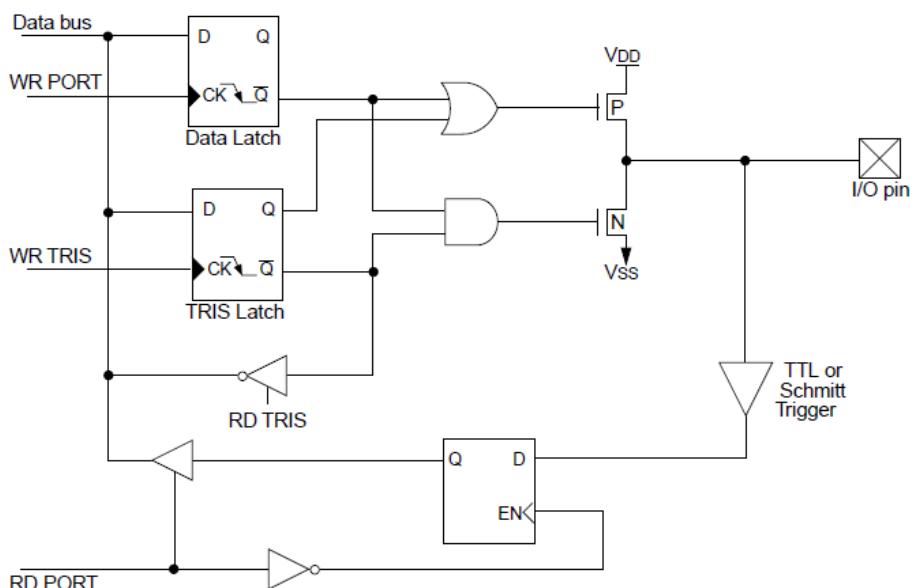


Figura 2.12 - Típico puerto de entrada/salida



2.5.7.2 Timer2

El PIC 16F877A dispone de tres timers: el Timer0, el Timer1 y el Timer2. Los tres timers tienen distintas características. En este proyecto se decidió trabajar con el Timer2 debido a que es el único de los tres que tiene pre-scaler y post-scaler. Los otros timers solo tienen pre-scaler.

El Timer2 es un timer de 8 bits. Sin embargo, si se utilizan el pre-scaler y el post-scaler en su máxima configuración, el tiempo de desborde es el mismo que el de un timer de 16 bits.

En la Figura 2.13 se muestra un diagrama de bloques del Timer2.

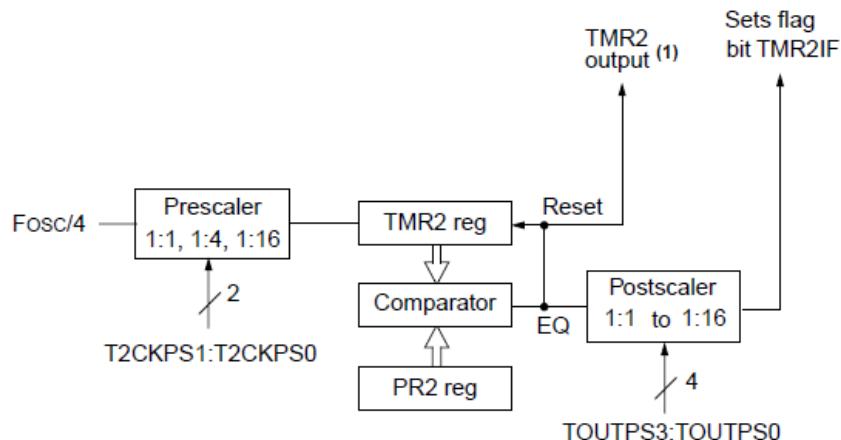


Figura 2.13 - Diagrama de bloques Timer2

El post-scaler cuenta cierta cantidad de veces que el registro **TMR2** haya coincidido con el registro **PR2**.

El registro de control del Timer2 es el registro T2CON. En la Figura 2.14 se muestra la estructura de este registro y los distintos valores que deben tomar sus bits para configurar el post-scaler, para configurar el pre-scaler y para encender el timer.



U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

bit 7
bit 6:3
bit 2
bit 1:0

- bit 7 **Unimplemented:** Read as '0'
- bit 6:3 **TOUTPS3:TOUTPS0:** Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale
- bit 2 **TMR2ON:** Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off
- bit 1:0 **T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend	
R = Readable bit	W = Writable bit
U = Unimplemented bit, read as '0'	- n = Value at POR reset

Figura 2.14 - Registro T2CON

El módulo Timer2 utiliza como reloj de entrada el reloj del dispositivo ($F_{osc}/4$). Con los bits T2CKPS1:T2CKPS0 se puede seleccionar la pre-escala que se le aplicará a este reloj.

El registro **TMR2** se puede leer y escribir, y se pone en "0" en todos los resets del dispositivo. El Timer2 se incrementa desde 00h hasta que coincide con **PR2** y luego se resetea a 00h en el siguiente ciclo de incremento. **PR2** es un registro que también se puede leer y escribir.

Hay cuatro bits que seleccionan el valor del post-scaler entre 1:1 y 1:16. Después de que el post-scaler se desborda, el flag de interrupción TMR2 (TMR2IF) se setea para indicar un desborde del Timer2.



2.6 Herramientas de desarrollo

Para el desarrollo de la aplicación a nivel del PIC se utilizaron un conjunto de herramientas, tales como el entorno de desarrollo MPLAB, el paquete MPASM, el simulador Proteus y el programador PICSTART Plus.

2.6.1 Entorno de desarrollo integrado MPLAB^v

Es el entorno en el cual opera todo el conjunto de herramientas de desarrollo. Esto permite que este conjunto de herramientas sea consistente, por lo que solo es necesario un mínimo conocimiento de la nueva herramienta.

Este entorno de desarrollo integra los siguientes aspectos del desarrollo:

- Edición del código fuente
- Gestión del proyecto
- Generación de código de máquina (a partir de Assembler o “C”)
- Simulación y emulación del dispositivo
- Programación del dispositivo

Este conjunto de herramientas permite el desarrollo completo de un proyecto sin salir del entorno MPLAB. El software MPLAB, incluido en el entorno de desarrollo, ofrece la posibilidad de realizar distintas tareas, tales como editar archivos fuente, ya sea en lenguaje assembler o en “C”; compilar de forma sencilla; depurar la aplicación; ejecutar emuladores, entre otras.

2.6.2 Paquete MPASM^{vi}

Para que el dispositivo opere como se desea en la aplicación específica, es necesario escribir un programa para el microcontrolador. En el caso de este proyecto se escribió un programa en lenguaje assembler, por lo que se generó un archivo .asm. Para grabar este programa en el PIC, primero es necesario ensamblarlo, es decir, generar un archivo de extensión .hex.



El paquete MPASM incluye el **MPASM Assembler**, el **MPLINK Object Linker** y el **MPLIB Librarian**.

Es posible producir el archivo *.hex* a partir de un único archivo *.asm* con código simple, o bien es posible generar un archivo *.hex* a partir de la unión de distintos códigos objeto y otros módulos ensamblados y/o compilados. Para unir estos módulos se utiliza la herramienta MPLINK. También es posible utilizar la herramienta MPLIB para generar bibliotecas, las cuales son una colección de códigos objeto listos para ser utilizados y que se almacenan todos juntos en un único archivo con extensión *.lib*.

2.6.3 Simulador Proteus

El simulador Proteus es un entorno gráfico de diseño electrónico sencillo e intuitivo, el cual permite simular un proyecto antes de probarlo en la práctica. El hecho de poder realizar esta simulación permite ahorrar tiempo y dinero al poder verificar el correcto funcionamiento del proyecto antes de armarlo en el laboratorio. Sin embargo, vale la pena aclarar que en la situación real pueden surgir inconvenientes que no surgieron en la simulación, principalmente debido a la presencia de una fuente espuria de ruido eléctrico.

La ventaja de la simulación es que permite depurar los circuitos que se desee armar y hacer modificaciones a medida que se vayan detectando errores o funcionamientos no esperados.

2.6.4 Programador PICSTART Plus

El programador PICSTART Plus es un programador de prototipos de bajo costo y fácil de utilizar, el cual se conecta a la PC a través del puerto serial. A través de este dispositivo es posible grabar el programa ensamblado en el microcontrolador.



2.7 Control de motores^{vii,viii}

El móvil construido utiliza dos motores para su desplazamiento: un motor de continua para la tracción (ruedas traseras) y un motor paso a paso (PaP) para la dirección (ruedas delanteras).

2.7.1 Motor de continua (Tracción)

En el artículo de Wikipedia sobre motores de corriente continua dice:

El **motor de corriente continua** es una máquina que convierte la energía eléctrica en mecánica, principalmente mediante el movimiento rotatorio. (...) Una máquina de corriente continua (generador o motor) se compone principalmente de dos partes, un estator que da soporte mecánico al aparato y tiene un hueco en el centro generalmente de forma cilíndrica. En el estator además se encuentran los polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre un núcleo de hierro. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas.¹

Para controlar el sentido de giro del motor de continua es necesario poder invertir la polaridad del voltaje aplicado a los terminales del motor. Existen varias formas de lograr esto: una es utilizando una fuente simétrica o dos fuentes de alimentación con un interruptor simple de dos contactos y otra es utilizar una fuente común con un interruptor doble.

¹ es.wikipedia.org. *Motor de corriente continua* [en línea] <http://es.wikipedia.org/wiki/Motor_de_corriente_continua> [citado en 4 de diciembre de 2009].



Para implementar la primera opción basta con utilizar dos transistores complementarios como muestra el siguiente esquema:

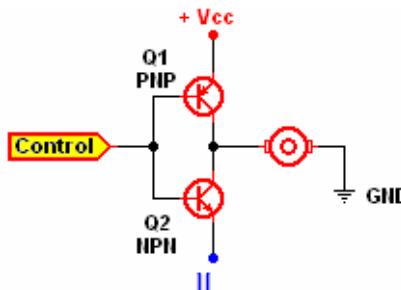


Figura 2.15 - Motor de continua con fuente simétrica

Para implementar la segunda opción es necesario implementar un puente H como se muestra a continuación:

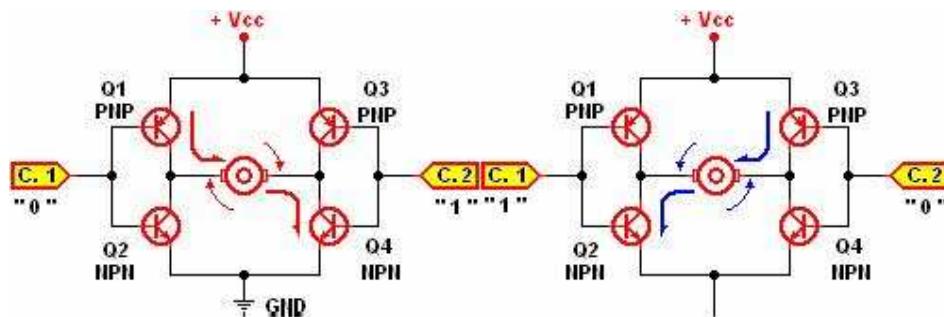


Figura 2.16 - Motor de continua con puente H

Si bien podría pensarse que es más sencilla la implementación con una fuente simétrica o con dos fuentes de alimentación, se decidió utilizar la implementación en la que se utiliza una fuente sencilla para evitar la complejidad de trabajar con una fuente simétrica. Es importante aclarar que existen circuitos integrados que implementan un puente H, por lo que la implementación de esta opción no llevará a complicaciones considerables. En el caso de este proyecto se decidió trabajar con el integrado L293D.

2.7.2 Motor PaP (Dirección)

Los motores PaP son motores cuyo giro se realiza a intervalos regulares en lugar de realizarse de continuo, como en el caso de los motores de continua. Estos motores giran en función de una secuencia de pulsos aplicados a sus devanados. El eje del motor gira



un determinado ángulo por cada impulso de entrada, este ángulo se denomina **paso** y es uno de los parámetros fundamentales del motor.

Los motores PaP pueden ser Bipolares o Unipolares.

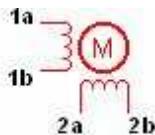


Figura 2.17 - Motor PaP bipolar

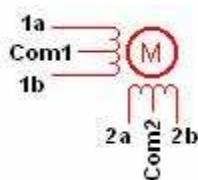


Figura 2.18 - Motor PaP unipolar

Los motores PaP bipolares utilizan dos bobinados independientes mientras que los motores PaP unipolares es como si tuvieran cuatro bobinados ya que tienen un terminal central que es el común de cada par de bobinados. Por este motivo los motores PaP bipolares tienen cuatro cables y los motores PaP unipolares tienen cinco o seis cables dependiendo de si Com1 y Com2 están unidos o no. En el caso de este proyecto se utilizó un motor PaP bipolar.

Para controlar los motores PaP es necesario invertir las polaridades de los terminales de las bobinas en una determinada secuencia para lograr un giro en un sentido y en secuencia opuesta para que gire en el otro sentido. Con este propósito se utilizarán los dos puentes de un driver L293D.

Para controlar el motor se debe tener en cuenta la siguiente tabla:

Número de Paso	RB4	RB5	RB6	RB7
Paso 1	+ Vcc	GND	+ Vcc	GND
Paso 2	+ Vcc	GND	GND	+ Vcc
Paso 3	GND	+ Vcc	GND	+ Vcc
Paso 4	GND	+ Vcc	+ Vcc	GND

Tabla 2-2 - Secuencia de pasos motor PaP



2.7.3 Driver L293D^{ix}

Como se mencionó anteriormente, la lógica de control del móvil será desarrollada a nivel de un PIC 16F877A. Por lo general la corriente de salida de un PIC es aproximadamente 25 mA, la cual es insuficiente para mover los motores, por lo que resulta necesario amplificarla de algún modo.

Se investigaron los circuitos integrados existentes que implementaran la función de amplificación y que permitieran el control de los motores. El integrado más comúnmente utilizado para aplicaciones similares a la de este proyecto es el L293. Se investigaron las variedades de L293 y se optó por utilizar el L293D, ya que éste incluye diodos de protección.

El driver L293D es un circuito integrado que implementa dos puentes H de transistores y está diseñado para aceptar niveles lógicos TTL o DTL estándar. Además, incluye diodos internos de protección para que en el momento en que se apaga el motor, el pico de tensión inversa que genera la inductancia del mismo no queme el circuito integrado.

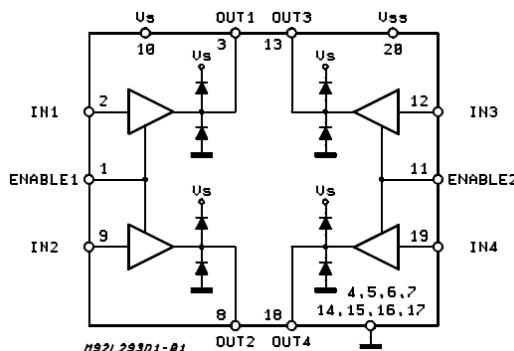


Figura 2.19 - Esquema interno L293D

Este integrado dispone de dos entradas: Enable 1 y Enable 2, las cuales permiten habilitar/deshabilitar los puentes 1 y 2 respectivamente, independientemente de otras señales de entrada.

El L293D dispone de dos voltajes de alimentación: Vs, que es el voltaje de alimentación para las etapas de salida de potencia (para los motores) y Vss, que es el voltaje de alimentación para la lógica. Vs puede ser 36 V como máximo y Vss debe estar entre 4.5 V y 36 V.



Se recomienda conectar un capacitor no inductivo (por lo general de 100nF) entre V_s y tierra y entre V_{ss} y tierra, lo más cerca posible al integrado.

Por último, es recomendable que la tierra asociada a la alimentación de la lógica del integrado y la tierra asociada a la alimentación de las etapas de salida de potencia estén separadas.



2.8 Manejo de sensores

El móvil dispone de tres sensores: un sensor que simula el sensor de vida y dos sensores de obstáculos (para poder detectar los obstáculos que se presenten en su recorrido y así poder esquivarlos). En las siguientes secciones se describen estos sensores en más detalle.

2.8.1 Simulación del sensor de vida

Inicialmente se decidió simular el sensor de vida con un sensor de sonido. Posteriormente, este sensor fue sustituido por un sensor infrarrojo (IR). Como sensor de sonido se utilizó un circuito compuesto por un micrófono, un amplificador operacional, un detector de tono, resistencias, capacitores y un potenciómetro. A continuación se describe el diseño inicial de este sensor.

El micrófono capta los sonidos del ambiente y entrega una señal de voltaje. Mediante pruebas prácticas se analizó la señal que entrega el micrófono al reproducir a través de un auricular un tono de 1kHz. Se pudo observar una sinusoides aproximadamente centrada en 0 V de aproximadamente 35 mV pico a pico con frecuencia 1 kHz.

El menor voltaje de entrada que puede captar el detector de tono es de 20 mVrms. Por este motivo, es necesaria una pequeña amplificación para entregar la salida del micrófono al detector de tono. Además, a la señal de salida del micrófono se le superpone ruido eléctrico. Por estos motivos se implementó un filtro activo utilizando un amplificador operacional, el cual permite filtrar el ruido eléctrico y amplificar la señal.

La señal que se obtiene a la salida del filtro es entregada al detector de tono, el cual hará cambiar de estado un relé cuando detecte una señal de la frecuencia configurada. Esto hará que el PIC reciba un “1” lógico en la entrada correspondiente al sensor de sonido.

2.8.1.1 Micrófono^x

Los micrófonos son transductores que detectan señales de sonido y producen una imagen eléctrica del sonido, es decir, producen un voltaje o una corriente que es proporcional a la señal de sonido.



Existen distintos tipos de micrófonos: dinámicos, de cinta, de condensador, de cristal y electret. En este proyecto se utiliza un micrófono electret por lo que se describirá este tipo en más detalle.

2.8.1.1.1 Micrófono electret

Los micrófonos electret son micrófonos de condensador que usan un material electret para sus diafragmas, el cual está permanentemente polarizado. De esta forma se evita el voltaje de continua de polarización que es necesario en los micrófonos de condensador convencionales.

Los micrófonos electret de mejor calidad incorporan un preamplificador FET (transistor de efecto de campo) para emparejar su alta impedancia y amplificar la señal.

En la Figura 2.20 se muestra un esquema del micrófono electret.

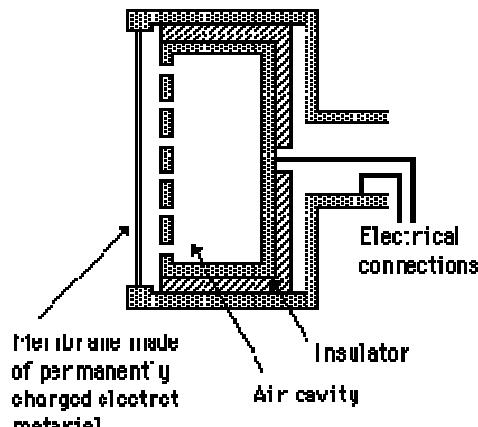


Figura 2.20 - Esquema micrófono electret

El material electret del diafragma puede tener un espesor menor a una milésima de pulgada. Aún así, está lo suficientemente polarizado para producir un cambio activo capacitivo en el voltaje entre la membrana y el plato posterior cuando ésta es movida por la presión de una onda de sonido.

La polarización se logra a través de una combinación de calor y alto voltaje durante la fabricación. El diafragma está respaldado por una película de metal evaporado.



2.8.1.1.2 Alimentación del micrófono^{xi}

Los micrófonos electret necesitan energía para funcionar. Esta energía (típicamente entre 4 y 10 V) es proporcionada al micrófono a través de una resistencia (típicamente entre 1 y 10 kΩ), llamada resistencia de polarización. Las cápsulas de los micrófonos electret por lo general consumen entre 0.5 y 2 mA de corriente al operar.

Existen micrófonos electret de dos polos y micrófonos electret de tres polos. Siempre uno de ellos se debe conectar a tierra. En el caso de los micrófonos de tres polos, uno de ellos es para alimentar al micrófono, otro es el polo de señal y el otro va conectado a tierra. En el caso de los micrófonos de dos polos, se utiliza el mismo polo para la alimentación y para obtener la señal.

La señal de salida tiene una pequeña componente de continua proveniente de la corriente de polarización, por lo que suele agregarse un condensador adecuado en serie con la salida de audio.

En la Figura 2.21 se muestra un diagrama de alimentación de un micrófono electret. A modo de ejemplo se muestra para un electret con tres polos.

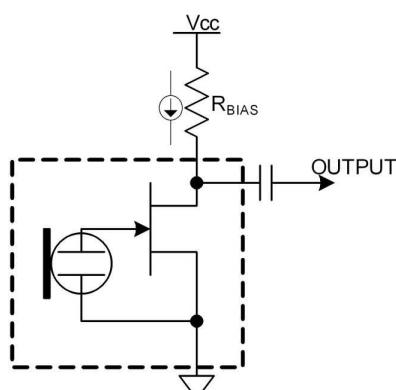


Figura 2.21 - Alimentación de micrófono electret

2.8.1.2 Filtros activos^{xii}

Un filtro es un dispositivo que permite pasar todos los armónicos de la señal de entrada que se encuentren en ciertos rangos de frecuencia, mientras evita el pasaje de otros armónicos.



En el caso de este proyecto, fue necesario agregar un filtro entre el micrófono y el detector de tono para eliminar el ruido eléctrico que se superponía a la señal de salida del micrófono.

A frecuencias altas ($> 1 \text{ MHz}$), los filtros suelen consistir en componentes pasivos tales como inductores (L), resistencias (R) y capacitores (C), por lo que se llaman filtros LRC.

En el rango de frecuencias bajas (1 Hz – 1 MHz), sin embargo, el valor de los inductores sería muy alto y el inductor en sí sería muy voluminoso. En estos casos, los filtros activos se vuelven importantes.

Los filtros activos son circuitos que utilizan un opamp como el dispositivo activo en combinación con algunas resistencias y capacitores para proveer un desempeño similar al de los filtros LRC pero a bajas frecuencias.

En la Figura 2.22 se compara un filtro pasa bajo pasivo de segundo orden con un filtro pasa bajo activo de segundo orden.

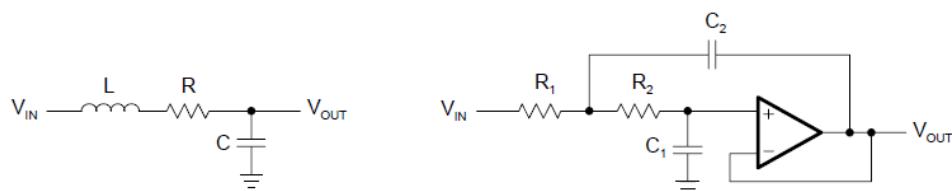


Figura 2.22 - Filtro pasivo vs. Filtro activo

En este proyecto fue necesario implementar un filtro pasa banda centrado en 1 kHz, ya que el tono que se desea que sea detectado por el sensor de sonido tiene esta frecuencia. Por este motivo, en esta sección se describirán más en detalle los filtros activos pasa banda.

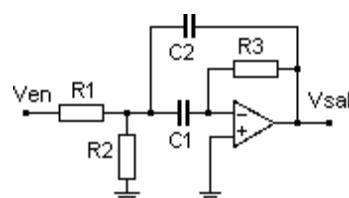


Figura 2.23 - Filtro activo pasa banda



El *filtro pasa banda* tiene una curva de respuesta en frecuencia como la que se muestra en la Figura 2.24. Dejará pasar todos los armónicos de la señal de entrada que se encuentren entre la frecuencia de corte inferior f_1 y la de corte superior f_2 . Los armónicos que se encuentren fuera de este rango de frecuencias serán atenuados y tendrán una amplitud menor al 70.7 % de su amplitud de entrada. La frecuencia central de este tipo de filtro se obtiene con la siguiente fórmula:

$$f_0 = 1 / [2\pi C \times (R_3 R)^{1/2}]$$

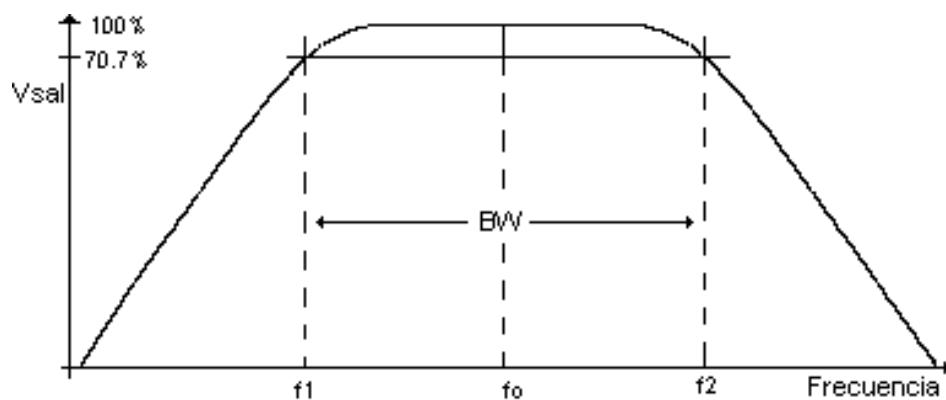


Figura 2.24 - Respuesta en frecuencia de un filtro pasa banda

Si se seleccionan los capacitores y resistores de modo que: $C_1 = C_2 = C$ y $R_1 = R_2 = R$, el ancho de banda será:

$$BW = f_2 - f_1 = 1.41 R / [CR_3 (R_3 R)^{1/2}]$$

Y el factor de calidad será:

$$Q = f_0 / BW$$

Las líneas discontinuas verticales sobre f_1 y f_2 y la línea horizontal del 70.7% representan la respuesta de un filtro pasa banda ideal.

Las frecuencias de corte (f_1 y f_2) son puntos en la curva de transferencia en que la salida ha caído 3 dB desde su valor máximo.



2.8.1.3 Detector de tono^{xiii}

Como detector de tono se utilizó el integrado LM567, cuyo diagrama de bloques se muestra a continuación:

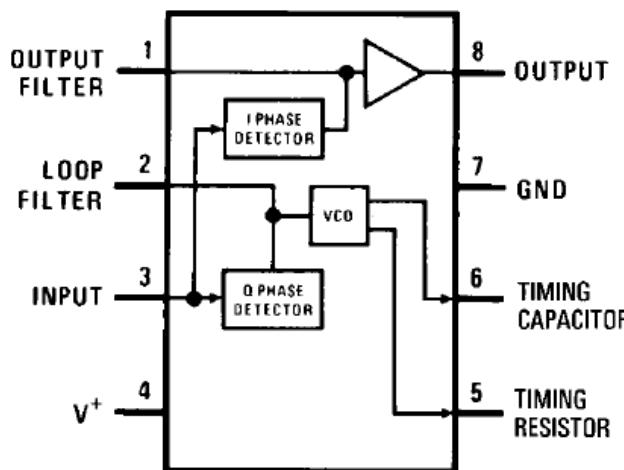


Figura 2.25 - Diagrama de bloques LM567

La frecuencia y el ancho de banda de detección se controlan con capacitores y resistencias externos. Para dimensionar el circuito se tuvieron en cuenta las siguientes ecuaciones:

$$f_0 = \frac{1}{1.1R_1 C_1}$$

$$BW = 1070 \sqrt{\frac{V_I}{f_0 C_2}} \text{ in \% of } f_0$$

$$V_I \leq 200mV_{RMS}$$

Where

V_I = Input voltage (V_{RMS})

C_2 = Low-pass filter capacitor (μF)

Figura 2.26 - Ecuaciones LM567

También se tuvieron en cuenta las siguientes recomendaciones mencionadas en la hoja de datos:

1. Se recomienda que R_1 esté entre 2 k Ω y 20 k Ω para mejor estabilidad de temperatura.



2. Para seleccionar el capacitor pasa-bajo C_2 referirse a la gráfica “Bandwidth vs. Input signal amplitude”.

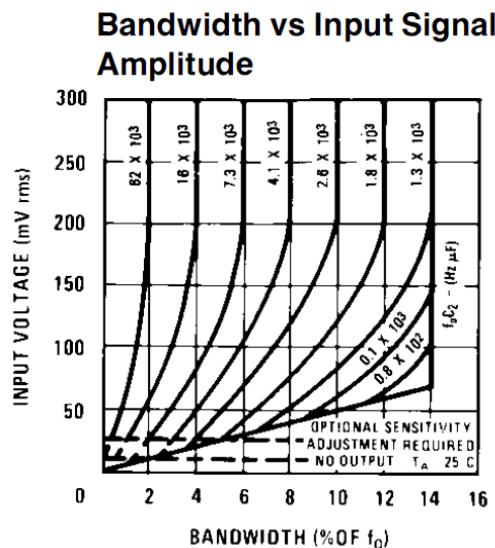


Figura 2.27 - Ancho de banda vs amplitud de la señal de entrada

3. Por lo general, el valor de C_3 no es crítico. C_3 establece el borde de la banda de un filtro pasa-bajo que atenúa las frecuencias fuera de la banda de detección para eliminar salidas espurias. Un valor típico para C_3 es $2C_2$.

4. Opcionalmente puede conectarse una resistencia entre los pines 1 (Output filter capacitor C_3) y 4 (Supply Voltage), el cual establece el umbral para el voltaje de entrada más alto que no produce salida. Para asegurar el límite testeado de $10 \text{ mV}_{\text{RMS}}$ como mínimo, se utiliza un valor de $130 \text{ k}\Omega$ para R_2 .

Cuando el integrado detecta una entrada cuya frecuencia está en la banda pasante y cuyo voltaje supera cierto umbral, se obtiene un “0” lógico en la salida.

2.8.1.4 Relé

Para que el PIC reciba un “1” lógico cuando el detector de tono entregue un “0” lógico es necesario invertir esta señal de alguna forma. Inicialmente se decidió utilizar un relé para cumplir esta función.



Un relé tiene contactos normalmente abiertos (NA), es decir que conectan el circuito cuando el relé es activado y contactos normalmente cerrados (NC), es decir que, cuando el relé es activado, desconectan el circuito. El relé también tiene un contacto de conmutación, el cual controla dos circuitos: unos NA y uno NC con una terminal común.

En este proyecto se conectó el contacto NA a 5 V y el NC a tierra. Cuando el detector de tono entrega un “0” lógico en su salida, circula corriente por la bobina del relé. En este momento se conecta el contacto de conmutación con el contacto NA, es decir que se obtienen 5 V en el contacto de conmutación, el cual es conectado a una entrada del PIC.

Esta solución no dio resultado debido a que la corriente de salida del detector de tono no era suficiente para accionar el relé. Por este motivo se decidió utilizar una compuerta NOT en lugar del relé, para invertir la salida del detector de tono.

2.8.1.5 Compuerta NOT

La compuerta NOT, también llamada compuerta inversora, es una compuerta lógica que tiene una entrada y una salida y que en su salida entrega el opuesto lógico de la entrada. Es decir que si en la entrada de la compuerta NOT hay un “1” lógico, en su salida habrá un “0” lógico. Si en la entrada de la compuerta NOT hay un “0” lógico, en su salida habrá un “1” lógico.

El símbolo utilizado para representar esta compuerta se muestra en la Figura 2.28.

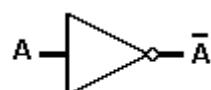


Figura 2.28 - Símbolo compuerta NOT

A continuación se muestra la tabla de verdad correspondiente a esta compuerta lógica:

A	Ā
0	1
1	0

Tabla 2-3 - Tabla de verdad NOT



2.8.2 Sensor infrarrojo

Luego de múltiples pruebas y modificaciones sobre el sensor de sonido, sin lograr el correcto funcionamiento del mismo, se decidió simular el sensor de vida con un sensor infrarrojo (IR) y no con un sensor de sonido.

2.8.2.1 Luz infrarroja

La luz infrarroja (IR) es un tipo de luz no visible, cuya longitud de onda es 850-900 nm, por lo que en el espectro se encuentra entre la luz visible y las microondas. La luz IR contiene información que no contiene la luz visible, contiene información respecto de la temperatura de un cuerpo. Cualquier cuerpo cuya temperatura sea mayor que 0 Kelvin (K) emite radiación IR.

2.8.2.2 Emisor IR

Como emisor IR se utilizó un LED (Light Emitting Diode) IR, el cual es un tipo especial de LED. Un LED es un dispositivo semiconductor que cuando se polariza en directa y es atravesado por una corriente eléctrica emite radiaciones electromagnéticas en una estrecha banda de longitudes de onda, dependiendo del semiconductor con el que esté fabricado. En el caso del diodo IR, las radiaciones electromagnéticas que emite están en la banda de longitudes de onda del IR.

2.8.2.3 Receptor IR

Como receptor IR se utilizó un fototransistor IR. Se llama fototransistor a un transistor sensible a la luz, normalmente a la luz IR. En este tipo de fototransistores, la base está reemplazada por un cristal fotosensible (por lo general, de silicio) que cuando recibe luz, produce una corriente y desbloquea el transistor. En el fototransistor la corriente circula solo en un sentido y el bloqueo del transistor depende de la luz; cuanta más luz hay, más corriente conduce.

2.8.3 Sensores de obstáculos

Como sensores de obstáculos se utilizaron microswitches que cambian de estado cuando uno de los dos paragolpes del móvil choca contra algo. Debido a que probablemente el



móvil choque y se aleje un poco del obstáculo, los interruptores entregarán un “1” lógico por un breve período de tiempo. Es decir que generarán un pulso de corta duración. Para retener este pulso y dar tiempo a que el programa del PIC lea esta información, se implementó un flip-flop RS asíncrono.

2.8.3.1 Microswitches

Un microswitch es un interruptor eléctrico que puede cambiar de estado con una fuerza física muy pequeña, a través del uso de un mecanismo de punto de inflexión. Es decir que el dispositivo se encuentra en un estado de equilibrio estable y al ser presionado el botón pasa a otro estado de equilibrio, distinto del primero.

Internamente, al presionar el botón se dobla una tira de metal rígido, activando el interruptor. Al remover la presión sobre el botón, la tira de metal vuelve a su estado original.

Existen diversos tipos de microswitches. En la Figura 2.29 se muestra el esquema de un microswitch similar al que se utilizó en este proyecto.

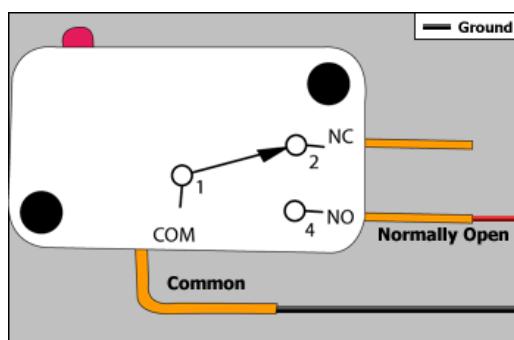


Figura 2.29 - Esquema microswitch

Como se puede observar en la figura, el microswitch dispone de un contacto NC, de un contacto NA y de un contacto común. Mientras el botón no se encuentra presionado, el común está conectado con el contacto NC. Mientras el botón se encuentre presionado, el común estará conectado al contacto NA.

En este proyecto se utilizó el común como salida y se conectó el contacto NC a tierra y el contacto NA a 5 V, por lo que al presionar el botón se generará un pulso de 5 V que será leído como un “1” lógico.



2.8.3.2 Flip-Flop RS asíncrono

Para retener los pulsos generados por los microswitches, se utilizaron flip-flops RS asíncronos.

Los flip-flops son los elementos básicos de memoria. Estando el flip-flop alimentado, puede mantener un estado binario indefinidamente hasta que reciba una señal de entrada que lo haga cambiar de estado.

Existen distintos tipos de flip-flops: RS, JK, T y D. Los últimos tres se pueden implementar a partir del primero. Además, cualquiera de estos puede ser de dos tipos: activado por nivel o maestro-esclavo. El primero actúa de acuerdo a los niveles de amplitud que recibe “0” o “1”, mientras que el segundo en realidad son dos flip-flops combinados de manera tal que uno “hace caso” al otro.

Por otro lado, los flip-flops pueden ser síncronos o asíncronos. Los primeros tienen una entrada de reloj y responden a los niveles de entrada durante la ocurrencia del reloj, mientras que los otros no utilizan reloj.

Teniendo en cuenta que los flip-flops JK, T y D se construyen a partir del flip-flop RS y que en este proyecto se utilizaron flip-flops RS asíncronos, en esta sección se describirá más en detalle este tipo de flip-flop.

El flip-flop RS asíncrono tiene dos salidas, Q y Q', y dos entradas, S (set) y R (reset).

Un flip-flop RS asíncrono puede implementarse utilizando compuertas NOR o bien utilizando compuertas NAND. A continuación se muestra la tabla de verdad con las dos posibles implementaciones:

R	S	Q (NOR)	Q' (NAND)
0	0	Q	N.D.
0	1	1	0
1	0	0	1
1	1	N.D.	Q

N.D. = Estado no determinado
Q = Estado anterior

Tabla 2-4 - Tabla de verdad flip-flop RS



Se optó por la implementación con compuertas NOR debido a que para esta aplicación se desea que si ambas entradas del flip-flop (Set y Reset) están en cero, el flip-flop mantenga su estado. En la Figura 2.30 se muestra un esquema de esta implementación.

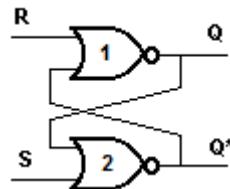


Figura 2.30 - Flip-flop RS con compuertas NOR

Para analizar el funcionamiento de un flip-flop RS implementado con compuertas NOR hay que tener en cuenta la tabla de verdad de estas compuertas, la cual se muestra en la Tabla 2-5.

Entrada A	Entrada B	Salida
0	0	1
0	1	0
1	0	0
1	1	0

Tabla 2-5 - Tabla de verdad NOR

Se asume como punto de partida que la entrada S está en “1” y que la entrada R está en “0”. Como la compuerta 2 tiene una entrada en “1”, su salida, Q’, es “0”. Esto coloca ambas entradas de la compuerta 1 en “0”, por lo que su salida, Q, es “1”.

Si la entrada S cambia a “0”, las salidas permanecerán iguales ya que la salida Q permanece como “1”, dejando una entrada de la compuerta 2 en “1”. Esto causa que la salida Q’ permanezca en “0”, por lo que ambas entradas de la compuerta 1 estarán en “0” y la salida Q será “1”.

De forma similar, si la entrada R se pone en “1”, la salida Q cambia a “0” y la salida Q’ cambia a “1”. Cuando la entrada R cambia a “0”, las salidas no cambian. Cuando las dos entradas, R y S, se ponen en “1”, las dos salidas, Q y Q’, quedan en “0”. Esta situación viola el hecho de que las salidas Q y Q’ son complementarias entre sí. En operación normal esta situación debe evitarse.



2.9 Comunicación entre el PIC y la placa BB

Como se explica en el capítulo 4, al intentar comunicar el PIC con la placa BB se detectó el problema de que éstas manejan lógicas de distintos niveles. Para solucionar este problema se utilizaron transistores como interruptores.

Se conectó la fuente de 5 V al colector de un transistor NPN a través de una resistencia; se conectó el emisor del transistor a tierra y se conectó la salida de la placa BB a la base a través de otra resistencia. La señal en el colector se utilizó como señal de salida.

La Figura 2.31 muestra un esquema de la conexión utilizada.

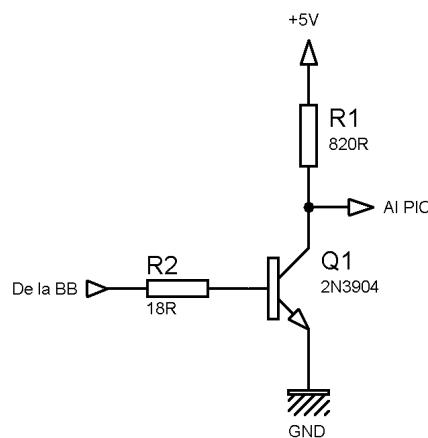


Figura 2.31 - Transistor como interruptor

En esta configuración, el transistor puede estar en uno de dos estados: apagado o encendido. Cuando la entrada es de 0 V, se dice que el interruptor está encendido. El voltaje V_{CE} (colector-emisor) es casi cero y el transistor está en corte. Es decir que no circula corriente por el colector. Al no circular corriente por el colector, no hay caída de tensión en la resistencia R1 y, por lo tanto, la señal de salida es de 5 V.

Cuando la entrada es el nivel alto de la placa BB (1.8 V), como la resistencia colocada en la base del transistor (R2) es pequeña, la corriente de base será grande, por lo que inducirá en el colector una corriente β veces más grande. El transistor estará saturado, es decir que la corriente de colector será máxima por lo que la señal de salida será de 0 V.



2.10 Fuente de alimentación^{xiv}

Teniendo en cuenta que el mayor voltaje que se requiere para alimentar a los componentes del sistema es de 12 V, se decidió utilizar como fuente de alimentación una fuente de continua de 12 V. Ciertos componentes de los circuitos electrónicos diseñados para el sistema de este proyecto requieren ser alimentados con voltajes menores a los 12V de dicha fuente, por lo tanto fue necesario utilizar reguladores de voltaje.

Se optó por trabajar con reguladores de la familia LM78XX, los cuales son reguladores positivos de tres terminales que ofrecen distintos valores fijos de voltaje de salida. Esta familia de reguladores emplea limitadores internos de corriente, corte térmico y protección de área para una operación segura. Estas características los hacen prácticamente indestructibles.

A continuación se muestra el diagrama de bloques de esta familia de reguladores:

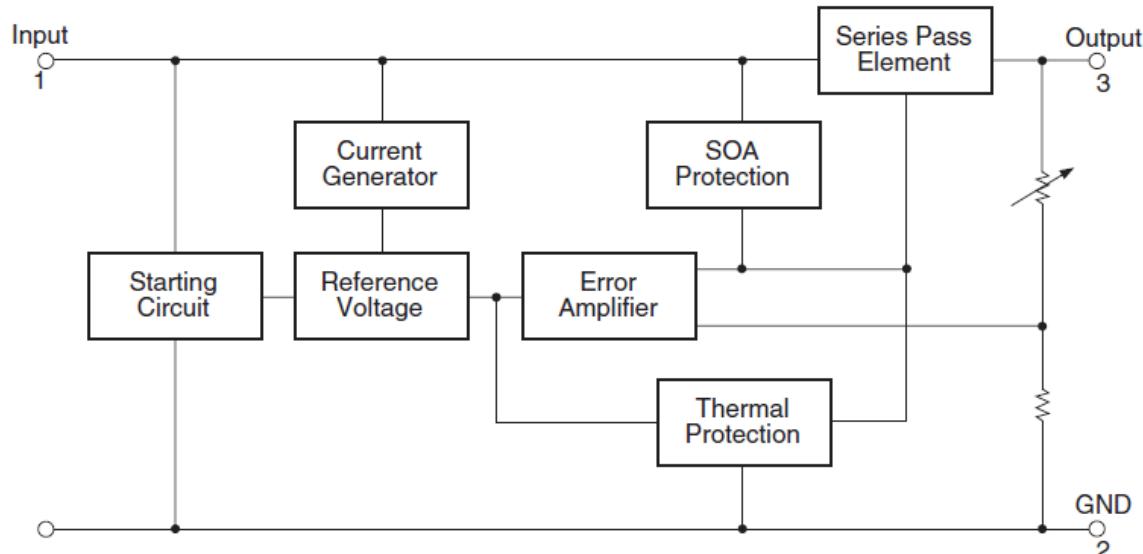


Figura 2.32 - Diagrama de bloques regulador de voltaje

La familia de reguladores LM78XX ofrece voltajes de salida de entre 5 V y 18 V para un voltaje de entrada de hasta 35 V, con la condición de que el voltaje de entrada esté al menos 2 V por encima del voltaje de salida.



El circuito recomendado en la hoja de datos para la utilización de estos reguladores con el fin de obtener un voltaje de salida fijo es el siguiente:

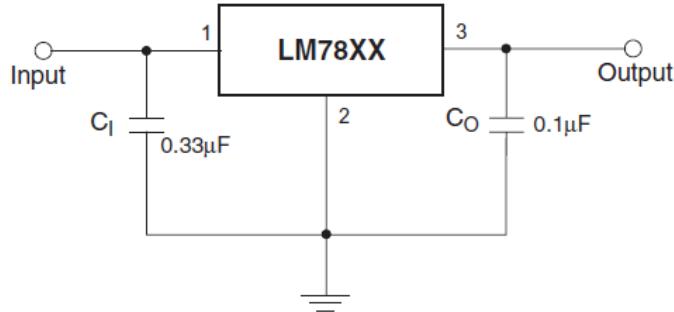


Figura 2.33 - Regulación de voltaje

La capacitancia C_I es necesaria si el regulador está lejos del filtro de la fuente de alimentación. La capacitancia C_O mejora la estabilidad y la respuesta transitoria del regulador.

Para especificar el voltaje de salida se debe sustituir "XX" por el valor deseado. En el caso de este proyecto se requiere obtener voltajes de salida de 5 V y de 9 V, por lo tanto se utilizará un regulador LM7805 y un regulador LM7809.



2.11 Global Positioning System (GPS)^{xv}

2.11.1 Introducción

El Global Positioning System (GPS o en español Sistema de Posicionamiento Global) es un sistema de radionavegación espacial de los Estados Unidos (EEUU), el cual provee servicios confiables de posicionamiento, navegación y reloj a usuarios civiles en una base continua y que abarca todo el mundo.

A cualquiera que tenga un receptor GPS, el sistema le proveerá la posición y la hora. GPS provee posición precisa e información de hora a un número ilimitado de personas en cualquier parte del mundo.

GPS se compone de tres partes: satélites orbitando la Tierra, estaciones de control y monitoreo en la Tierra y los receptores GPS de los usuarios. Los satélites GPS envían señales desde el espacio, las cuales son recibidas e identificadas por los receptores GPS. Cada receptor GPS luego provee la posición tridimensional (latitud, longitud y altitud) y, además, la hora.

El sistema de satélites que orbita la Tierra está formado por 24 unidades con trayectorias sincronizadas para cubrir toda la superficie del globo terráqueo. Más concretamente, repartidos en 6 planos orbitales de 4 satélites cada uno. Además, hay 3 satélites de respaldo. La energía eléctrica que requieren para su funcionamiento la adquieren a partir de dos paneles compuestos de celdas solares adosados a sus costados.

Las estaciones terrestres envían información de control a los satélites para controlar las órbitas y realizar el mantenimiento de toda la constelación.

2.11.2 Funcionamiento

La situación de los satélites es conocida por el receptor con base en las efemérides, parámetros que son transmitidos por los propios satélites. La colección de efemérides de toda la constelación se completa cada 12 minutos y se guarda en el receptor GPS.

El receptor GPS funciona midiendo su distancia a los satélites, y usa esa información para calcular su posición. Esta distancia se mide calculando el tiempo que la señal tarda



en llegar al receptor. Conocido ese tiempo y basándose en el hecho de que la señal viaja a la velocidad de la luz (salvo algunas correcciones que se aplican), se puede calcular la distancia entre el receptor y el satélite.

Cada satélite indica que el receptor se encuentra en un punto en la superficie de una esfera con centro en el propio satélite y de radio la distancia total hasta el receptor. Obteniendo información de dos satélites se nos indica que el receptor se encuentra sobre la circunferencia que resulta cuando se intersectan las dos esferas. Si adquirimos la misma información de un tercer satélite notamos que la nueva esfera solo corta la circunferencia anterior en dos puntos. Uno de ellos se puede descartar porque ofrece una posición absurda. De esta manera ya tendríamos la posición en 3-D.

Sin embargo, dado que el reloj que incorporan los receptores GPS no está sincronizado con los relojes atómicos de los satélites GPS, los dos puntos determinados no son precisos. Teniendo información de un cuarto satélite, eliminamos el inconveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición 3-D exacta (latitud, longitud y altitud).

Al no estar sincronizados los relojes entre el receptor y los satélites, la intersección de las cuatro esferas con centro en estos satélites es un pequeño volumen en vez de ser un punto. La corrección consiste en ajustar la hora del receptor de tal forma que este volumen se transforme en un punto.

2.11.3 Fiabilidad de los datos

Debido al carácter militar del sistema GPS, el Departamento de Defensa de los EE. UU. se reservaba la posibilidad de incluir un cierto grado de error aleatorio, que podía variar de los 15 a los 100 m. La llamada **disponibilidad selectiva** (S/A) fue eliminada el 2 de mayo de 2000. Aunque actualmente no se aplique tal error inducido, la precisión intrínseca del sistema GPS depende del número de satélites visibles en un momento y posición determinados.

Con un elevado número de satélites siendo captados (7, 8 ó 9 satélites), y si éstos tienen una geometría adecuada (están dispersos), pueden obtenerse precisiones inferiores a 2,5 metros en el 95% del tiempo.



Si se activa el sistema (Differential GPS) DGPS llamado SBS (WAAS-EGNOS-MSAS), la precisión mejora siendo inferior a un metro en el 97% de los casos. El sistema SBS se basa en la existencia de satélites geoestacionarios cuya posición es conocida, por lo que se puede utilizar para corregir la posición del receptor GPS. Estos sistemas SBS no aplican en Sudamérica, ya que esta parte del mundo no cuenta con este tipo de satélites geoestacionarios.



2.12 Datos NMEA^{xvi}

2.12.1 Introducción

La Asociación Nacional de Electrónica Marina (National Marine Electronics Association – NMEA) ha desarrollado una especificación que define la interfaz entre varios equipos de electrónica marina. El estándar permite a la electrónica marina enviar información a las computadoras y a otros equipos marinos.

El estándar NMEA 0183 define una interfaz eléctrica y un protocolo de datos para las comunicaciones entre instrumentación marina. La comunicación de los receptores GPS está definida en dicha especificación. La mayoría de los programas de computadora que proveen información de posición en tiempo real entienden NMEA y esperan que los datos estén en el formato NMEA. Estos datos incluyen la solución completa PVT (posición, velocidad y tiempo) computada por el receptor GPS.

La idea de NMEA es enviar una línea de datos, llamada sentencia, que es totalmente autocontenido e independiente de otras sentencias. Las sentencias son estándar para cada categoría de dispositivo y también es posible definir sentencias propietarias para uso de cada compañía. Todas las sentencias estándar tienen un prefijo de dos letras que define el dispositivo que usa ese tipo de sentencia. (Para receptores GPS el prefijo es GP.) Este prefijo va seguido de una secuencia de tres letras que define el contenido de la sentencia.

Además, NMEA permite que los fabricantes de equipos definan sus propias sentencias propietarias. Todas las sentencias propietarias comienzan con la letra “P” y están seguidas de tres letras que identifican al fabricante que controla esa sentencia. Por ejemplo, una sentencia Garmin empezaría con PGRM y una sentencia Magellan empezaría con PMGN.

2.12.2 Sentencias NMEA

Cada sentencia comienza con un “\$” y termina con una secuencia retorno de carro/alimentación de línea y no puede tener más de 80 caracteres de texto visible. Los datos están contenidos en esa única línea y los ítems de datos están separados por comas. Los datos en sí son texto ASCII.



Está provisto un checksum al final de cada sentencia, el cual puede o no ser controlado por la unidad que lee los datos. El campo de checksum consiste en un “*” y dos dígitos hexadecimales que representan un OR exclusivo de 8 bits de todos los caracteres entre el “\$” y el “*” (sin incluir estos símbolos).

2.12.3 Interfaz de hardware

La interfaz de hardware de las unidades GPS está diseñada para cumplir los requerimientos de NMEA. Además, es compatible con el puerto serial de la mayoría de las computadoras que utilizan RS232, aunque estrictamente el estándar NMEA no es RS232.

La velocidad de la interfaz se puede ajustar en algunos modelos pero el estándar NMEA establece 4800 bps con ocho bits de datos, sin paridad y un bit de parada. Todas las unidades que soportan NMEA deberían soportar esta velocidad.

NMEA está diseñado para correr como un proceso en el background enviando sentencias que luego son capturadas cuando sea necesario por el programa que se utilice.



2.13 *Información de posición^{xvii}*

2.13.1 Introducción

Un punto cualquiera de la superficie terrestre se localiza por su longitud y latitud. Las distancias se expresan con unidades de medida angular, es decir grados, minutos y segundos, sexagesimales o en grados decimales. El problema es que la forma de la tierra no es regular, sino que es un geoide.

Para realizar cálculos de posicionamiento, distancias, etc. se utiliza como base un cuerpo geométrico definido denominado elipsoide de revolución, debido a que éste es el que más se parece al geoide.

Diferentes países han adoptado referencias a diversos elipsoides de revolución ya que determinado elipsoide de revolución que es bueno para determinada área o país, no se adapta a otra área geográfica. Cada región o país, trata de que la superficie del elipsoide de revolución coincida con la del geoide en la mayor extensión posible de su territorio. Pero lo que sucede habitualmente es que al tratar de hacer coincidir estas superficies, el centro del elipsoide de revolución se desplaza un poco respecto del centro del geoide. El ajuste se realiza determinando un punto del territorio, llamado punto fundamental, donde se hace que la vertical del geoide coincida con la normal al elipsoide de revolución.

A este elipsoide de revolución se lo llama elipsoide de referencia y al conjunto de todos estos datos, Datum.

2.13.2 Tipos de Datum

Se pueden diferenciar dos tipos de Datum:

- **Centrados y coincidentes con el centro de masa de la Tierra.** El WGS 1984, es ampliamente usado y sirve para mediciones en el ámbito mundial. La definición del mismo parte de las mediciones realizadas por los satélites en los últimos años.
- **Locales.** Son los que se utilizan para hacer corresponder el geoide con el elipsoide de revolución en determinada localización geográfica.



Antes del GPS los Datum eran referencias locales, pero a partir del GPS se pudo crear un Datum universal y se adoptó un elipsoide de revolución geocéntrico cuyo punto fundamental es el centro de la tierra.

2.13.3 De la esfera al plano

2.13.3.1 Coordenadas Cartográficas

Para trasladar las posiciones tridimensionales de la tierra al plano (una carta, un mapa, etc.), se utilizan proyecciones.

Un sistema coordenado plano es una grilla sobre la cual se puede dibujar un mapa en dos dimensiones. La elaboración de un mapa requiere un método por el cual se hace corresponder a cada punto de la Tierra con un punto en el mapa. Para obtener esa correspondencia, se utilizan las proyecciones cartográficas.

Al pasar de un elipsoide de revolución a un plano se producen deformaciones en la forma, en el área, en la dirección o en la distancia. En general determinadas proyecciones mantienen constantes algunos atributos para zonas pequeñas y localizadas. La elección de determinada proyección se realiza en función de los objetivos del proyecto y de las propiedades que se desea preservar.

2.13.3.2 Tipos de proyecciones

Para pasar de la esfera al plano se adoptan distintos métodos, los cuales difieren en la superficie de proyección utilizada. Existen proyecciones cónicas, cilíndricas y planas. El Datum Yacaré, que es el utilizado en Uruguay y el que se utilizará para este proyecto, se basa en la proyección Gauss-Krüger, la cual es una proyección cilíndrica. Por este motivo se explicará este tipo de proyección en mayor detalle.

2.13.3.2.1 Proyección Cilíndrica

Para realizar una proyección cilíndrica se rodea la esfera con un cilindro y se proyectan los paralelos y meridianos sobre el mismo. Luego, se corta el cilindro a lo largo de una generatriz y se extiende el mismo sobre el plano. De esta forma se obtiene un sistema de meridianos y paralelos sobre el que puede dibujarse el mapa.



Las proyecciones así obtenidas tienen la totalidad de la información de la Tierra en una superficie continua. En esta proyección los meridianos son rectas verticales equidistantes y los paralelos son perpendiculares a los meridianos. Mientras la distancia entre meridianos se mantiene, la distancia entre paralelos se acorta al acercarse a los polos.



Figura 2.34 - Proyección cilíndrica

2.13.3.3 Sistema Gauss-Krüger

Se lo conoce también como Mercator Transversa. El sistema Gauss-Krüger es el sistema geométrico de referencia empleado para expresar numéricamente la posición geodésica de un punto sobre el terreno en Uruguay.

Desde este punto de partida, la ubicación de un punto se establece en relaciones de distancia a una grilla. Es representado por dos números: uno correspondiente a la distancia al eje x y otro asociado a la distancia al eje y.

Se utiliza un cilindro transverso como superficie de proyección donde se define un meridiano central como lugar de contacto con la tierra (en lugar del Ecuador). El resultado es una proyección conforme que no mantiene las direcciones. A lo largo del meridiano central no se observan deformaciones.

2.13.3.4 Parámetros

Para poder definir el sistema coordenado es necesario conocer determinados parámetros.



2.13.3.4.1 Parámetros lineales

- **Falso Este:** Es un valor lineal aplicado al origen de los valores x para lograr que los valores sean positivos. En el caso de Uruguay, este valor es 500.000.
- **Falso Norte:** Es un valor lineal aplicado al origen de los valores y. En el caso de Uruguay, su valor es cero porque la grilla parte del polo Sur.
- **Factor de Escala:** Es un valor usualmente menor que 1 que convierte una proyección tangente en una secante. Si el sistema de referencia no tiene un valor de escala, se entiende que éste es igual a 1. Se utiliza este parámetro para disminuir las distorsiones generales de la proyección. En el caso de Uruguay vale 1.

2.13.3.4.2 Parámetros angulares

- **Azimut:** Dirección medida en grados respecto del Norte. Define la línea central de la proyección.
- **Meridiano central o longitud de origen:** Define el origen de los valores x. En el caso de Uruguay, vale $55^{\circ}48' 0.0''$ W.
- **Paralelo central o latitud de origen:** Define el origen de los valores y. En el caso de Uruguay, $90^{\circ}0' 0.0''$ S, el Polo Sur.



2.14 WGS84

El WGS84 es un sistema de coordenadas cartográficas mundial que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas. WGS84 son las siglas en inglés de World Geodetic System 84 (que significa Sistema Geodésico Mundial 1984).

WGS84 también se refiere a la superficie de referencia esferoidal estándar (Datum o elipsoide de referencia) para datos de altitud y a la superficie equipotencial gravitacional (el geoide) que define el nivel del mar nominal. Se estima un error de cálculo menor a 2cm por lo que el Sistema de Posicionamiento Global (GPS) utiliza este sistema como sistema de coordenadas de referencia.

Previo a la existencia de WGS84 se utilizaba WGS72. A principios de los años 80 surgió la necesidad de un nuevo sistema geodésico mundial, ya que WGS72 no proveía suficientes datos, ni cobertura geográfica ni precisión para todas las aplicaciones del momento y para aquellas aplicaciones que se anticipaban. Se disponía de nueva información proveniente de la altimetría de radar satelital y de otras tecnologías, además de que se disponía de nuevos métodos matemáticos.

Como se mencionó antes, WGS84 es el sistema de referencia de GPS, por lo que GoogleMaps trabaja con coordenadas en este sistema. Es por este motivo que será necesario utilizar WGS84 en este proyecto.



2.15 Coordenadas geodésicas vs Coordenadas geocéntricas

La misma posición en un esferoide tiene un ángulo distinto para la latitud dependiendo de si el ángulo es medido desde la normal al esferoide (ángulo α) o alrededor del centro del esferoide (ángulo β).

El ángulo α se denomina latitud geodésica, mientras que el ángulo β se denomina latitud geocéntrica.

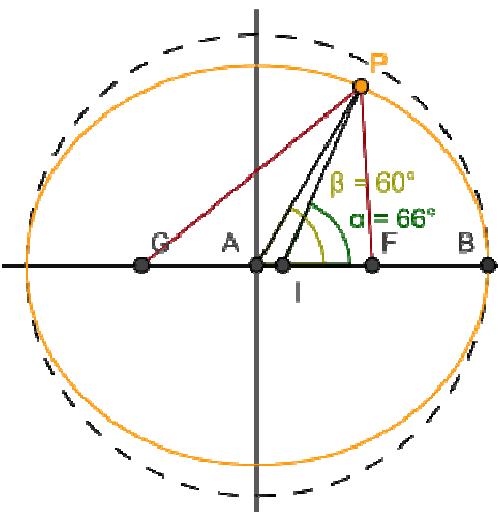


Figura 2.35 - Coordenadas geodésicas y geocéntricas

En coordenadas geodésicas, la superficie de la Tierra es aproximada por un elipsoide y las ubicaciones cerca de la superficie se describen en términos de latitud (φ), longitud (λ) y altura (h). El elipsoide está totalmente parametrizado por el semi-eje mayor a y el achatamiento f .



2.16 Manejo de dispositivos en Linux

Antes de las versiones linux-2.3/linux-2.4 era necesario que existieran una multitud de archivos en el directorio /dev para todos los posibles dispositivos que podía ser necesario que el kernel manejara, la mayoría de los cuales nunca eran usados.

Esto tenía varios problemas: la forma de acceder a un periférico concreto no era siempre la misma, ya que dependía de qué otros dispositivos estuvieran conectados; el directorio /dev era enorme y difícil de manejar; los números *mayor* y *menor* que se asociaban a cada dispositivo se estaban acabando; los programas necesitaban poder detectar cuándo se había conectado o desconectado un dispositivo, y cuál era el nodo que se le había asociado en /dev.

Para evitar estos problemas surgió *devfs*, el cual es un sistema de archivos cuyo propósito es controlar los archivos de dispositivos que se hallan almacenados en el directorio /dev. *Devfs* permite crear los archivos de dispositivos dinámicamente cuando se carga el módulo correspondiente. Se crean únicamente aquellos archivos que corresponden a los dispositivos que existen en el sistema. Además, el autor del módulo puede controlar el nombre del archivo y los derechos de acceso a éste. También, se pueden crear los enlaces simbólicos y directorios para organizar los archivos, aunque ésta es la tarea del *devfsd*.

A partir de Linux-2.6, el mecanismo para poblar el directorio /dev de forma dinámica con los nodos de los dispositivos pasó a ser la combinación de *udev* con *sysfs* que solucionó algunos problemas que no eran solucionados por *devfs*.

2.16.1 Udev

Según Wikipedia: “**Udev** es el gestor de dispositivos que usa el kernel Linux en su versión 2.6. Su función es controlar los ficheros de dispositivo en /dev. Es el sucesor de *devfs* y de *hotplug*, lo que significa que maneja el directorio /dev y todas las acciones del espacio de usuario al agregar o quitar dispositivos, incluyendo la carga de firmwares.”²

² es.wikipedia.org. *udev* [en línea] <<http://es.wikipedia.org/wiki/Udev>> [citado en 30 de enero de 2010].



La principal mejora de *udev* respecto de *devfs* es que permite acceder a un dispositivo con un nombre siempre fijo.

Udev se ejecuta mediante un demonio: el proceso *udevd*, que detecta cuando se ha conectado o desconectado un dispositivo del sistema. Cuando pasa uno de estos eventos, *udev* obtiene información de contexto (subsistema del kernel, conector físico usado, nombre dado por el kernel, etc.) y también del propio dispositivo (número de serie, fabricante). Esta información la puede encontrar mediante los datos que hay en /sys, en donde está montado el sistema de ficheros sysfs, del que se encarga el kernel (versión 2.6 y posteriores).

Un conjunto de reglas (en /etc/udev/rules.d/) deciden qué acción hay que realizar a partir de los datos obtenidos. Lo que la regla hará será, probablemente, dar un nombre al dispositivo, crear el fichero de dispositivo apropiado, y ejecutar el programa que se haya configurado para que termine de hacer funcionar el dispositivo.

Las reglas pueden asociar un nombre fijo a un dispositivo, pero también pueden llamar a un programa externo que dé más información sobre el dispositivo; así se puede conseguir un nombre más específico.



2.17 Gestión de paquetes en Linux

Apt es la sigla correspondiente a Advanced Packaging Tool (Herramienta avanzada de empaquetamiento). Apt es una interfaz de usuario, libre que trabaja con las bibliotecas del kernel para gestionar paquetes. Esta interfaz fue creada por el proyecto Debian y simplifica en gran medida el proceso de gestión de software, automatizando la obtención, configuración, instalación y eliminación de programas en los sistemas GNU/Linux.

Apt es una biblioteca de funciones C++ que es empleada por varios programas de línea de comandos para distribuir paquetes, por ejemplo apt-get.

Existe un repositorio central con más de 21.000 paquetes apt utilizados por apt-get y programas derivados para descargar e instalar aplicaciones directamente desde Internet, conocida como una de las mejores cualidades de Debian.

2.17.1 Apt-get

Como se mencionó en la sección anterior, apt-get es un programa de línea de comandos que hace uso de la biblioteca Apt. En este proyecto se utilizó apt-get para instalar muchas de los programas necesarios.

Existe un archivo asociado a apt-get, /etc/apt/sources.list, en el cual se almacenan las direcciones desde las cuales apt descargará la información necesaria para actualizar o instalar un programa cuando el comando *apt-get* sea invocado.

Apt-get ofrece comandos que permiten actualizar la lista de paquetes (*apt-get update*), actualizar o instalar algún paquete en particular (*apt-get install <paquete>*), actualizar la distribución del sistema operativo (*apt-get dist-upgrade*), etc.

Una función a destacar es que apt-get se encarga de obtener e instalar automáticamente aquellos paquetes requeridos por el paquete que se desea instalar.



2.18 GPS Daemon^{xviii}

Inicialmente se comenzó a desarrollar código para parsear las sentencias NMEA enviadas por el GPS. Sin embargo, al descubrir la existencia de GPS Daemon (GPSD), se decidió utilizar la información del GPS procesada por dicho programa.

GPSD es un demonio que monitorea uno o más receptores GPS conectados a una computadora a través del puerto USB o serial, haciendo que todos los datos de posición/cursor/velocidad de los receptores esté disponible para ser consultada en el puerto TCP 2947 de la computadora.

GPSD responde a las consultas en un formato que es sustancialmente más sencillo de parsear que el formato NMEA 0183 emitido por la mayoría de los GPSes.

Cada consulta por lo general consiste en un único carácter ASCII seguido de un carácter de fin de línea. Cada consulta devuelve una línea de texto de respuesta que termina con CR/LF (retorno de carro y salto de línea).

A efectos de este proyecto se utilizarán los comandos “p” y “d”. El comando “p” solicita la posición actual y la respuesta de GPSD a esta consulta tiene el formato: "GPSD, P=%f", en donde el primer campo representa la latitud y el segundo, la longitud. El comando “d” solicita la fecha y hora actuales y la respuesta de GPSD a esta solicitud tiene el formato: "GPSD, D=AAAA-MM-DDThh:mm:ss", en donde **A** representa las cifras correspondientes al año, **M** las cifras correspondientes al mes, **D** las cifras correspondientes al día, **h** las cifras correspondientes a la hora, **m** las cifras correspondientes a los minutos y **s** las cifras correspondientes a los segundos.



2.19 *Sockets^{xix}*

Para acceder a la información que GPSD coloca en el puerto TCP 2947 se utilizaron sockets. Los *sockets* son mecanismos de comunicación entre procesos que permiten que un proceso interactúe con otro proceso, incluso estando éstos en distintas computadoras.

La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar archivos. Debido a esta característica, se pueden realizar redirecciones de los archivos de E/S estándar (descriptores 0,1 y 2) a los sockets y así combinar aplicaciones de la red.

La comunicación entre procesos a través de sockets se basa en la filosofía cliente-servidor. En esta comunicación, un proceso actuará de proceso servidor creando un socket cuyo nombre conocerá el proceso cliente, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho socket nombrado. Sobre el descriptor del socket creado, es posible leer o escribir, lo cual permite que la comunicación sea bidireccional.

Los pasos para establecer una comunicación a través de sockets son los siguientes:

1. El proceso servidor crea un socket con nombre y espera la conexión.
2. El proceso cliente crea un socket sin nombre.
3. El proceso cliente realiza una petición de conexión al socket servidor.
4. El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.

Todo socket viene definido por dos características fundamentales:

- El **tipo** del socket, que indica la naturaleza del mismo, es decir, el tipo de comunicación que puede generarse entre los sockets.
- El **dominio** del socket, que especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.



2.19.1 Tipos de sockets

El tipo del socket define las propiedades de las comunicaciones en las que se ve envuelto un socket, esto es, el tipo de comunicación que se puede dar entre cliente y servidor.

Estas propiedades pueden ser:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- Comunicación orientada a conexión o no orientada a conexión.
- Envío de mensajes urgentes.

Los tipos disponibles son los siguientes:

- **SOCK_DGRAM**: Sockets para comunicaciones en modo no orientado a conexión, con envío de datagramas de tamaño limitado.
- **SOCK_STREAM**: Para comunicaciones fiables en modo orientado a conexión, de dos vías y con tamaño variable de los mensajes de datos.
- **SOCK_RAW**: permite el acceso a protocolos de más bajo nivel como el IP.
- **SOCK_SEQPACKET**: tiene las características del SOCK_STREAM pero, además, el tamaño de los mensajes es fijo.

2.19.2 Dominios

El dominio de un socket indica el formato de las direcciones que podrán tomar los sockets y los protocolos que soportarán dichos sockets.

Los dominios disponibles son los siguientes:

- **AF_UNIX** (Address Family UNIX): El cliente y el servidor deben estar en la misma máquina.



- **AF_INET** (Address Family INET): El cliente y el servidor pueden estar en cualquier máquina de la red Internet.
- **AF_NS**: Servidor y cliente deben estar en una red XEROX.
- **AF_CCITT**: Para protocolos CCITT, protocolos X25, etc.



2.20 Lenguaje C^{xx, xxi, xxii}

2.20.1 Introducción

El lenguaje de programación C es un lenguaje de uso general, cuya sintaxis es sumamente compacta y de alta portabilidad. Es un lenguaje de nivel intermedio, lo cual significa que trabaja a nivel de los elementos básicos presentes en todas las computadoras: caracteres, números y direcciones. Pretende ser un lenguaje de alto nivel, pero con la versatilidad del bajo nivel.

Además, el lenguaje C tiene otras particularidades, tales como que no posee operaciones de entrada salida, no maneja arreglos de caracteres ni maneja la asignación de memoria. Estas operaciones son realizadas por medio de llamadas a funciones contenidas en bibliotecas externas al lenguaje en sí. Esto es lo que hace que el lenguaje C sea altamente portable, ya que es independiente del hardware sobre el que corren los programas.

El lenguaje C se diseñó junto con el sistema operativo Unix y está muy orientado a trabajar en su entorno.

Las características anteriores fueron las que llevaron a decidir utilizar este lenguaje para el desarrollo de la mayoría de los módulos del sistema que habrían de ejecutarse en la placa BB.

Si bien se puede profundizar en muchas de las características del lenguaje C, a continuación se resumen los temas en los que fue necesario poner foco para desarrollar en este lenguaje.

2.20.2 Manejo de arreglos

Los arreglos ó conjuntos de datos ordenados (arrays) recolectan variables del mismo tipo, guardándolas en forma secuencial en la memoria. La cantidad máxima de variables que se pueden almacenar está sólo limitada por la cantidad de memoria disponible. El tipo de las variables involucradas puede ser cualquiera de los tipos de datos existentes en C, con la única restricción de que todos los componentes de un arreglo deben ser del mismo tipo.



En C los arreglos son de tamaño fijo, es decir, su tamaño se especifica en tiempo de compilación. No se comprueba la longitud del arreglo a la hora de acceder, por lo que es responsabilidad del programador controlar los accesos fuera de rango. Si se produjera un acceso fuera de rango, no se generará ningún error (ni siquiera en tiempo de ejecución). Esto puede provocar que se altere el contenido de otra variable distinta a aquella con la que se está trabajando, que se modifique parte del programa, o peor aún, se puede destruir parte del sistema operativo.

La forma de declarar arreglos en el lenguaje C es la siguiente:

cualificador tipo variable[expresión][expresión]... ={valor, ...};

Por ejemplo: `extern int vector[] = {0, 1, 2, 3};`

La expresión y los valores deben ser constantes. Si se declara la parte de valores se puede eliminar la expresión. En este caso el tamaño del vector es el mismo que el número de valores que se especifican.

La forma de hacer referencia a un elemento del vector es:

nombre[expresión][expresión]...

Si un arreglo se declara como global (afuera del cuerpo de todas las funciones), cada uno de sus elementos será automáticamente inicializado a cero. En cambio, si su declaración es local a una función, no se realiza ninguna inicialización, quedando a cargo del programa cargar los valores de inicio.

2.20.3 Asignación dinámica de memoria

La asignación dinámica de memoria es una técnica en la cual los programas determinan dónde almacenar la información mientras se están ejecutando. Esta técnica es necesaria cuando la cantidad de bloques de memoria necesarios depende de los datos sobre los que se está trabajando. En el caso de este proyecto, al hacer una consulta en la base de datos, la cantidad de resultados no se conoce previamente a la ejecución de la consulta, por lo que fue necesario asignar la memoria de forma dinámica.



La herramienta más general de asignación dinámica es la función `malloc()`, la cual permite asignar bloques de memoria de cualquier tamaño en cualquier momento, hacerlos más grandes o más chicos en cualquier momento, y liberar los bloques individualmente en cualquier momento (o nunca).

El encabezado de dicha función es el siguiente: `void * malloc (size_t size)`. La función devuelve un puntero al bloque de memoria que se acaba de asignar y que tiene `size` bytes de largo o devuelve un puntero nulo si el bloque no se pudo asignar. El contenido de los bloques no está definido, es necesario que el programador los inicialice.



2.21 **GNU Compiler Collection (GCC)**^{xxiii}

GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran, el cual puede recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de máquina correspondiente al sistema en donde se ha de ejecutar.

2.21.1 Etapas de compilación

El proceso de compilación involucra cuatro etapas sucesivas: pre-procesamiento, compilación, ensamblado y enlazado. Para pasar de un programa fuente escrito por un usuario a un archivo ejecutable es necesario pasar por estas cuatro etapas en forma sucesiva. El comando `gcc` es capaz de realizar todo el proceso de una sola vez.

2.21.1.1 Pre-procesamiento

En esta etapa se interpretan las directivas al preprocesador. Entre otras cosas, las variables inicializadas con `#define` son sustituidas en el código por su valor en todos los lugares donde aparece su nombre. Llamando al comando `gcc` con la opción `-E` es posible ejecutar solamente esta etapa.

2.21.1.2 Compilación

En la etapa de compilación se transforma el código C en el lenguaje ensamblador propio del procesador de nuestra máquina. Para ejecutar solo estas dos primeras etapas se puede utilizar la opción `-S` del comando `gcc`.

2.21.1.3 Ensamblado

El proceso de ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto, un archivo binario en lenguaje de máquina ejecutable por el procesador. El ensamblador se denomina `as`.

Es posible obtener un archivo en código objeto a partir de un archivo en lenguaje ensamblador invocando directamente al ensamblador `as`. También es posible realizar las tres primeras etapas pasándole la opción `-c` al comando `gcc`.



En los programas extensos, donde se escriben muchos archivos fuente en código C, es muy frecuente usar gcc con la opción -c para compilar cada archivo fuente por separado, y luego enlazar todos los módulos objeto creados. Estas operaciones se automatizan colocándolas en un archivo llamado makefile, interpretable por el comando make, quien se ocupa de realizar las actualizaciones mínimas necesarias toda vez que se modifica alguna porción de código en cualquiera de los archivos fuente.

2.21.1.4 Enlazado

Las funciones de C/C++ incluidas en el código, se encuentran ya compiladas y ensambladas en bibliotecas existentes en el sistema. Es preciso incorporar de algún modo el código binario de estas funciones al ejecutable con el que se desee trabajar. En esto consiste la etapa de enlace, donde se reúnen uno o más módulos en código objeto con el código existente en las bibliotecas.

El enlazador se denomina *ld*. El uso directo del enlazador *ld* es muy poco frecuente. En su lugar suele proveerse a gcc los códigos objeto directamente, es decir, suele invocarse gcc pasándole como argumento el código objeto.

2.21.2 Proceso completo

En programas con un único archivo fuente todo el proceso anterior puede hacerse en un solo paso, pasándole como argumento al comando gcc el archivo que contiene el código fuente.

2.21.3 Otras opciones de compilación

Existen otras opciones con las que se puede invocar el comando gcc muy útiles a la hora de compilar. Algunas de ellas son las siguientes:

- **-v (verbose)** Permite obtener un informe detallado de todos los pasos de compilación. Muestra los comandos ejecutados en cada etapa de compilación y la versión del compilador.
- **-o archivo (output)** Permite especificar el archivo de salida. Si no se especifica el archivo de salida, por defecto se crea un ejecutable denominado a.out.



- **-I ruta (include)** Permite especificar la ruta hacia el directorio donde se encuentran los archivos marcados para incluir en el programa fuente.
- **-L ruta (library)** Permite especificar la ruta hacia el directorio donde se encuentran los archivos de biblioteca con el código objeto de las funciones referenciadas en el programa fuente.



2.22 Construcción de bibliotecas estáticas^{xxiv, xxv, xxvi}

Para agrupar todas las funciones asociadas a un tema en particular suele utilizarse una biblioteca estática. Además, el uso de bibliotecas estáticas permite compartir código entre más de un programa de forma sencilla. En el caso de este proyecto, se utilizó una biblioteca estática para agrupar todas las funciones asociadas al acceso a la base de datos y todas las funciones asociadas a la obtención de información de posición.

2.22.1 Creación de archivos

Para construir una biblioteca estática es necesario crear uno o más archivos fuente (con extensión .c) y uno o más archivos de cabecera (con extensión .h). Los archivos fuente deben contener el código fuente de las funciones, mientras que los archivos de cabecera deben contener los tipos (typedefs, structs y enums) y prototipos de las funciones que se desea que se puedan utilizar desde otros programas.

Un detalle importante a tener en cuenta es que al construir una biblioteca no se sabe en qué futuros programas va a ser utilizada ni cómo estos estarán organizados. Puede ocurrir que un programa utilice el macro *include* para incluir dos archivos de cabecera y que estos, a su vez, incluyan una cierta biblioteca. Al compilar el programa surgirá un error debido a que todo lo que se define en la biblioteca se estaría definiendo dos veces.

Para evitar esto se suele utilizar un bloque *#ifndef - #endif*, dentro del cual se realizan todas las definiciones. A continuación se muestra un ejemplo esquemático:

```
#ifndef _BIBLIO_1_H
#define _BIBLIO_1_H
#define ...
#define ...
#endif
```

Dentro del bloque *#ifndef - #endif* se define el nombre utilizado para el bloque y luego se hacen todas las otras definiciones necesarias. De esta forma, la primera vez que se incluya esta biblioteca _BIBLIO_1_H no estará definido por lo que se entrará en el bloque *#ifndef - #endif* y se definirán todos los tipos y prototipos. Cuando la biblioteca sea incluida por segunda vez, _BIBLIO_1_H ya estará definido por lo que no se entrará en el bloque *#ifndef - #endif* y no se volverán a definir todos los tipos y prototipos.



2.22.2 Compilación y enlace

Una vez que se crearon los archivos necesarios, es necesario obtener los archivos objeto de todos los archivos fuente y crear la biblioteca a partir de los mismos.

Para obtener los archivos objeto (con extensión .o) de los archivos fuente es necesario compilar los archivos fuente con la opción –c, la cual le especifica al compilador que no cree un ejecutable, sino solo un fichero objeto. Por ejemplo:

```
$ gcc -c -o prueba.o prueba.c
```

Para crear la biblioteca (con extensión .a) se debe invocar al comando *ar* con los parámetros que se desee y pasándole los archivos objeto que se desea incluir en la biblioteca. El nombre de la biblioteca debe comenzar con *lib*.

En realidad *ar* es un comando mucho más genérico que permite empaquetar cualquier tipo de archivo (no solo archivos objeto). A continuación se muestra un ejemplo de una invocación al comando *ar*:

```
$ ar rcs libprueba.a prueba1.o prueba2.o
```

En este caso se utilizaron las opciones más comúnmente utilizadas al crear bibliotecas (r, c y s). Estas opciones especifican que se debe reemplazar el archivo de destino si éste ya existía (r), crear el archivo de destino si no existía (c) y construir un índice del contenido (s), lo cual hace que el enlace posterior con el programa principal sea más rápido.

2.22.3 Utilización de la biblioteca

Para utilizar las variables y funciones contenidas en una biblioteca estática es necesario enlazar la biblioteca al crear un ejecutable. Para ello, se debe indicar al compilador qué bibliotecas se desean utilizar y el lugar donde éstas se encuentran.



Existen tres opciones respecto de la ubicación de las bibliotecas:

1. La biblioteca creada es copiada en el directorio de bibliotecas del compilador, y su archivo de cabecera en el directorio de archivos de cabecera.
2. Ambos archivos son copiados en el directorio de trabajo del programa que va a hacer uso de la biblioteca. Esta opción impide el enlace y reutilización desde otros programas.
3. Los archivos se encuentran en un directorio cualquiera.

Para la opción 1 se puede obtener el ejecutable del programa que utilizará la biblioteca simplemente invocando al compilador con la opción `-o` para especificarle el nombre del archivo de salida. Por ejemplo, al compilar el programa `apli1.c` se ejecuta el siguiente comando:

```
$ gcc -o apli1 apli1.c
```

Para las opciones 2 y 3, al momento de obtener el archivo objeto del programa que va a utilizar la biblioteca se le debe indicar al compilador la ubicación de los archivos de cabecera mediante la opción `-I` (i mayúscula). Por ejemplo, para obtener el archivo objeto del programa `apli2.c` se invoca el siguiente comando:

```
$ gcc -c -o apli2.o apli2.c -Idir_lib
```

Donde `dir_lib` es la ruta del directorio donde están la biblioteca y los archivos de cabecera necesarios.

Luego, se debe enlazar el programa con la biblioteca indicando dónde se encuentra la misma (con la opción `-L`) y su nombre (con la opción `-l` (l minúscula)). Por ejemplo:

```
$ gcc -o apli2 apli2.o -Ldir_lib -lfich
```

Nota: Al utilizar la opción `-l` (l minúscula) debe indicarse el nombre de la biblioteca que se desea incluir sin escribir el prefijo `/lib` ni la extensión `.a`.

Por ejemplo:

```
$ gcc -o apli2 apli2.o -L/home -lGPS
```

Al compilar `apli2` utilizando el comando anterior, se está especificando que se incluya la biblioteca `libGPS.a` que se encuentra en el directorio `/home`.



2.23 API C de MySQL^{xxvii}

MySQL ofrece una API (Application Programming Interface) que permite establecer una conexión con un servidor MySQL desde un programa escrito en lenguaje C y de esta forma realizar transacciones sobre bases de datos alojadas en dicho servidor.

En esta sección se describirán los tipos de datos y las funciones de esta API que fueron utilizadas para cumplir con algunos de los objetivos del proyecto.

2.23.1 Tipos de datos

- `MYSQL`

Esta estructura representa un puntero a una conexión con una base de datos. Es utilizada por casi todas las funciones MySQL.

- `MYSQL_RES`

Esta estructura representa el resultado de una consulta que devuelve filas (SELECT, SHOW, DESCRIBE, EXPLAIN). La información devuelta por una consulta se denomina conjunto de resultados (*result set*).

- `MYSQL_ROW`

Esta es una representación de tipo seguro de una fila de datos. Actualmente está implementada como un arreglo de cadenas de bytes contadas.

2.23.2 Funciones

2.23.2.1 `mysql_init()`

La función `MYSQL *mysql_init(MYSQL *mysql)` permite inicializar o asignarle memoria a un objeto `MYSQL` adecuado para invocar la función `mysql_real_connect()`, la cual se explicará más adelante.

Si el parámetro `mysql` es un puntero a `NULL`, la función le asigna memoria, inicializa y devuelve un nuevo objeto. De lo contrario, se inicializa el objeto que se pasó como parámetro y se devuelve la dirección del mismo.



Si `mysql_init()` le asigna memoria a un nuevo objeto, ésta es liberada cuando se invoque la función `mysql_close()` para cerrar la conexión.

2.23.2.2 `mysql_real_connect()`

La función `MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)` intenta establecer una conexión con un motor de bases de datos MySQL ejecutándose en `host`. Es necesario que esta función se complete con éxito antes de poder ejecutar cualquier otra función de la API que requiera un puntero a una conexión `MYSQL` válida.

A continuación se especifican los parámetros que recibe esta función:

- `mysql` debe ser la dirección a una estructura `MYSQL` existente.
- `host` puede ser el nombre o bien la dirección IP del servidor con el que se desea establecer la conexión. En Unix, el cliente se conecta utilizando un socket.
- `user` debe contener el nombre de usuario del usuario MySQL con el que se iniciará sesión.
- `passwd` debe contener el password correspondiente al usuario especificado en `user`.
- `db` es el nombre de la base de datos a la que se desea acceder.
- Si `port` no es 0, se utiliza su valor como el número del puerto para la conexión TCP/IP.
- Si `unix_socket` no es NULL, especifica el socket que debe ser utilizado.
- El valor de `client_flags` por lo general es 0, pero se puede setear como una combinación de ciertos flags para habilitar algunas funciones especiales.

2.23.2.3 `mysql_real_query()`

La función `int mysql_real_query(MYSQL *mysql, const char *stmt_str, unsigned long length)` permite ejecutar la sentencia SQL a la que apunta el parámetro `stmt_str`, el cual debería ser una cadena de caracteres de `length` bytes de largo.

También existe la función `mysql_query()`, sin embargo, ésta no puede ser usada para sentencias que contengan datos binarios. En ese caso debe usarse `mysql_real_query()`.



Además, *mysql_real_query()* es más rápida que *mysql_query()* debido a que no llama a la función *strlen()* para conocer el tamaño de la sentencia.

Esta función devuelve cero si la sentencia fue exitosa y un valor distinto de cero si ocurrió algún error.

2.23.2.4 mysql_store_result()

La función *MYSQL_RES *mysql_store_result(MYSQL *mysql)* debe ser invocada luego de invocar *mysql_query()* o *mysql_real_query()* para cada sentencia que produzca con éxito un conjunto de resultados (SELECT, SHOW, DESCRIBE, EXPLAIN, CHECK TABLE, etc.). También se debe invocar *mysql_free_result()* cuando se haya terminado de trabajar con el conjunto de resultados. Esta función se explicará más adelante.

Esta función lee el resultado completo de una consulta, asigna memoria a una estructura *MYSQL_RES* y ubica el resultado en esta estructura. Si la sentencia no devolvió un conjunto de resultados (por ejemplo, si era una sentencia INSERT) o si la lectura del conjunto de resultados falló, la función devuelve un puntero nulo.

2.23.2.5 mysql_num_rows()

La función *my_ulonglong mysql_num_rows(MYSQL_RES *result)* devuelve la cantidad de filas en un conjunto de resultados. Esta función está pensada para ser usada con sentencias que devuelven un conjunto de resultados, como SELECT.

2.23.2.6 mysql_affected_rows()

La función *my_ulonglong mysql_affected_rows(MYSQL *mysql)* debe ser invocada luego de ejecutar una sentencia utilizando *mysql_query()* o *mysql_real_query()* y devuelve la cantidad de filas que fueron cambiadas (para UPDATE), borradas (para DELETE) o insertadas (para INSERT). Para sentencias SELECT, *mysql_affected_rows()* funciona como *mysql_num_rows()*.

Si esta función devuelve un entero mayor que cero, indica la cantidad de filas afectadas u obtenidas. Si devuelve cero, indica que no se actualizó ningún registro para una sentencia UPDATE, que ninguna fila coincidió con la cláusula WHERE en la consulta o



que aún no se ha ejecutado ninguna consulta. -1 indica que la consulta devolvió un error o que, para una sentencia SELECT, *mysql_affected_rows()* fue invocada antes de llamar a la función *mysql_store_result()*.

2.23.2.7 mysql_fetch_row()

La función *MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)* trae la próxima fila de un conjunto de resultados. Cuando es usada luego de *mysql_store_result()*, esta función devuelve NULL si no hay más filas que traer o si ocurrió un error.

La cantidad de valores en la fila está dado por la función *mysql_num_fields(result)*. Si se denomina **row** a la variable que contiene el valor de retorno de una llamada a *mysql_fetch_row()*, los punteros a los valores se acceden como **row[0]** hasta **row[mysql_num_fields(result)-1]**. Los valores nulos en la fila se indican como punteros a NULL.

2.23.2.8 mysql_free_result()

La función *void mysql_free_result(MYSQL_RES *result)* libera la memoria que fue asignada a un conjunto de resultados utilizando la función *mysql_store_result()*, *mysql_use_result()*, *mysql_list_db()*, etc.

2.23.2.9 mysql_close()

La función *void mysql_close(MYSQL *mysql)* cierra una conexión que fue abierta previamente. Esta función, además, desasigna la memoria asignada al puntero *mysql*, si ésta fue asignada automáticamente por la función *mysql_init()* o por la función *mysql_connect()*.



2.24 Lenguajes para desarrollo Web

Para el desarrollo de la interfaz Web de este proyecto se optó por utilizar el lenguaje HTML conjuntamente con algunas instrucciones específicas de los lenguajes PHP y JavaScript. Por este motivo, a continuación se describen de forma resumida los mencionados lenguajes.

2.24.1 HTML^{xxviii, xxix}

Para publicar información y que ésta pueda ser distribuida de forma global, es necesario un lenguaje que sea entendido universalmente por cualquier computadora. El lenguaje de publicación utilizado en Internet es HTML (HyperText Markup Language).

HTML es una aplicación del SGML (Standard Generalized Markup Language), el cual es un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar esos mismos documentos. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas. HTML puede incluir un *script* (por ejemplo, JavaScript), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

El entorno para trabajar con HTML es simplemente un procesador de texto. El conjunto de etiquetas que se crean, se deben guardar con la extensión .htm o .html. Estos documentos pueden ser mostrados por los visores o navegadores Web, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer.

2.24.2 PHP^{xxx}

PHP es un lenguaje de scripting de propósito general y de código abierto, ampliamente utilizado, el cual es especialmente adecuado para desarrollo Web y puede estar embebido en HTML.

PHP es un acrónimo recursivo de “*PHP: Hypertext Preprocessor*” (PHP: Preprocesador de Hipertexto).



En lugar de utilizar muchos comandos HTML para mostrar datos, las páginas PHP contienen HTML con código PHP embebido que hace “algo”. El código PHP se encierra entre instrucciones especiales de comienzo y fin de procesamiento (`<?php` y `?>`) que permiten entrar y salir del “modo PHP”.

Lo que distingue a PHP de JavaScript es que el código se ejecuta en el servidor, generando código HTML, el cual luego es enviado al cliente. El cliente recibirá los resultados de ejecutar ese script, pero no sabrá cuál era el código subyacente.

2.24.3 Java Script^{xxx}

JavaScript es un lenguaje de tipo script compacto, orientado a objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. Este lenguaje suele utilizarse para habilitar el acceso programático a objetos tanto desde la aplicación cliente como desde otras aplicaciones.

Los programas JavaScript van embebidos en los documentos HMTL, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos, etc.



2.25 Cargador de arranque

Un cargador de arranque es un programa sencillo que no tiene la totalidad de las funcionalidades de un sistema operativo, diseñado exclusivamente para preparar todo lo que necesita el sistema operativo para funcionar. Normalmente se utilizan los cargadores de arranque multietapas, en los que varios programas pequeños se suman los unos a los otros, hasta que el último de ellos carga el sistema operativo.

En los ordenadores modernos, el proceso de arranque comienza con la CPU ejecutando los programas contenidos en una dirección predefinida de la memoria ROM y se configura la CPU para ejecutar este programa, sin ayuda externa, al encender el ordenador.

2.25.1 Tipos de cargadores de arranque

2.25.1.1 Gestor de arranque de segunda etapa

Este programa contiene funcionalidades rudimentarias que le permiten buscar unidades que se puedan seleccionar para participar en el arranque, y cargar un pequeño programa desde una sección especial de la unidad más prometedora. El pequeño programa no es, en sí mismo, un sistema operativo sino, simplemente, un cargador de arranque de segundo nivel, como Lilo o Grub, que es capaz de cargar el sistema operativo propiamente dicho y, finalmente, transferirle el control.

El proceso de arranque se considera completo cuando el ordenador está preparado para contestar a los requerimientos del exterior.

2.25.1.2 Gestor de arranque Flash

Un gestor de arranque Flash reside en la memoria Flash, y siempre es la primera aplicación que se ejecuta después de un reinicio. El gestor de arranque Flash decide si una aplicación está lista y, por tanto, o bien se queda en la memoria principal del sistema o salta a la solicitud para iniciar la ejecución.



2.25.1.3 Gestor de arranque de red

La mayoría de los ordenadores también son capaces de arrancar en una red informática. En este escenario, el sistema operativo se almacena en el disco duro de un servidor, y ciertas partes del mismo se transfieren al cliente mediante un simple protocolo, como TFTP (Trivial File Transfer Protocol).

2.25.1.4 Otros tipos de secuencia de arranque

Algunos procesadores tienen otros tipos de modos de arranque; la mayoría de los procesadores de señal digital tienen los siguientes modos de arranque:

- Modo de arranque de serie
- Modo paralelo de arranque
- HPI boot

2.25.2 Cargador de arranque en la placa BeagleBoard

2.25.2.1 uBoot

uBoot es un cargador de arranque para diversas arquitecturas de procesadores, como son: PPC, ARM, AVR32, MIPS, X86, etc. Licenciado bajo la licencia GPL, actualmente uBoot es el cargador de arranque más utilizado en sistemas Linux embebidos. uBoot es el cargador de arranque que se utiliza en la placa BB.

A continuación se mencionan las características teóricas de la placa BB que es necesario tener en cuenta a la hora de compilar manualmente uBoot (teniendo en cuenta los cambios que se desean realizar).

2.25.2.2 Secuencia de arranque

La placa BB tiene una secuencia de arranque predeterminada que se ejecuta siempre que se enciende, a no ser que se interrumpa dicha secuencia mediante la presión de los botones de *User* o *Reset*. Esto significa que la placa BB buscará por defecto cargadores de arranque en los dispositivos en el siguiente orden:

1. NAND
2. UART3
3. FLASH
4. MMC1



En caso de no encontrar un cargador de arranque válido en ninguno de dichos dispositivos aparecerá el mensaje “40T” en la consola de la conexión serial. La solución a este problema se explica detalladamente en el documento BeagleBoard Recovery referenciado en la Bibliografía de este documento.

2.25.2.3 Puerto de expansión

Una de las ventajas que posee el procesador OMAP 3530 es que permite configurar, dentro de determinados límites, de qué señales se desea disponer en cada pin. Esto se debe a que dicho procesador cuenta con un multiplexor de salida que permite elegir entre 4 opciones básicas. A continuación se presenta una tabla que indica las señales de las que se dispone según cada opción configurada.

Pin	Option A	Option B	Option C	Option D
1		VIO_1V8		
2		DC_5V		
3	MMC2_DAT7	GPIO_139		
4	McBSP3_DX	GPIO_140	UART2_CTS	
5	MMC2_DAT6	GPIO_138		
6	McBSP3_CLKX	GPIO_142	UART2_TX	
7	MMC2_DAT5	GPIO_137		
8	McBSP3_FSX	GPIO_143	UART2_RX	
9	MMC2_DAT4	GPIO_136		
10	McBSP3_DR	GPIO_141	UART2_RTS	
11	MMC2_DAT3	McSPI3_CS0	GPIO_135	
12	McBSP1_DX	McSPI4_SIMO	McBSP3_DX	GPIO_158
13	MMC2_DAT2	McSPI3_CS1	GPIO_134	
14	McBSP1_CLKX	McBSP3_CLKX	GPIO_162	
15	MMC2_DAT1	GPIO_133		
16	McBSP1_FSX	McSPI4_CS0	McBSP3_FSX	GPIO_161
17	MMC2_DAT0	McSPI3_SOMI	GPIO_132	
18	McBSP1_DR	McSPI4_SOMI	McBSP3_DR	GPIO_159
19	MMC2_CMD	McSPI3_SIMO	GPIO_131	
20	McBSP1_CLKR	McSPI4_CLK	SIM_CD	GPIO_156
21	MMC2_CLKO	McSPI3_CLK	GPIO_130	
22	McBSP1_FSR			GPIO_157
23	I2C2_SDA	GPIO_183		
24	I2C2_SCL	GPIO_168		
25		REGEN		
26		nRESET		
27		GND		
28		GND		

Tabla 2-6 - Señales puerto de expansión



3 Marco Metodológico del Proyecto Final de Carrera

3.1 Introducción a la metodología

Las tareas asociadas a este proyecto se pueden dividir en tareas de gestión y tareas técnicas. Dentro de las tareas de gestión están incluidas las tareas de planificación, control de avance y redacción de documentos. Dentro de las tareas técnicas están incluidas las tareas de investigación teórica, implementación y testeo.

3.2 Gestión del proyecto

3.2.1 Planificación

Para planificar el desarrollo del proyecto se creó un cronograma utilizando la aplicación Microsoft Project. En este cronograma se realizó un desglose de las tareas en sub-tareas y se asignó la responsabilidad de las mismas a los integrantes del equipo de trabajo. Además, la aplicación genera un diagrama de Gantt, el cual permite visualizar de forma clara el desarrollo del proyecto.

Para la creación del cronograma fue necesario estimar los tiempos que habría que dedicarle a cada tarea, teniendo como referencia el trabajo realizado en los proyectos de las asignaturas cursadas durante la carrera. Además, al decidir la duración de cada tarea se agregó un margen de seguridad teniendo en cuenta que podían surgir imprevistos.

Para la asignación de tareas se tuvo en cuenta la modularidad del proyecto, la cual permitió asignar las tareas de forma de que cada integrante pudiera avanzar con las tareas bajo su responsabilidad, en paralelo al avance del otro integrante. Es decir, que el avance de uno de los integrantes del equipo condicionaba lo mínimo posible el avance del otro integrante.

Es importante tener en cuenta que en algunos casos no fue posible lograr esta independencia, ya que algunas tareas requerían de la participación en simultáneo de ambos integrantes. Por ejemplo, aquellas tareas que requerían definir interfaces que afectan a ambas partes o bien aquellas tareas que requerían tomar decisiones de alcance global.



3.2.2 Control de avance

Mientras se cursaba la Asignatura Proyecto de Telemática se realizaron controles de avance semanales. En estos controles se verificaba que el proyecto se estuviera desarrollando conforme a lo previsto en el cronograma y se asentaban los avances en actas, las cuales eran firmadas tanto por los tutores del proyecto como por los integrantes del equipo.

En el Anexo II – Actas se pueden encontrar las actas generadas.

Para poder representar los avances de forma gráfica, se utilizó una funcionalidad de la aplicación Microsoft Project que permite establecer el porcentaje de completitud de cada tarea. Esto facilita la toma de decisiones respecto de la prioridad de las tareas a realizar.

3.2.3 Redacción de documentos

Además del presente informe, se redactó un documento de Alcance y un documento denominado Plan de Proyecto.

3.2.3.1 Documento de Alcance

En el documento de Alcance se realizó una introducción al proyecto, comentando sobre la oportunidad de negocio detectada y describiendo el producto que se pretendía desarrollar. Además, se especificaron las funcionalidades que serían incluidas y aquellas que no. También se especificaron los entregables del proyecto y se anexó el cronograma inicial en el que se podían observar las distintas fechas de entrega de los mismos, además de las fechas de realización de las distintas tareas.

3.2.3.2 Plan de Proyecto

En el Plan de proyecto se incluyó el Acta del Proyecto, la cual está compuesta por los siguientes elementos:

- Nombre del proyecto
- Fecha de comienzo
- Áreas de conocimientos/procesos



- Áreas de aplicación (sector/actividad)
- Fecha de inicio del Proyecto
- Fecha tentativa de finalización del Proyecto

En este documento también se incluyeron los objetivos del proyecto, general y específicos; una descripción del producto; la necesidad del proyecto; el alcance (detallando las funcionalidades incluidas y las no incluidas); la justificación del impacto del proyecto; los entregables del proyecto, las restricciones del proyecto y se identificaron los grupos de interés (directos e indirectos) del proyecto.

En el “Anexo I – Plan del Proyecto Final de Carrera” se adjunta el plan de proyecto.

3.3 Tareas técnicas

3.3.1 Investigación

Como primer sub-tarea de la mayoría de las tareas previstas en el cronograma, se planificó la investigación sobre todos los temas teóricos relacionados con esa tarea.

En todos los casos se realizaron investigaciones de tipo secundario, buscando en Internet, en manuales, en tutoriales, en foros y en distintos libros, creando la base teórica necesaria para poder proceder a la implementación.

Toda la información recolectada durante las etapas de investigación fue respaldada en un repositorio Subversion para luego ser usada en la redacción del marco teórico de este informe.

3.3.2 Implementación y testeo

Para los módulos asociados al PIC se dividió la implementación y el testeo en dos etapas: primero se implementó y testeó en una simulación (utilizando el programa Proteus) y luego se armaron los circuitos físicos y se testearon en el laboratorio. Esta última etapa, a su vez se dividió en dos: primero se armaron los circuitos sobre ProjectBoards y luego de verificado su correcto funcionamiento, se soldaron sobre placas de pertinax.



La simulación en la aplicación Proteus permitió depurar el programa del PIC y permitió corregir algunos detalles de los circuitos diseñados.

De todas formas, fue necesario seguir realizando modificaciones luego de armados los circuitos físicos. Esto se debe a que en la práctica existen diversos factores que afectan el funcionamiento de estos circuitos, como por ejemplo el ruido eléctrico, que no son tenidos en cuenta en la simulación.



4 Desarrollo del proyecto

En este capítulo se describe el desarrollo de las tareas que se realizaron para poder cumplir con los objetivos del proyecto. Se describen los procedimientos que se llevaron a cabo, las dificultades encontradas y cómo éstas fueron solucionadas. Además, se mencionan las decisiones que fue necesario tomar y los motivos que llevaron proceder de la forma en que se procedió.

A continuación se presenta un esquema modular del sistema, el cual permite visualizar los módulos que se desarrollaron.

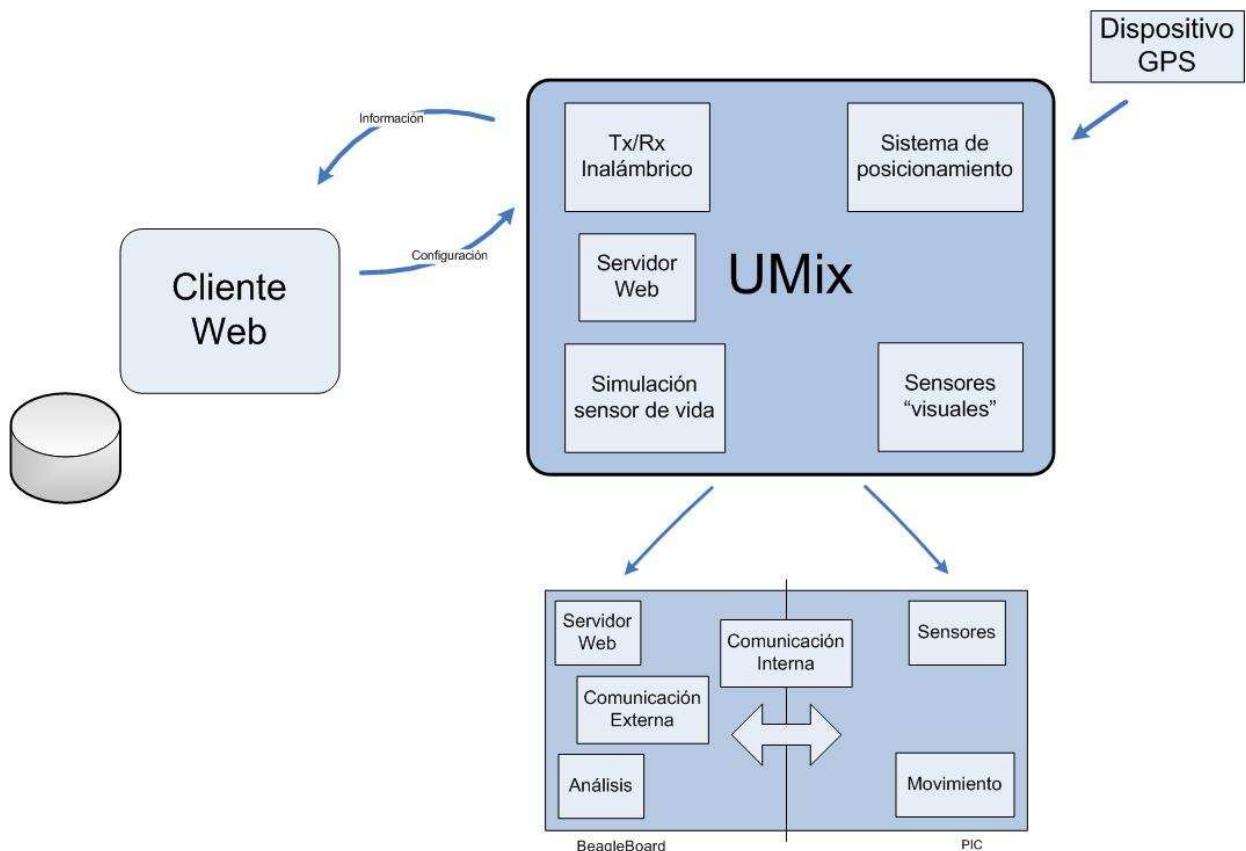


Figura 4.1 - Esquema modular



4.1 Tareas iniciales

4.1.1 Definición del tema

La primera tarea de este proyecto consistió en definir el tema del mismo. Se partió de la idea de aprovechar la placa de desarrollo BeagleBoard (BB) utilizada por otra alumna de la facultad en su proyecto de final de carrera. Por otro lado, era necesario utilizar algún dispositivo para implementar la interfaz entre la placa BB y los motores y sensores. Se decidió que este dispositivo fuera un microcontrolador PIC, debido a que ya se había trabajado con este tipo de microcontroladores en asignaturas anteriores de la carrera. Por otro lado, se planteó la idea de integrar al proyecto el uso de un sistema GPS.

Integrando todas estas ideas, el concepto del producto final de este proyecto empezó a tomar forma. Luego se buscó una aplicación que motivara al proyecto y que fuera adecuada a estos tiempos y a la situación presente.

De esta forma se llegó al tema del proyecto: “Desarrollo de un sistema de búsqueda de supervivientes de catástrofes”.

4.1.2 Estudio de la placa

Se leyeron los manuales y documentación disponibles para familiarizarse con la placa BB y su funcionamiento. Además, se buscaron proyectos de desarrollo basados en esta misma placa, analizando los informes u otros documentos de dichos proyectos.

4.1.3 Definición del formato de documentos

Para que hubiera una coherencia entre todos los documentos, se creó una plantilla como template. Esta plantilla establece el formato de la carátula, la información de historial y de referencias que debe incluirse, el formato de los títulos y del texto, etc. La misma se almacenó en el repositorio para que ambos integrantes del equipo tuvieran acceso a él.

4.1.4 Redacción del plan de proyecto

Esta tarea ya fue explicada en el capítulo 3.



4.2 Compilación e instalación de ARM EABI Ubuntu 9.04

4.2.1 Fundamento de la elección del sistema operativo

Una de las decisiones más importantes que hay que tomar al momento de realizar proyectos de estas características es el sistema operativo a utilizar.

En principio es posible realizar el desarrollo de todo el código del sistema operativo y compilarlo, sin embargo, esta opción demanda mucho tiempo y recursos, teniendo en cuenta las funcionalidades que se le pretende dar a esta aplicación en particular. Principalmente el desarrollo de todos los protocolos de red e implementación de servidores requeriría de un tiempo que, en este caso, era inadmisible.

La segunda opción es utilizar alguno de los sistemas operativos que se encuentran en el medio y adaptarlo para que funcione en la plataforma de hardware establecida, como por ejemplo Windows ó algunas de las distribuciones de Linux.

La amplia documentación, soporte y escalabilidad del kernel de Linux, hacen que cualquier distribución del mismo tenga grandes ventajas con respecto a cualquier otro tipo de sistema operativo, cuando se trata de sistemas embebidos.

Es por estas razones que se optó por utilizar el kernel de Linux para implementar UMix y, en particular, se eligió la distribución Ubuntu por varios motivos:

1. Ubuntu se basa en el desarrollo de la distribución Debian, que es, por su parte, ARM compliant. De esta forma, al trabajar con un sistema operativo avalado por el propio fabricante del procesador que tiene la placa en cuestión, se minimizan los riesgos de no funcionamiento.
2. Por otro lado, en el caso de Ubuntu, es más sencillo encontrar material de apoyo para esta distribución debido a la gran inserción que ha tenido en el mercado de las computadoras personales en los últimos años.
3. En último lugar, pero no menos importante, el equipo de trabajo del proyecto tiene más experiencia trabajando con dicha distribución que con cualquier otra. De esta forma se pueden aprovechar los tiempos que llevaría habituarse a otras distribuciones de Linux para realizar otras actividades más críticas.



4.2.2 Conexión Serial

Previo a la instalación de Ubuntu en la BB es necesario establecer una conexión serial con la placa BB. A continuación se detalla el procedimiento para establecer dicha conexión.

En primer lugar se deben conseguir los cables y conectores necesarios. Es decir, un cable **Null-modem**, también conocido como cable **serial cruzado**, un “Ribbon cable” y un puerto en la computadora huésped que permita conexiones seriales (puede ser un **puerto serial PCI** o un **adaptador USB-serial**).

La Figura 4.2 muestra cómo se conectaron los cables.



Figura 4.2 - Conexión Serial

El diagrama de pines de cada uno de los cables puede ser encontrado en el manual de referencia de la placa BB.

Para comprobar el funcionamiento de la comunicación serial se puede configurar una conexión utilizando el programa HyperTerminal. El procedimiento para establecer esta



conexión se explica en el Anexo III. Si la comunicación serial funciona correctamente, debería verse una pantalla similar a la que se muestra a continuación.

The screenshot shows a window titled "bb - HyperTerminal". The menu bar includes "Archivo", "Edición", "Ver", "Llamar", "Transferir", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main window displays the following text:

```
Texas Instruments X-Loader 1.41
Starting OS Bootloader...

U-Boot 1.3.3 (Jul 10 2008 - 16:33:09)
OMAP3530-GP rev 2, CPU-OPP2 L3-165MHz
OMAP3 Beagle Board + LPDDR/NAND
DRAM: 128 MB
NAND: 256 MiB
In: serial
Out: serial
Err: serial
Audio Tone on Speakers ... complete
```

Figura 4.3 - Conexión establecida

4.2.3 Actualización de uBoot

La BB posee en su memoria un gestor de arranque, llamado **uBoot**, que es conveniente actualizar antes de comenzar la instalación del sistema operativo. Las instrucciones sobre cómo realizar esta operación se encuentran claramente detalladas en el manual de usuario de la placa BB y por eso no serán tratadas en este documento.

4.2.4 Compilación del kernel (Método 1)

Actualmente existen dos formas para obtener una versión ejecutable del kernel de Ubuntu para la placa BB. La primera forma, la cual se describe en ésta sección, consiste en obtener el código pre-compilado para el hardware que utilizaremos. La segunda forma, la cual se describe en la siguiente sección, consiste en obtener el código fuente del kernel y luego compilarlo manualmente.

En el caso de este proyecto, y debido a que no se tenía previsto hacer modificaciones en el kernel se optó por utilizar la primera forma.



4.2.4.1 Adaptación de la PC huésped

Para comenzar la instalación de Ubuntu con un kernel pre-compilado es necesario primero adaptar una determinada PC, que denominaremos huésped, en donde se adaptará el kernel seleccionado y se desarrollarán las aplicaciones que luego se ejecutarán en la placa BB.

- **Ubuntu Jaunty**

Debido a la gran disponibilidad de compiladores y el gran soporte que ofrece este sistema operativo se optó por utilizar el mismo para la PC huésped. Además, es conveniente, a la hora de desarrollar aplicaciones embebidas, que la PC huésped tenga la mayor similitud con la plataforma en donde residirá definitivamente la aplicación.

La instalación de este sistema operativo es intuitiva e incluso se puede realizar directamente desde el live-cd de la distribución. No se incluye en este manual una descripción de la instalación de dicho sistema operativo.

- **Qemu**

Una vez instalado el sistema operativo de la PC huésped es necesario instalar alguna herramienta que permita simular la plataforma de la placa BB, de forma de poder emular el comportamiento de cualquier aplicación en la PC huésped como si se estuviera directamente trabajando sobre la placa. Para ello existe una aplicación llamada Qemu.

Esta herramienta se encuentra dentro de las aplicaciones disponibles en el repositorio de Ubuntu y por lo tanto su instalación consiste simplemente en la ejecución de los siguientes comandos desde una terminal:

```
$sudo apt-get install qemu
```

Cabe aclarar que para instalar cualquier aplicación desde los repositorios es necesario disponer de una conexión a Internet.

- **Debootstrap**

Otra de las herramientas que facilitan la instalación del sistema operativo Ubuntu es la herramienta denominada **debootstrap**. La función de esta aplicación es simplificar todo el proceso de instalación de cualquier sistema operativo basado en Debian, sobre todo a la hora de manejar el particionamiento de las unidades de memoria de la plataforma objetivo.



Para instalar la última versión de la herramienta se deben ejecutar los siguientes comandos en una terminal:

```
$wget http://ports.ubuntu.com/pool/main/d/debootstrap/debootstrap_1.0.13~jaunty1_all.deb  
$sudo dpkg -i debootstrap_1.0.13~jaunty1_all.deb
```

- **Rootstock**

Finalmente existe un script que automatiza todo el proceso de compilación e instalación de Ubuntu llamado rootstock. Cabe destacar que este script depende fuertemente de las dos aplicaciones antes instaladas. Su obtención se realiza mediante los siguientes comandos:

```
$wget http://launchpad.net/project-rootstock/trunk/0.1/+download/rootstock-0.1.3.tar.gz  
$tar xf rootstock-0.1.3.tar.gz  
$cd rootstock-0.1.3
```

- **Uboot-mkimage**

Para adaptar la imagen del kernel al cargador de arranque uBoot es necesario realizar determinadas operaciones con el fin de dicha imagen quede correctamente instalada en la partición destino. Para esto existe un programa llamado **uboot-mkimage** que automatiza dicho proceso de adaptación. Esta aplicación se encuentra dentro de los repositorios de Ubuntu y por lo tanto su instalación se realiza de la siguiente forma:

```
$sudo apt-get install uboot-mkimage
```

4.2.4.2 Generación de imágenes y kernel

Una vez que se dispone de las herramientas mencionadas anteriormente en la PC de desarrollo, es necesario utilizarlas para generar la imagen del kernel y el sistema de archivos que necesita la plataforma para funcionar adecuadamente. Para esto el comando en su forma más amplia sería el siguiente:

```
$sudo ./rootstock --fqdn <hostname> --login <rootuser> --password <rootuserpasswd> --imagesize  
<qemu image size> --seed <packages> --dist <jaunty/karmic>  
--serial <ttySx> --kernel-image <http>
```

En el caso de este proyecto el comando resultó como se muestra a continuación:

```
$sudo ./rootstock --fqdn beagleboard --login ubuntu --password temppwd --imagesize 2G --seed  
xfce4,gdm --dist jaunty  
--serial ttyS2 --kernel-image http://rcn-ee.net/deb/kernel/beagle/jaunty/v2.6.29-58cf2f1-  
oer44.1/linux-image-2.6.29-oer44.1_1.0jaunty_armel.deb
```

De dicho comando cabe destacar que, en primer lugar, el usuario y la contraseña establecidos fueron “ubuntu” y “temppwd” respectivamente. En segundo lugar, la imagen del sistema de archivos debe ser de 2 GB. Y por último, los paquetes agregados a la



instalación común fueron los paquetes xfce4 y gdm aunque se podrían haber considerado otros como son los paquetes para los escritorios gráficos, adaptadores de red, etc.

Al final de la ejecución del comando, en caso de ser exitosa, se debería disponer de los archivos **armel-rootfs-<date>.tgz** y **vmlinuz-2.6.<version>**

4.2.5 Compilación del kernel (Método 2)

Si bien el procedimiento descripto en la sección anterior ofrece sus comodidades debido a la rapidez y facilidad con que se realiza, también tiene el inconveniente de que no es siempre aplicable y que oculta gran parte de las operaciones que se realizan impidiendo de esta forma aprender sobre el funcionamiento de Linux. Es por esto que a continuación se describe un método más general de compilación e instalación.

4.2.5.1 Adaptación de la PC huésped

Antes de comenzar el proceso de compilación es necesario instalar determinadas herramientas:

1. Compilador

Como ya se mencionó anteriormente, Linux es un kernel implementado casi en su totalidad en el lenguaje C, con algunas instrucciones directamente escritas en lenguaje ensamblador. Por lo tanto, para compilar el kernel es necesario disponer de un compilador de C que genere un código que pueda ser directamente interpretado por el procesador de la plataforma objetivo.

El compilador de C más conocido y utilizado actualmente es **gcc**, que es mantenido y desarrollado por la comunidad GNU. Usualmente gcc viene pre-instalado en todas las distribuciones de Linux, pero si se desea obtenerlo simplemente hay que dirigirse a la página Web del proyecto <http://gcc.gnu.org>, descargarlo y compilarlo.

2. Linker

Debido a la complejidad del proceso de compilación de aplicaciones hechas en C, el compilador en si no es capaz de realizar esta tarea por sí solo. Es por esto que necesita de otro grupo de herramientas, denominadas **binutils**, para hacer el enlace (link) y armado de todo el código fuente.



Este grupo de herramientas viene empaquetado en un paquete que lleva el mismo nombre y puede ser descargado de <http://www.gnu.org/software/binutils>, aunque usualmente cada distribución ya dispone de una versión pre-compilada de este paquete.

3. Make

Make es una herramienta que recorre el árbol de archivos de código fuente del kernel, decide cuáles de ellos es necesario compilar e invoca luego al compilador y a otras herramientas.

En general todas las distribuciones vienen con una versión de make ya incorporada, pero si se desea el código fuente, el mismo está disponible en la Web <http://www.gnu.org/software/make> y por lo tanto puede ser descargado y compilado.

Cabe destacar que las herramientas citadas son aquellas que son ABSOLUTAMENTE indispensables para la compilación. Dependiendo del sistema objetivo es posible que sea necesario incorporar otras herramientas.

4.2.5.2 Selección de la rama del kernel

Una vez completada la instalación de las herramientas es necesario escoger qué versión del kernel se va a instalar. Actualmente existen varias ramas en desarrollo, cada cual con sus características distintivas. La Figura 4.4 ilustra el árbol correspondiente a los núcleos disponibles y en desarrollo.

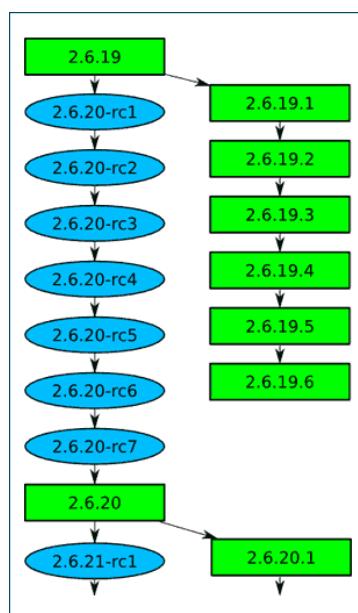


Figura 4.4 - Núcleos disponibles



4.2.5.3 Obtención del código fuente

El código fuente de todo el árbol de núcleos está disponible en la Web <http://www.kernel.org>. Para descargar el código de una rama en particular basta con hacer clic en la ‘F’ correspondiente a dicha rama.

Concluido el paso anterior, se debería tener un archivo comprimido que contiene todos los archivos de código fuente. Por lo cual, para terminar con la obtención del código se debe descomprimir dicho archivo a algún directorio independiente.

4.2.5.4 Crear y modificar una configuración

En esta etapa cada desarrollador debe elegir los módulos y características que desea obtener del kernel, como son, por ejemplo, los drivers a ser incluidos en el mismo. Toda la configuración del núcleo elegida se guarda en el archivo “**.config**” que se encuentra en la raíz de la estructura de directorios del kernel.

Para crear dicho archivo existen dos métodos:

- **Configuración desde cero**

Este método consiste en ir seleccionando una a una si se desea incluir determinada característica ó no. Para realizar esto es necesario posicionarse en el directorio raíz del código fuente y ejecutar:

```
$make config
```

- **Configuración por defecto**

Cada rama del kernel cuenta con una configuración por defecto que incluye las aplicaciones que el encargado de esa rama considera idóneas para ejecutar ese kernel. Por este motivo, es suficiente ejecutar el siguiente comando para generar un archivo de configuración con dichas opciones:

```
$make defconfig
```

Una vez generado un archivo de configuración es posible realizarle modificaciones. Para ello existen varias herramientas como son:

- **Menuconfig**

Esta herramienta se trata de una aplicación que funciona desde una consola. Para iniciarla es necesario ejecutar el siguiente comando:

```
$make menuconfig
```



- **Gconfig**

En este caso se trata de una herramienta que se ejecuta en entorno grafico basada en GTK+.

- **Xconfig**

Otra aplicación grafica pero basada en QT.

No se detalla en este documento cómo se utiliza cada una de las herramientas.

4.2.5.5 Compilación del kernel

Una vez establecido el archivo de configuración, lo único que resta hacer es compilarlo. Para eso se ejecuta el comando **make**, seguido, en este caso, de una opción para establecer la arquitectura del procesador correspondiente a la placa BB.

```
$make ARCH=arm CROSS_COMPILE=<ruta a toolchain de ARM>
```

En donde <ruta a toolchain de ARM> indica la ruta hacia el directorio en donde se haya configurado el compilador de C para ARM.

4.2.6 Particionado y armado de la SD

4.2.6.1 Particionado con gparted

La placa BB cuenta, como dispositivo de memoria, con un lector de memorias SD que es adecuado para la instalación del sistema operativo. En el caso de este proyecto, se decidió utilizar una tarjeta SD de 2 GB por considerarla suficiente para cumplir con los requerimientos de memoria previstos y por ser de relativo bajo costo.

De cualquier forma, independientemente del tamaño de la tarjeta de memoria, es necesario particionar dicha tarjeta para que se pueda utilizar para la instalación del sistema operativo.

Para esto se utilizó la herramienta **gparted**, cuyas últimas versiones funcionan como un sistema operativo independiente, contenido dentro de un live CD. Dicho software está licenciado bajo la licencia GPL y, por lo tanto, puede ser descargado y utilizado libremente.



Lo importante de la tabla de particiones que se debe generar con gparted es lo siguiente:

- Que exista una partición primaria del tipo **FAT32 booteable** al principio de la memoria de la tarjeta de aproximadamente 50 MB.
- Que el resto de la unidad de memoria sea llenado con una partición **primaria del tipo ext3/ext2**.

Para poder hacer uso de las funcionalidades que ofrece gparted es necesario colocar el live CD en la compactera y reiniciar la PC para que ésta arranque desde dicho CD.

A continuación se explica el proceso de creación de las particiones con los requerimientos recién referidos:

1. Seleccionar la partición FAT32 existente.



Figura 4.5 – Partición FAT32 existente

2. Haciendo clic con el botón derecho del mouse, seleccionar la opción **Delete**, para eliminar la partición FAT32 existente.

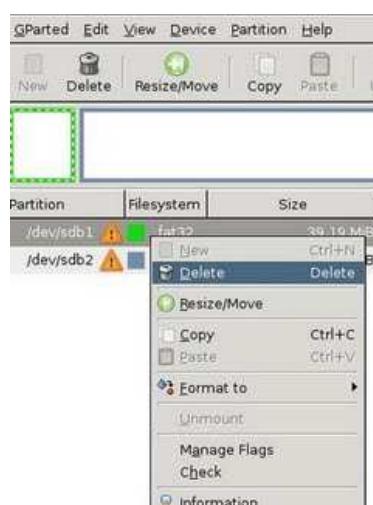


Figura 4.6 – Eliminar partición FAT32 existente



- Crear una nueva partición primaria de tipo FAT32.

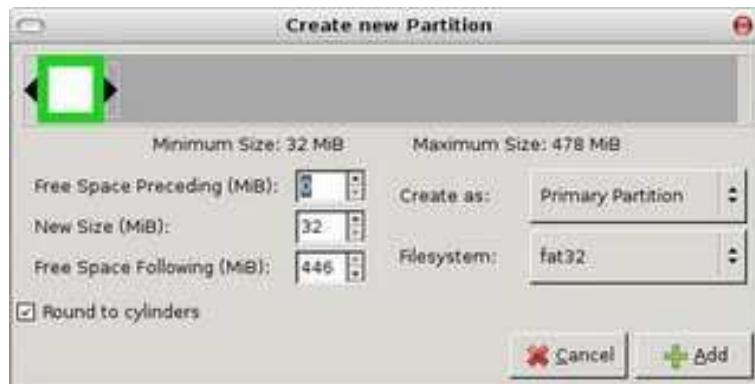


Figura 4.7 – Creación de nueva partición FAT32

- Crear una nueva partición primaria de tipo ext3.

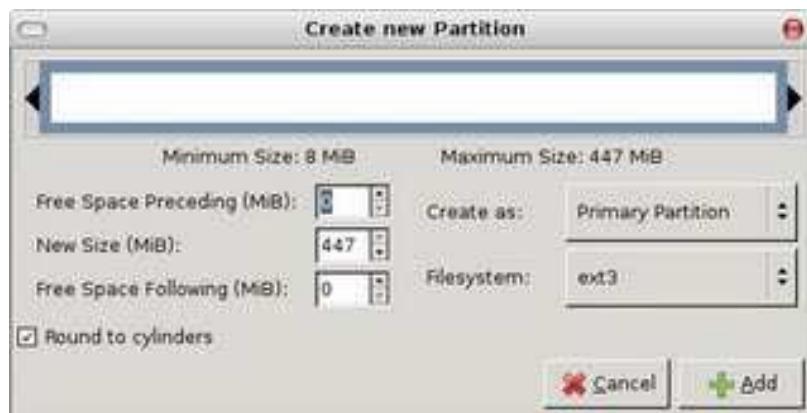


Figura 4.8 - Creación de nueva partición ext3

- Hacer clic derecho sobre la nueva partición FAT32 y seleccionar la opción **Manage Flags**.

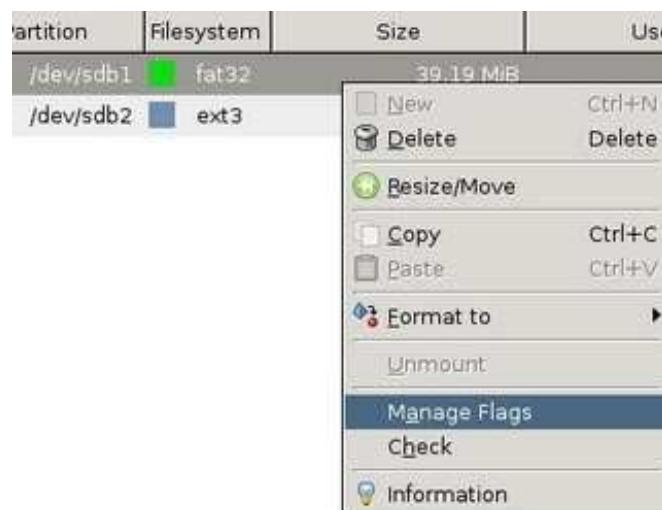


Figura 4.9 – Opción Manage flags



6. Chequear la casilla correspondiente a la opción **boot** para que la partición FAT32 creada sea booteable.



Figura 4.10 – Partición booteable

4.2.6.2 Instalación de uBoot/ulmage en la partición de la SD

Terminado el particionado de la tarjeta de memoria hay que instalar la imagen del kernel dentro de la partición FAT32 creada. Para esto hay que montar dicha partición en algún directorio (en el caso de este proyecto: /media/disk) y luego ejecutar los comandos que se muestran a continuación.

Dentro de la carpeta donde se crearon las imágenes del kernel:

```
$mkimage -A arm -O linux -T kernel -C none -a 0x80008000 -e 0x80008000 -n "Linux" -d ./vmlinuz-  
* ./ulmage
```

Luego:

```
$cd /media/disk  
$sudo cp ./<directorio de las imágenes>/ulmage ulmage
```

4.2.6.3 Instalación de la partición ext2/ext3 de la SD

En la partición ext2/ext3 que se generó en el paso anterior, es necesario copiar el sistema de archivos que será montado como '/' cuando se inicie el sistema operativo en la placa BB. Dichos archivos se encuentran comprimidos dentro del archivo **armel-rootfs-<date>.tgz** que se obtuvo en la sección “Generación de imágenes y kernel”. Por lo tanto,



es necesario descomprimir dicho archivo y copiar su contenido manteniendo el sistema de permisos. Para esto hay que montar la partición ext2/ext3 en algún directorio (en el caso de este proyecto se utilizó /media/disk) y ejecutar el siguiente comando:

```
$sudo tar xfp armel-rootfs-[date].tgz -C /media/disk
```

Terminado el copiado de archivos a la SD, es conveniente modificar el archivo de las interfaces de red que posee la placa, para luego poder establecer conexiones entre la placa BB y una PC. Para esto se debe montar nuevamente la partición en caso de que haya sido desmontada y ejecutar el siguiente comando (nuevamente se utilizó el directorio /media/disk como punto de montaje):

```
$sudo gedit /media/disk/etc/network/interfaces
```

Este comando abrirá un editor de texto en donde se deben ingresar las siguientes líneas:

```
auto eth0
iface eth0 inet static
    address 192.168.1.254
    netmask 255.255.255.0
```

4.2.7 Configuración de uBoot

Para culminar la instalación de Ubuntu en la placa BB, se debe configurar uBoot para que pueda descargar adecuadamente la imagen del kernel a la memoria de la placa.

En este paso es necesario disponer de la conexión serial que se explicó en la sección “Conexión Serial” de este informe. Se debe encender la placa BB con la memoria SD insertada en el slot correspondiente e interrumpir la ejecución de uBoot presionando cualquier tecla.

Una vez interrumpida la ejecución de uBoot, OMAP ofrecerá una terminal en donde se deben ingresar los siguientes comandos:

```
setenv bootcmd 'mmcinit; fatload mmc 0:1 0x80300000 ulimage; bootm 0x80300000'
setenv bootargs 'console=ttyS2,115200n8 console=tty0 root=/dev/mmcblk0p2 rootwait
rootfstype=ext3 ro omapfb.mode=dvi:1280x720MR-16@60'
saveenv
boot
```

Estos comandos pueden variar según de qué revisión de la placa se disponga. De ser exitosa la instalación, debe aparecer en la pantalla toda la secuencia de mensajes de información del arranque de Linux.



4.3 Instalación de aplicaciones y actualización del kernel

4.3.1 Conexión de red cableada

El primer paso, luego de haber terminado la instalación del sistema operativo, es configurar una red cableada para disponer de una conexión de datos alternativa a la conexión serial. A partir de esta conexión se podrá luego controlar la placa BB desde cualquier PC independientemente de que se disponga o no de una conexión serial. Además facilitará toda la transferencia de archivos que sea necesaria para la instalación de las aplicaciones que más adelante se mencionarán.

En la sección “Instalación de la partición ext2/ext3 de la SD” se explicaron los pasos de configuración para disponer de una red Ethernet cableada dentro de la placa BB. Resta ahora configurar adecuadamente la PC host y armar la interconexión física de los equipos.

Debido a que la placa BB no cuenta con ningún puerto que permita directamente la entrada de un conector RJ-45, que es el más utilizado en las redes Ethernet cableadas, fue necesario utilizar un adaptador USB/Ethernet.

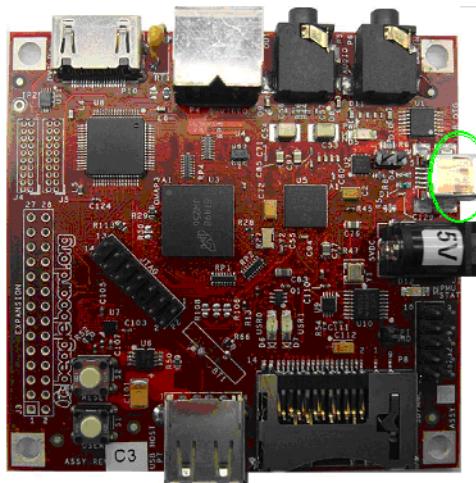


Figura 4.11 - Puerto USB

En primera instancia se podría pensar en conectar directamente un adaptador USB-Ethernet al puerto USB de la placa BB. Lamentablemente, la placa no cuenta con la energía suficiente como para alimentar un dispositivo USB. Es por esto que, para poder establecer una conexión de red, es indispensable contar con un HUB USB que disponga



de un puerto maestro. De esta forma, se conecta el adaptador USB-Ethernet a cualquiera de las entradas del HUB y se conecta la placa BB al puerto maestro del HUB como muestra la Figura 4.12.



Figura 4.12 - Conexión de red

Es necesario aclarar que no todos los conectores USB que se pueden introducir en el puerto USB de la placa sirven para que ésta funcione en modo maestro, que es como debe funcionar para establecer una conexión de red.

Para terminar de armar la plataforma física de la red solo resta conectar el adaptador USB-Ethernet con la PC host mediante un cable UTP.

La configuración de la PC host requiere únicamente que se le asigne una dirección IP que esté dentro del rango de direcciones IP de la red que se configuró para la placa, y que se le asigne la misma máscara.

Finalmente se debe encender la placa BB con el HUB USB y la red conectada (la versión de Ubuntu instalada en la placa no reconoce el adaptador USB a menos que éste esté conectado en la etapa de arranque).

Para comprobar que se ha configurado todo correctamente, se puede ejecutar un ping desde la PC host hacia la placa BB y verificar que responde a dicho comando.



4.3.2 Conexión cableada a Internet

Una vez que se dispone de una red cableada entre una PC y la placa se abre la posibilidad de conectar la placa BB directamente a Internet. Dicha conexión permitiría descargar e instalar los paquetes que sean necesarios en las secciones siguientes, directamente desde los servidores de repositorios de Ubuntu, sin necesidad de ningún intermediario.

A continuación se detallan los pasos para establecer una conexión a Internet, en una red donde existe un router como intermediario entre la red e Internet, mediante un servicio ADSL.

En base a la plataforma física de la red armada en la sección anterior, lo único que hay que modificar es que, en vez de conectar el adaptador USB-Ethernet a una PC, hay que conectarlo al router.

En cuanto a la configuración establecida para las interfaces de red de la placa BB, resta únicamente configurar el Gateway de la red y los servidores DNS (esto siempre y cuando la dirección IP y la máscara de la red establecidas anteriormente sean compatibles con las del router). Para esto hay que iniciar nuevamente una conexión serial para acceder a una terminal en la placa.

Cuando se disponga de la terminal, hay que comenzar estableciendo el Gateway de la red con el siguiente comando:

```
$sudo route add default gw <ip del router>
```

De esta forma se le indica al sistema operativo de la placa que cualquier paquete de red que desee enviar a una dirección IP fuera de su área lo debe enviar al router.

Los servidores DNS son necesarios para que el sistema operativo de la placa BB pueda asociar el nombre de los servidores de Internet con su dirección IP. En Ubuntu, la configuración de los servidores DNS se guarda en el archivo /etc/resolv.conf, por lo que la configuración de los mismos se resume al siguiente comando que debe ser ejecutado como root:

```
#echo nameserver <ip del servidor DNS> > /etc/resolv.conf
```



Para obtener la dirección IP del servidor DNS, cada usuario deberá consultar con su ISP (Internet Service Provider) particular.

De aquí en más se asumirá que la placa cuenta con una conexión directa a Internet para descargar los paquetes directamente de los servidores de repositorios. En caso de que no se disponga de esta conexión, se deberán descargar los paquetes de instalación (*.deb) manualmente, guardarlos en la memoria de la placa (mediante un servidor SSH o copiándolos en la partición ext3 de la tarjeta SD) y luego instalarlos ejecutando el siguiente comando:

```
$sudo dpkg -i nombredelpaquete.deb
```

4.3.3 Servidor SSH

Una aplicación que simplifica enormemente la ejecución de comandos y la manipulación y transferencia de archivos en la placa BB, es un servidor SSH. Mediante este servidor podremos disponer de una terminal dentro de la placa y transmitir archivos desde y hacia la placa BB desde cualquier PC que tenga una conexión de red y un cliente SSH. De esta forma, se evitan las limitaciones asociadas con contar únicamente con una conexión serial.

Para instalarlo, simplemente hay que establecer una conexión serial de la forma que se explicó anteriormente e iniciar sesión en la placa BB utilizando el usuario y la contraseña establecidos.

Una vez que se dispone de una terminal a través de la conexión serial, hay que ejecutar los siguientes comandos:

```
$sudo apt-get install openssh-server openssh-client
```

Desde la PC host se puede comprobar el correcto funcionamiento del servidor ejecutando los siguientes comandos a fin de obtener una terminal dentro del sistema operativo de la placa:

```
$ssh <usuario>@<ip de la placa>
```

Si todo funcionó correctamente, se solicitará el ingreso de la contraseña del usuario y, una vez comprobada la identidad del mismo, se le otorgará una terminal para ejecutar



comandos de la misma forma que si los estuviera escribiendo con un teclado conectado directamente a la placa.

4.3.4 Actualización del kernel

Antes de comenzar a trabajar con el sistema operativo, es conveniente actualizar el kernel a la última versión disponible, principalmente para disponer de los últimos drivers para los distintos componentes.

En Ubuntu existe un script que automatiza todo el proceso de obtener el último kernel compilado para ARM, generar las imágenes y actualizar la configuración de uBoot.

El primer paso a llevar a cabo es descargar este script en la PC host. Esto se realiza siguiendo el link <http://www.rcn-ee.net/deb/kernel/ubuntu-update-kernel.sh>

Una vez que se haya descargado el script, hay que pasarlo a la placa BB. Para esto se utilizará el servidor SSH instalado en la sección anterior. En Ubuntu, el comando para iniciar una transmisión mediante SSH es scp (secure copy), el cual se invoca como se muestra a continuación:

```
scp <path local del script> <usuario>@<IP de la BB>:<path remoto para el script>
```

Por último hay que abrir una terminal en la placa BB y ejecutar el comando como root. El sistema preguntará si se desea actualizar el link simbólico de vmlinuz, lo cual se debe aceptar. El script informará el resultado de la operación. Para que los cambios sean aplicados, se debe reiniciar el sistema operativo de la placa.

4.3.5 Servidor Apache

En el alcance del proyecto se especificó la necesidad de disponer de una interfaz web para administrar y manipular el móvil. Esto requiere que se instale alguna aplicación que permita alojar dicha interfaz en la placa BB.

En este proyecto se optó por implementar una interfaz en base al lenguaje PHP por los siguientes motivos:

- Es relativamente fácil de utilizar en comparación con los otros lenguajes.
- Tiene una gran capacidad para interactuar con el sistema operativo.



- No requiere demasiados recursos. En particular el consumo de energía adicional que requiere su utilización no es significativo.

Teniendo en cuenta que se iba a utilizar este lenguaje, se decidió utilizar el servidor Apache con su módulo para PHP para sustentar la interfaz.

Dicho servidor se encuentra en los repositorios conjuntamente con su conector para PHP. Es por esto que, para instalarlo, basta con ejecutar el siguiente comando en una terminal de la placa BB:

```
sudo apt-get install apache2 php5 libapache2-mod-php5 php5-gd php5-mysql
```

4.3.6 DNS Dinámico

Las características del proyecto UMix hacen que sea deseable conocer en todo momento la dirección IP que le ha sido asignada a la placa BB. De esta forma es posible acceder a la interfaz Web para administrar y manipular el móvil.

Pensando en no limitar el proyecto a un servicio de IP fija, se optó por incorporar al proyecto una aplicación que permitiera aprovechar los servicios que ofrecen los proveedores de DNS dinámicos.

En este proyecto se decidió utilizar los servicios de no-ip por ser una de las empresas líderes proveyendo este tipo de servicios. No se hará en este informe un seguimiento sobre cómo registrar una cuenta y un dominio en no-ip, simplemente se detallarán los pasos a seguir para instalar el cliente y configurarlo en Ubuntu.

El cliente de no-ip también se encuentra dentro de los repositorios de Ubuntu, bajo el nombre de noip2. Por lo tanto, su instalación consiste en ejecutar el siguiente comando en una terminal de la placa:

```
$sudo apt-get install noip2
```

Aparecerá una pantalla en la cual se piden los datos que se configuraron en la cuenta de no-ip. Finalizado esto, la aplicación hará todo el trabajo de actualización de la dirección IP automáticamente.



4.4 Conexión inalámbrica a Internet

La funcionalidad del proyecto requiere que el móvil pueda comunicarse mediante algún protocolo de red inalámbrico. En principio, casi cualquier protocolo puede ser utilizado por la placa. Sin embargo, por la gran disponibilidad y cobertura de las redes 3G y teniendo en cuenta que, en caso de no haber cobertura, se podría hacer uso de una radio base móvil para solucionar esta dificultad, se decidió hacer uso de esta tecnología a través de un modem 3G USB. Por otro lado, es necesario utilizar alguna aplicación que realice la conexión.

En principio se pensó utilizar **wvdial** ya que es la aplicación más difundida para los propósitos mencionados, sin embargo, al momento de su utilización se encontraron problemas debido a que determinadas llamadas de la API de Debian no fueron implementadas para la versión para ARM.

Es por esto que finalmente se optó por utilizar **pppdial**. Los comandos para su instalación son los siguientes:

```
$sudo apt-get install pppdial
```

La configuración de la aplicación es un tanto engorrosa, pero puede resumirse en los siguientes pasos:

1. Crear el archivo **/etc/chatscripts/claro**, con el siguiente contenido

```
#!/bin/sh
/usr/sbin/chat -v "" ATZ "" AT+CGDCONT=1,"IP","internet.ctimovil.com.uy" "" "ATD*99#"
CONNECT ""
```

Luego, aplicarle permisos de ejecución como root con el siguiente comando:

```
#chmod +x /etc/chatscripts/claro
```

2. Crear el archivo **/etc/ppp/peers/claro**, con el siguiente contenido:

```
connect /etc/chatscripts/claro
/dev/ttyUSB0
360000
noipdefault
crtsccts
modem
noccp
defaultroute
novj
noauth
ipcp-accept-remote
ipcp-accept-local
```



```
debug
idle 60
```

El parámetro **/dev/ttyUSB0** puede variar dependiendo del nodo creado por udev al detectar el modem 3G.

Luego se le ajustan los permisos con los siguientes comandos, ejecutados como root:

```
# chmod 0640 /etc/ppp/peers/claro
# chown root:dip /etc/ppp/peers/claro
```

3. Por último, es posible conectarse o desconectarse con los siguientes comandos:

```
$pon claro
$poff claro
```

4. En caso de que la conexión no funcione correctamente, es posible que haya que configurar adecuadamente las rutas y DNS. Para ello se debe consultar al proveedor de Internet y ejecutar los siguientes comandos:

Para cambiar el Gateway:

```
$sudo route add default gw <IP del Gateway>
```

Para cambiar el DNS:

```
#echo nameserver <IP del DNS> > /etc/resolv.conf
```



4.5 Diseño y construcción del móvil

En esta sección se describe la tarea de diseño y construcción del móvil, incluyendo también los circuitos anexos necesarios para controlar los motores.

Se decidió que el móvil utilizara para su desplazamiento dos motores: un motor de continua para la tracción (ruedas traseras) y un motor paso a paso (PaP) para controlar la dirección (ruedas delanteras). Para mantener bajos los gastos del proyecto se decidió reciclar todo lo que fuera posible. Con este propósito se utiliza el motor de continua de la bandeja de una compactera y el motor PaP bipolar que controla la posición de la cabeza lectora de un disco duro. Además, se reciclaron el chasis y las ruedas de autos de juguete.

4.5.1 Motor de continua

Para controlar el motor de continua se utiliza uno de los puentes del driver L293D. Se conectaron 3 pines del puerto B del PIC: RB0, RB1 y RB2 a los pines Enable1, In1 e In2 del L293D respectivamente. Los pines Out1 y Out2 del L293D se conectaron a los terminales del motor de continua. De esta forma, cuando RB0 está en 1, el puente 1 del L293D está habilitado. En caso contrario, dicho puente queda deshabilitado por lo que el motor queda libre. Estando el puente 1 del L293D habilitado: si RB1 = 1 y RB2 = 0 el motor gira en un sentido (hacia adelante), si RB1 = 0 y RB2 = 1 el motor gira en sentido contrario (hacia atrás) y si RB1 = RB2 el motor es detenido.

4.5.2 Motor PaP

Para identificar los terminales del motor PaP se utilizó un tester: entre dos de los cables del motor se midió la misma resistencia que entre los otros dos y al medir en forma cruzada no se detectó continuidad. Es decir que el primer par de cables está conectado a los bornes de una de las bobinas y el otro par de cables está conectado a los bornes de la otra bobina.

Como se mencionó en el capítulo 2, para controlar los motores PaP es necesario invertir las polaridades de los terminales de las bobinas en una determinada secuencia para



lograr un giro en un sentido y en secuencia opuesta para que el motor gire en el otro sentido. Con este propósito se utilizaron los dos puentes de un driver L293D.

Se conectó el pin RB3 del PIC a los pines Enable1 y Enable2 del L293D. Con el bit RB3 se habilitan/deshabilitan los puentes del L293D. Se conectó el pin RB4 del PIC al pin In1 del L293D, el pin RB5 del PIC al pin In2 del L293D, el pin RB6 del PIC al pin In3 del L293D y el pin RB7 del PIC al pin In4 del L293D. Se conectaron los pines Out1 y Out2 a los bornes de una de las bobinas y los pines Out3 y Out4 a los bornes de la otra bobina.

La cantidad de pasos que es necesario que el motor realice para que el móvil doble hacia la derecha o hacia la izquierda se determinó de forma práctica. Del mismo modo, se determinó a través de pruebas la duración necesaria de los pulsos de control de entrada al L293D, teniendo en cuenta las especificaciones de este integrado.

En la Figura 4.13 se muestra un esquema con las conexiones mencionadas anteriormente.

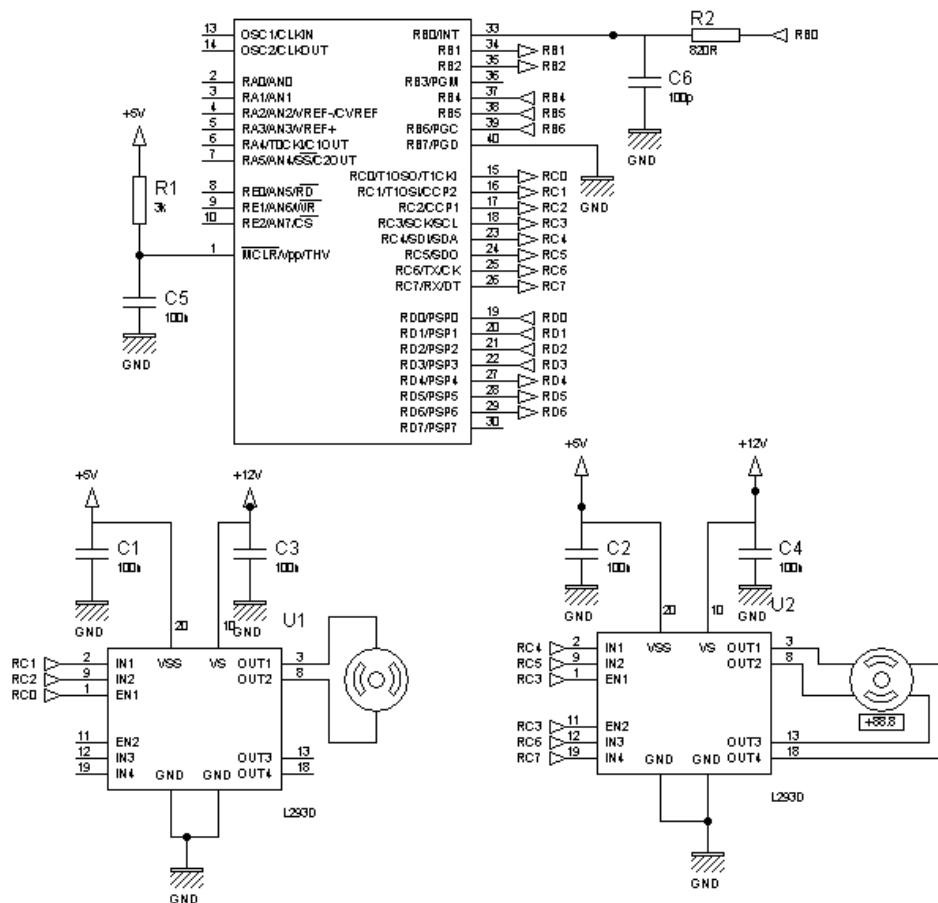


Figura 4.13 - Esquema PIC y drivers



4.6 Implementación del Módulo de Movimiento

4.6.1 Estudio del módulo PWM

En un principio se consideró que era necesario utilizar el módulo PWM del PIC para el control de los motores. Luego de estudiar el funcionamiento de dicho módulo y sus aplicaciones se llegó a la conclusión de que la principal aplicación del módulo PWM en el control de motores es el control de velocidad de motores de continua. En este caso no era necesario este control.

Por otro lado, se detectó la oportunidad de utilizar el módulo PWM para generar las señales de control que actuarían como entradas para el driver del motor PaP. Sin embargo, debido al desplazamiento de medio pulso requerido para estas señales de control, se consideró más adecuado generar los trenes de pulsos necesarios utilizando un bucle que escriba ceros y unos en cuatro bits de uno de los puertos del PIC. Para que estos bits se mantuvieran en el nivel correspondiente el tiempo que fuera necesario, se incluyeron llamadas a una rutina de espera entre cada par de instrucciones de escritura en el puerto.

En resumen, aunque se consideró al planificar el proyecto que sería necesario utilizar el módulo PWM del PIC para controlar los motores, llegado el momento se decidió que éste no sería utilizado.

4.6.2 Desarrollo del código

Para desarrollar el código del módulo de movimiento para el PIC se utilizó el lenguaje assembler. Fue fundamental para el desarrollo de este código la experiencia obtenida en asignaturas de la carrera tales como Sistemas Digitales y Laboratorio de Telemática, en las cuales se realizaron proyectos utilizando el microcontrolador PIC.

A continuación se presentan los diagramas de flujo de las rutinas que se desarrollaron para este módulo con sus correspondientes descripciones.

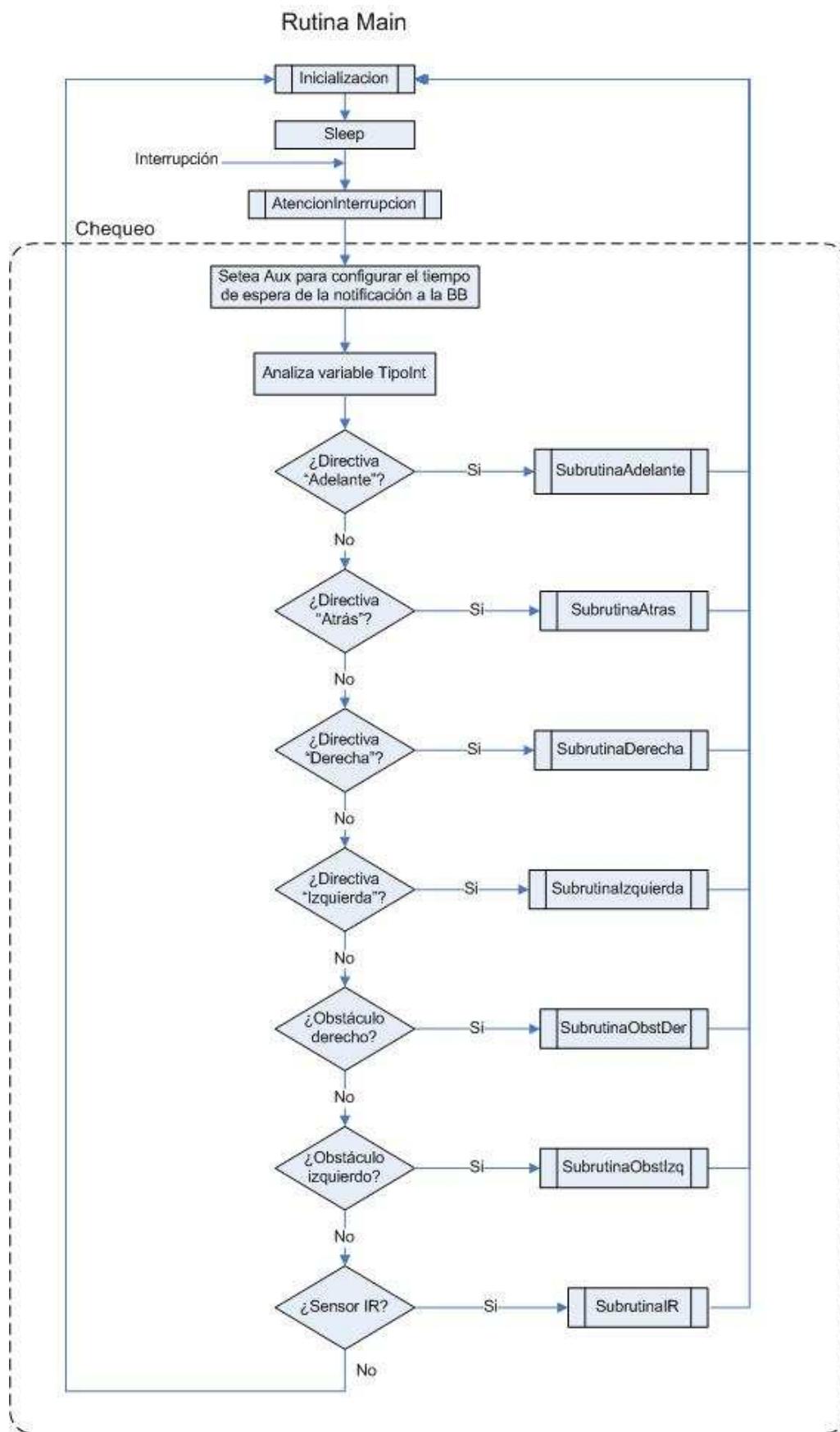


Figura 4.14 - Diagrama de flujo rutina Main



El main del programa ejecuta una rutina de inicialización, la cual pone en cero todas las variables, configura los bits de los puertos como entradas o como salidas según corresponda, habilita y configura las interrupciones que se van a utilizar y por último pone en cero todos los puertos. A continuación ejecuta la instrucción *sleep*. Al ser interrumpido, el PIC se despierta del modo SLEEP y ejecuta la rutina de atención de interrupción.

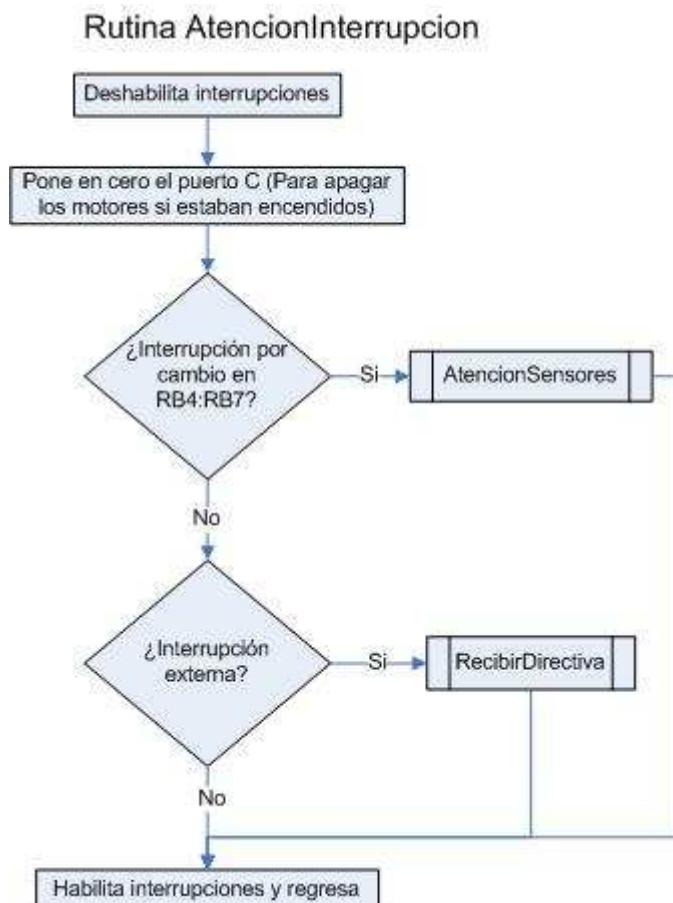


Figura 4.15 - Diagrama de flujo rutina AtencionInterrupcion

La rutina de atención de interrupción deshabilita todas las interrupciones y pone en cero el puerto C para apagar los motores en caso que estuvieran encendidos. Luego chequea si la interrupción fue debido a una condición positiva en algún sensor o bien por el envío de una directiva y ejecuta la subrutina correspondiente (*AtencionSensores* o *RecibirDirectiva*) para luego volver a habilitar las interrupciones y regresar.



AtencionSensores

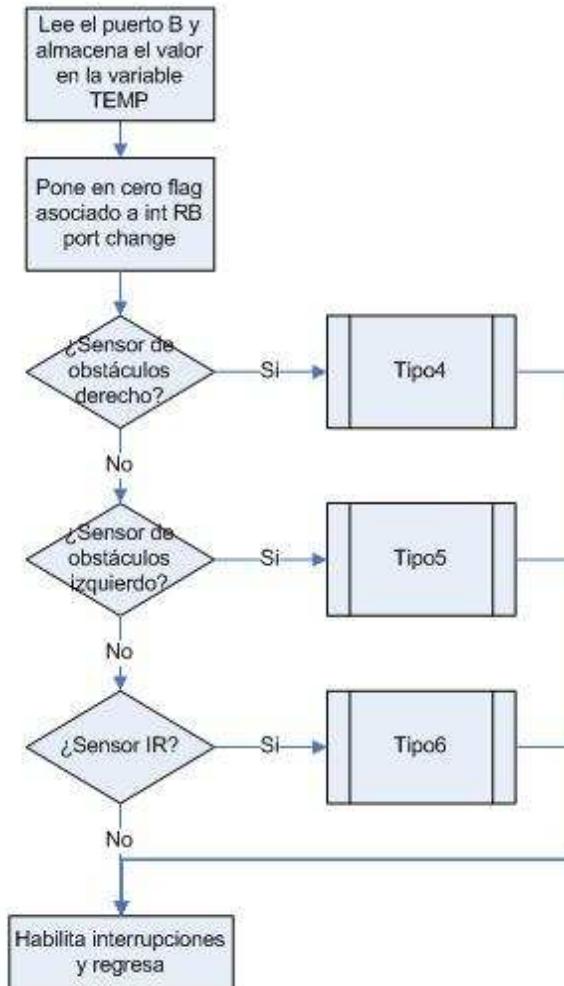


Figura 4.16 - Diagrama de flujo rutina AtencionSensores

La rutina de atención de sensores en primer lugar lee el puerto B y almacena lo leído en la variable TEMP. Luego, pone en cero el flag asociado a la interrupción por cambio en el puerto B, analiza la variable TEMP para determinar cuál de los sensores provocó la interrupción y por último ejecuta la rutina correspondiente (Tipo4, Tipo5 o Tipo6).

Las rutinas Tipo4 y Tipo5 en primer lugar setean el bit correspondiente en la variable Tipoint. Luego, envían un pulso de reset al flip-flop correspondiente para luego volver a habilitar las interrupciones y regresar de la interrupción. La rutina Tipo6 es similar a las anteriores, la única diferencia es que no envía pulso de reset a los flip-flops.



Rutina RecibirDirectiva

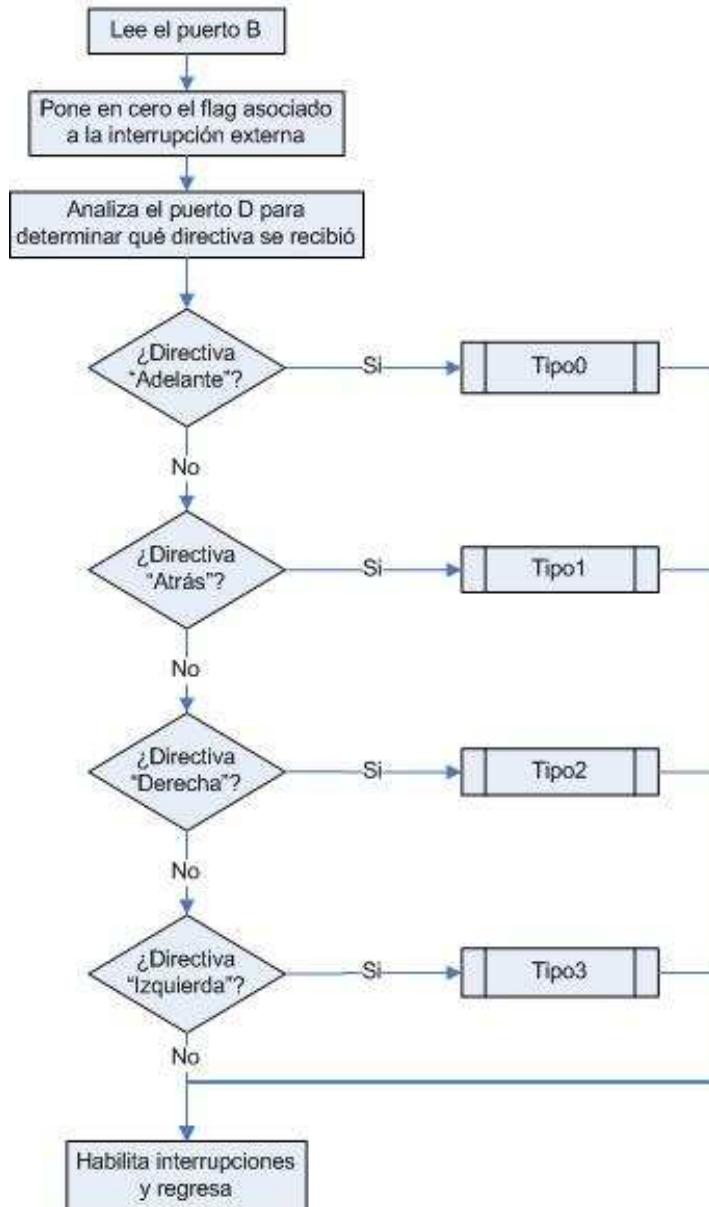


Figura 4.17 - Diagrama de flujo rutina RecibirDirectiva

La rutina RecibirDirectiva lee el puerto B y pone en cero el flag asociado a la interrupción externa. Luego analiza el puerto D para determinar qué directiva se está recibiendo y ejecutar la rutina correspondiente (Tipo0, Tipo1, Tipo2 o Tipo3).

Las rutinas Tipo0, Tipo1, Tipo2 y Tipo3 setean el bit correspondiente en la variable Tipoint, luego vuelven a habilitar las interrupciones y regresan de la interrupción.



Al regresar de la rutina de atención de interrupción, se configura el tiempo de espera para notificar a la placa BB y se analiza la variable Tipoint para determinar qué subrutina se debe ejecutar (SubrutinaAdelante, SubrutinaAtras, SubrutinaDerecha, Subrutinalzquierda, SubrutinaObstDer, SubrutinaObstIzq o SubrutinalR).

SubrutinaAdelante, SubrutinaAtras, SubrutinaDerecha, Subrutinalzquierda



Figura 4.18 - Diagrama de flujo subrutinas directivas

Las subrutinas asociadas a directivas limpian el flag correspondiente de la variable Tipoint y ejecutan la directiva que corresponda (Adelante, Atrás, Derecha o Izquierda). Luego ejecutan una rutina de espera, para luego detener los motores, liberar el motor de tracción y, por último, regresar a la sección Chequeo de la rutina main para verificar si durante su ejecución se produjo otra interrupción.

SubrutinaObstDer, SubrutinaObstIzq, SubrutinalR

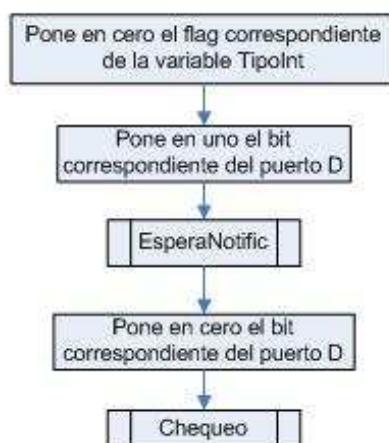


Figura 4.19 - Diagrama de flujo subrutinas sensores



Las subrutinas asociadas a condiciones positivas en los sensores ponen en cero el flag correspondiente de la variable Tipoint y ponen en uno el bit correspondiente del puerto D para enviar la notificación a la placa BB. Luego ejecutan una rutina de espera para dar tiempo a la placa BB a que reciba la notificación. Por último, ponen en cero el bit del puerto D y regresan a la sección Chequeo de la rutina main para verificar si se produjo otra interrupción durante su ejecución.

El código desarrollado se encuentra adjunto en el Anexo IV – Código del PIC.

4.6.3 Testeo del módulo de movimiento

Como primer paso se verificó el funcionamiento del motor de continua conectándolo directamente a una fuente y se identificaron las polaridades de sus cables. Por otro lado, se identificaron las bobinas del motor paso a paso midiendo la impedancia entre sus cables.

Luego se verificó el funcionamiento del driver L293D conectado al motor de continua. Se verificó que al poner un “1” lógico en las entradas En1 e In1 y un “0” lógico en la entrada In2 del L293D, el vehículo se movía hacia adelante. También se verificó que al poner un “1” lógico en las entradas En1 e In2 y un “0” lógico en la entrada In1 del L293D, el vehículo se movía hacia atrás.

Para depurar el código del PIC se utilizaron la herramienta de debug incluida en el entorno de desarrollo MPLAB y la herramienta de debug que ofrece el simulador Proteus. Ambas herramientas permiten ejecutar el programa paso a paso, viendo el estado de los distintos registros del PIC. Esto facilita la detección de comportamientos distintos al esperado y permite corregir errores.

Luego de depurado el código se grabó en el PIC y se verificó su funcionamiento simulando las entradas con pulsadores y observando las salidas con el osciloscopio.

Una vez verificado el funcionamiento del driver L293 conectado al motor de continua y depurado el código del PIC, se realizaron las conexiones entre dicho driver y el PIC y se generó una interrupción simulando una directiva “Adelante” enviada por la BB y el vehículo se movió hacia adelante.



Durante esta prueba se experimentaron comportamientos extraños por parte del PIC. Luego de investigar las posibles causas y realizar diversas pruebas, se llegó a la conclusión de que la causa era el ruido eléctrico que ingresaba al PIC por las entradas flotantes (aquellas que no estaban conectadas a nada). Se solucionaron estos problemas conectando a tierra aquellas entradas que no se utilizarían.

También se experimentaron problemas con las rutinas de espera. Se pudo observar que la ejecución nunca salía de estas rutinas. Se investigó el problema y se descubrió que éste radicaba en un error al configurar el oscilador utilizado por el PIC. Se configuraba XT en lugar de HS. Se solucionó el problema corrigiendo los bits correspondientes en el registro de configuración.

Luego se verificó que el PIC reaccionara correctamente al ser interrumpido por un sensor de obstáculos. Es decir, que al recibir una interrupción en uno de los pines RB4 o RB5, pone en nivel alto durante un breve período de tiempo uno de los bits RD4 o RD5 según corresponda. También se verificó que al tener seteado uno de los bits RD0 o RD1 (simulando una directiva “Adelante” o “Atrás” de la BB) y recibir una interrupción en el pin INT (RB0), el motor de continua se mueve hacia adelante o hacia atrás respectivamente, lo cual representa el correcto funcionamiento de esta sección.

Durante estas pruebas se experimentó un problema: se lograba el correcto funcionamiento del motor de continua controlado por el PIC, solo si los sensores de obstáculos no estaban conectados a los flip-flops. Se analizaron las diferencias entre la situación en la que se lograba el correcto funcionamiento y la situación en la que no y se concluyó que el contacto NC de los microswitches no debía estar conectado directamente a tierra, sino que debía conectarse a través de una resistencia pull-down. Luego de hacer esta corrección se logró el correcto funcionamiento del motor de continua en todo momento.

Al testear el funcionamiento de la lógica de control del motor PaP, no se lograron los resultados esperados por lo que fue necesario ajustar la duración de los pulsos enviados por el PIC al driver L293D. Una vez ajustada dicha duración, se volvió a testear el funcionamiento del motor PaP lográndose el comportamiento esperado.



4.7 Implementación Módulo Comunicación PIC - Placa BB

La información que es necesario transmitir entre el PIC y la placa BB son bits independientes: para directivas de movimiento (Adelante, Atrás, Derecha, Izquierda) desde la placa hacia el PIC y para avisos de condiciones positivas en los sensores (de obstáculos e IR) desde el PIC hacia la placa.

4.7.1 Investigación de los protocolos del puerto de expansión

Como se mencionó en el capítulo 2, el puerto de expansión de la placa BB puede manejar los protocolos: General Port Input Output (GPIO), I2C y SPI. Luego de investigar estos protocolos, se decidió utilizar para la comunicación con el PIC el protocolo GPIO (General Purpose Input/Output) por considerarse el más adecuado para esta aplicación.

4.7.2 Desarrollo protocolo de comunicación

4.7.2.1 A nivel del PIC

Se utilizan los puertos B y D para esta función. Se utiliza la interrupción externa del bit RB0 para advertir al PIC que se le enviará una directiva y los bits RD0:RD3 para enviar las directivas en sí. Para recibir las condiciones positivas de los sensores se utilizan los bits RB4:RB6 y para enviarle a la placa notificaciones de estas condiciones positivas se utilizan los bits RD4:RD6. A modo de resumen la Tabla 4-1 muestra la función de cada bit utilizado para la comunicación con la placa BB.

Pin del PIC	Función
RB0	Entrada. Genera interrupción externa para advertir que se va a transmitir una directiva de movimiento
RD0	Entrada. Directiva “Adelante”
RD1	Entrada. Directiva “Atrás”
RD2	Entrada. Directiva “Derecha”
RD3	Entrada. Directiva “Izquierda”
RB4	Entrada. Sensor de obstáculo Derecho
RB5	Entrada. Sensor de obstáculo Izquierdo
RB6	Entrada. Sensor infrarrojo
RD4	Salida. Notificación sensor de obstáculo Derecho
RD5	Salida. Notificación sensor de obstáculo Izquierdo
RD6	Salida. Notificación sensor infrarrojo

Tabla 4-1 - Resumen de funciones de pines del PIC



4.7.2.2 A nivel de la placa BB

Para esta función se utiliza el protocolo GPIO en el puerto de expansión de la placa BB.

Para definir los pines del puerto de expansión que se utilizarían para la comunicación con el PIC se tuvo en cuenta la Tabla 4-2, la cual se encuentra en el manual de referencia de la placa BB.

Pin	Option A	Option B	Option C	Option D
1		VIO_1V8		
2		DC_5V		
3	MMC2_DAT7	GPIO_139		
4	McBSP3_DX	GPIO_140	UART2_CTS	
5	MMC2_DAT6	GPIO_138		
6	McBSP3_CLKX	GPIO_142	UART2_TX	
7	MMC2_DAT5	GPIO_137		
8	McBSP3_FSX	GPIO_143	UART2_RX	
9	MMC2_DAT4	GPIO_136		
10	McBSP3_DR	GPIO_141	UART2_RTS	
11	MMC2_DAT3	McSPI3_CS0	GPIO_135	
12	McBSP1_DX	McSPI4_SIMO	McBSP3_DX	GPIO_158
13	MMC2_DAT2	McSPI3_CS1	GPIO_134	
14	McBSP1_CLKX	McBSP3_CLKX	GPIO_162	
15	MMC2_DAT1	GPIO_133		
16	McBSP1_FSX	McSPI4_CS0	McBSP3_FSX	GPIO_161
17	MMC2_DAT0	McSPI3_SOMI	GPIO_132	
18	McBSP1_DR	McSPI4_SOMI	McBSP3_DR	GPIO_159
19	MMC2_CMD	McSPI3_SIMO	GPIO_131	
20	McBSP1_CLKR	McSPI4_CLK	SIM_CD	GPIO_156
21	MMC2_CLKO	McSPI3_CLK	GPIO_130	
22	McBSP1_FSR			GPIO_157
23	I2C2_SDA	GPIO_183		
24	I2C2_SCL	GPIO_168		
25		REGEN		
26		nRESET		
27		GND		
28		GND		

Tabla 4-2 - Pines GPIO

La selección de los pines a utilizar tuvo en cuenta que las señales GPIO correspondientes estuvieran bajo la misma opción para evitar un funcionamiento incorrecto por interferencias con señales del sistema.



Se utilizaron como salidas los pines 6, 7, 8, 9 y 10 correspondientes a las señales GPIO_142, GPIO_137, GPIO_143, GPIO_136 y GPIO_141 respectivamente. El pin 6 se utiliza para interrumpir al PIC, el pin 7 se utiliza para enviar la directiva “Adelante”, el pin 8 se utiliza para enviar la directiva “Atrás”, el pin 9 se utiliza para enviar la directiva “Derecha” y el pin 10 se utiliza para enviar la directiva “Izquierda”.

Como entradas se utilizaron los pines 15, 23 y 24 correspondientes a las señales GPIO_133, GPIO_183 y GPIO_168 respectivamente. El pin 15 se utiliza para recibir la notificación del sensor de obstáculos derecho, el pin 23 se utiliza para recibir la notificación del sensor de obstáculos izquierdo y el pin 24 se utiliza para recibir la notificación del sensor infrarrojo.

La Tabla 4-3 resume la utilización de los pines del puerto de expansión de la placa BB.

Pin de la BB	Señal	Función
15	GPIO_133	Entrada. Notificación de sensor de obstáculos derecho.
23	GPIO_183	Entrada. Notificación de sensor de obstáculos izquierdo.
24	GPIO_168	Entrada. Notificación de sensor infrarrojo.
6	GPIO_142	Salida. Interrupción al PIC.
7	GPIO_137	Salida. Directiva “Adelante”
8	GPIO_143	Salida. Directiva “Atrás”
9	GPIO_136	Salida. Directiva “Derecha”
10	GPIO_141	Salida. Directiva “Izquierda”

Tabla 4-3 - Resumen de funciones de pines de la BB

4.7.3 Integración con otros módulos

4.7.3.1 A nivel del PIC

Cuando el PIC es interrumpido por un nivel alto en el pin INT (RB0), se despierta del modo SLEEP y se ejecuta la rutina de atención de interrupción, en la cual se verifica cuál de los bits RD0:RD3 está en estado alto. Dependiendo de cuál sea este bit, se setea un flag en la variable Tipoint y se regresa de la interrupción. Una vez en la ejecución normal, se llamará a la subrutina correspondiente (SubrutinaAdelante, SubrutinaAtras, SubrutinaDerecha, Subrutinalzquierda), la cual hará moverse los motores.

Las condiciones positivas en los sensores también generan interrupciones, pero éstas son del tipo RB Port Change. Al detectarse un obstáculo o un tono IR se genera una interrupción del tipo RB Port Change, la cual despierta al PIC del modo SLEEP. De forma análoga a la comentada en el párrafo anterior, se examina cuál fue el bit que generó la



interrupción y se setea el flag correspondiente en la variable Tipoint. Luego se regresa de la interrupción y se llama a la subrutina correspondiente que se encargará de enviar un “1” lógico a la placa BB en el bit que corresponda (RD4:RD6). A continuación se ejecuta una rutina de espera para dar tiempo a la placa a leer este dato y luego se pone el bit en cero.

4.7.3.2 A nivel de la placa BB

Cuando el usuario hace clic en alguno de los botones de la interfaz Web que están asociados a directivas, se ejecuta un script el cual coloca en estado alto el pin asociado a la directiva que se desea enviar y luego pone en estado alto el pin que genera la interrupción del PIC. Luego ejecuta la rutina *sleep* para dar tiempo a que el PIC reciba la directiva. Por último, pone en estado bajo el pin que genera la interrupción y pone en estado bajo el pin asociado a la directiva.

Cada vez que se envía una directiva, se chequea si se recibió una notificación de alguno de los sensores de obstáculos. En caso de que el móvil se haya chocado con algo y se haya recibido la notificación correspondiente, se notifica de esta situación al usuario final a través de un ícono en la interfaz Web.

Periódicamente se monitorea el estado del pin en el que se recibe la notificación de detección de un superviviente. En caso de recibir una notificación de este tipo, se escribe un “1” en un archivo. El proceso encargado de almacenar en la base de datos las posiciones por las que pasa el móvil (*GPSStore*) es responsable de leer dicho archivo para almacenar en la base de datos el punto con la información del estado del sensor de vida.

En la sección “Desarrollo de la interfaz Web” se explica con mayor detalle el comportamiento de la interfaz Web y se presentan diagramas que permiten visualizar dicho comportamiento.

En principio se había decidido desarrollar un driver que permitiera que los cambios de estado en los pines configurados como entradas produjeran una interrupción y que ésta fuera atendida de la forma adecuada. Sin embargo, no se logró el correcto funcionamiento de este driver por lo que se optó por monitorear el estado de dichos pines



cuando fuera necesario. En el Anexo XII – Desarrollo de drivers en Linux se documenta la investigación realizada para desarrollar el mencionado driver.

4.7.4 Testeo protocolo comunicación

4.7.4.1 A nivel del PIC

Para testear el protocolo de comunicación a nivel del PIC, en principio se simularon las entradas provenientes de la placa BB con pulsadores y las salidas se conectaron a LEDs. De esta forma fue posible generar interrupciones sin que la placa BB estuviera conectada al PIC y ver si el programa del PIC reaccionaba de forma adecuada.

4.7.4.2 A nivel de la placa BB

A nivel de la placa BB, se testeó el protocolo de comunicación colocando LEDs en los pines configurados como salidas y encendiéndolos o apagándolos. Una vez testeado el correcto comportamiento de las rutinas asociadas al envío de directivas (pines configurados como salidas), se testeó el comportamiento de las rutinas asociadas a la recepción de notificaciones (pines configurados como entradas). En este caso se conectaron los pines configurados como entradas a pines configurados como salidas y se verificó que las rutinas de recepción de notificaciones reaccionaran como era esperado ante una notificación.

4.7.4.3 PIC y placa BB

Una vez testeado el protocolo de comunicación desde ambos extremos (PIC y placa BB), se realizó la conexión entre el puerto de expansión de la placa BB y los puertos B y D del PIC y se testeó la comunicación.

Surgieron complicaciones debido a que la placa BB trabaja con una lógica de 1.8 V (es decir que entrega 1.8 V como un “1” lógico), mientras que el PIC trabaja con una lógica de 5 V, con un umbral de 2 V. Por este motivo, al enviar la placa BB un “1” lógico, no es detectado por el PIC.

Se investigaron otros proyectos que integraban la placa BB con PICs intentando buscar la solución a este problema. Sin embargo, el equipo se encontró con la dificultad de que los



circuitos integrados utilizados en otros proyectos no estaban disponibles en el mercado local. Por este motivo fue necesario buscar otra solución.

Se decidió utilizar transistores como interruptores, cuyo funcionamiento fue explicado en el capítulo 2.

La Figura 4.20 muestra un esquema del circuito utilizado.

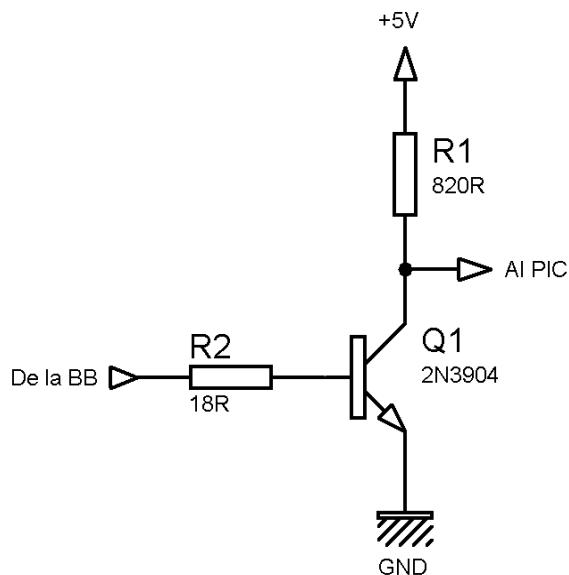


Figura 4.20 - Transistor como interruptor

En esta configuración, cuando se recibe un “1” lógico de la placa BB (1.8 V), el transistor se satura, por lo que la corriente en el colector es máxima. Es decir que a la salida que va al PIC se obtienen 0 V.

Cuando se recibe un “0” lógico de la placa BB (0 V), el transistor está en corte. La corriente en la base es cero, por lo que la corriente en el colector también es cero. Por este motivo, no hay caída de voltaje en la resistencia R1, por lo que la salida que va al PIC entrega 5 V.

Como se puede observar, esta configuración es inversora: cuando se obtiene un “1” lógico de la placa BB, se envía un “0” lógico al PIC y viceversa. Por lo tanto, fue necesario agregar una compuerta NOT entre la salida del transistor y el PIC para no invertir la lógica.



Por otro lado, para enviar notificaciones desde el PIC hacia la placa BB fue necesario utilizar divisores de voltaje para convertir el “1” lógico de 5 V entregado por el PIC en un “1” lógico de 1.8 V. Se diseñó un divisor de voltaje para lograr esta conversión, obteniéndose el circuito que se muestra a continuación:

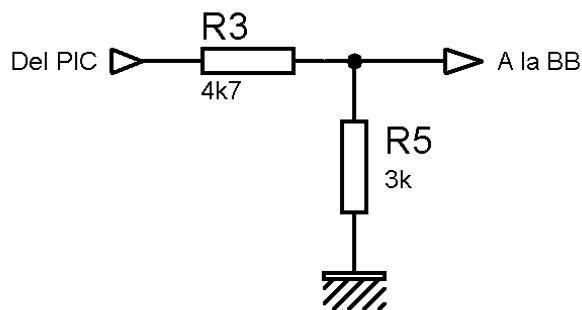


Figura 4.21 - Divisor de voltaje

Debido a que la impedancia equivalente de la placa BB es alta comparada con las resistencias utilizadas en el divisor, la corriente que circula hacia la placa BB es despreciable y no se modifica la diferencia de potencial en el punto donde se conecta la placa BB.



4.8 Implementación Módulo de Detección de Supervivientes

4.8.1 Estudio de integración de simulación de sensor de vida

Se decidió que, para mantener bajos los costos del proyecto y para no perder el foco en el objetivo del proyecto, el módulo de detección de vida sería simulado por otro sensor. Inicialmente se decidió simular el sensor de vida con un sensor de sonido.

El sensor de sonido debe entregarle al PIC un “1” lógico al detectar un tono de una frecuencia determinada. Este “1” lógico entregado por el sensor de sonido genera una interrupción del tipo RB Port Change haciendo que el PIC despierte del modo SLEEP.

Para lograr este objetivo se diseñó un sensor de sonido compuesto por un micrófono, un amplificador operacional, un detector de tono, resistencias, capacitores y un potenciómetro.

El micrófono capta los sonidos del ambiente y entrega una señal de voltaje. Mediante pruebas prácticas se analizó la señal que entrega el micrófono al reproducir a través de un auricular un tono de 1 kHz. Se pudo observar una sinusoides aproximadamente centrada en 0 V de aproximadamente 35 mV pico a pico con frecuencia 1 kHz.

El menor voltaje de entrada que puede captar el detector de tono es de 20 mV rms. Por este motivo, es necesaria una pequeña amplificación para entregar la salida del micrófono al detector de tono. Con este propósito se implementó una etapa de amplificación utilizando un amplificador operacional.

La señal que se obtiene a la salida del amplificador es entregada al detector de tono, el cual activará un interruptor controlado por voltaje cuando detecte una señal de la frecuencia configurada. Esto hará que el PIC reciba un “1” lógico en la entrada correspondiente al sensor de sonido (RB6).

Inicialmente se utilizó como opamp el conocido LM741. Luego de múltiples pruebas e intentos de lograr el correcto funcionamiento de la etapa de amplificación, se concluyó que el LM741 no era adecuado para la aplicación del proyecto. Por este motivo se procedió a investigar sobre opamps más adecuados a aplicaciones de audio y se optó por el TL070.



El TL070 es un amplificador operacional de bajo ruido con entrada JFET. Tiene bajas corrientes de polarización de entrada, bajas corrientes de offset y una rápida tasa de respuesta. Esto logra baja distorsión de los armónicos y bajo ruido, por lo que el opamp TL070 resulta ideal para aplicaciones de pre-amplificación de audio y alta fidelidad.^{xxxii}

Como detector de tono se decidió utilizar el integrado LM567, el cual al detectar una entrada cuya frecuencia está en la banda pasante y cuyo voltaje supera cierto umbral, entrega un “0” lógico en su salida.

Inicialmente se decidió que la señal de salida del detector de tono activaría un relé. Sin embargo, luego de realizar pruebas se concluyó que la corriente de salida del detector de tono no era suficiente para accionar el relé. Por este motivo se decidió que la salida del detector de tono sería entregada a una compuerta NOT de forma de obtener un “1” lógico cuando el detector de tono entregara un nivel bajo como salida y un “0” lógico, cuando el detector de tono entregara un nivel alto como salida.

Teniendo en cuenta las recomendaciones que aparecen en las hojas de datos de los componentes a utilizar, se diseñó el circuito del sensor de sonido que se muestra en la Figura 4.22.

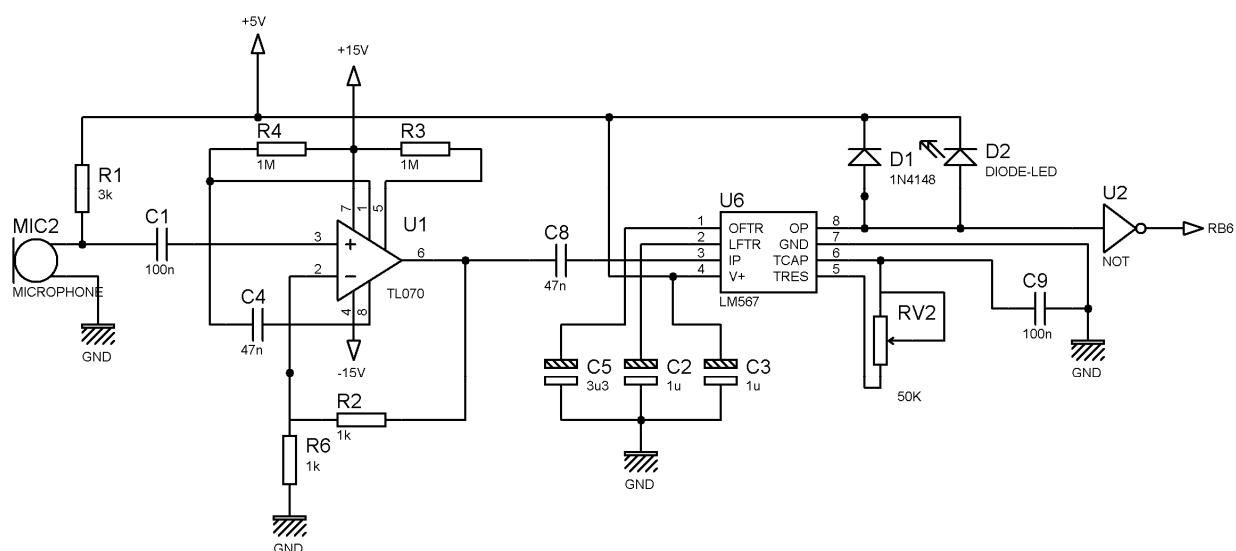


Figura 4.22 - Circuito sensor de sonido



4.8.2 Desarrollo de código de sensor de vida

El código del PIC asociado al sensor de vida, cumple con la función de enviar una notificación a la placa BB cuando el sensor que simula el sensor de vida detecte una condición positiva.

Al generarse una interrupción debido a un cambio en el pin RB6, el PIC se despierta del modo SLEEP y ejecuta la rutina de atención de interrupción. Al verificar que fue el sensor de vida el que generó la interrupción, pone en “1” el bit 6 de la variable Tipoint y regresa de la interrupción.

Una vez de regreso a la ejecución normal, llama a la subrutina correspondiente (SubrutinaIR), la cual pone en “1” el pin RD6 para notificar a la placa BB de la detección del tono. Luego, ejecuta una rutina de espera para dar tiempo a la placa BB a que lea la notificación y por último, pone en “0” el pin RB6 y vuelve a ejecutar la rutina Main.

4.8.3 Testeo sensor de simulación de vida

Inicialmente se armó el circuito diseñado y se testeó su funcionamiento, pero no se lograron los resultados esperados. Por este motivo se decidió testear el circuito por partes.

En principio se armó la etapa del micrófono y se observó la señal de salida al reproducir a través de un auricular un tono de 1 kHz. Se pudo observar una sinusoides aproximadamente centrada en 0 V, de 35 mV pico a pico aproximadamente, con frecuencia 1 kHz.

Conociendo la forma de la onda, se realizaron pruebas en la etapa de detección generando la señal de entrada con un generador de onda. Observando con el osciloscopio la señal en el pin 5 del LM567 se ajustó la frecuencia central a 1 kHz. Luego se verificó que al ingresar por el pin 3 del LM567 una sinusoides de 1 kHz, el integrado entregó un nivel bajo en su salida (pin 8).

Una vez implementada y testeada la etapa de amplificación, se procedió a integrar esta etapa con la etapa de detección de tono. Se utilizó como entrada al opamp una sinusoides de frecuencia 1 kHz generada por el generador de onda. Luego de amplificar esta



sinusoide se la utilizó como entrada al LM567 y se verificó el correcto funcionamiento de la etapa de detección de tono. Es decir, mientras la sinusoide generada por el generador de onda sea de 1 kHz de frecuencia, el LM567 proporciona un nivel bajo a su salida (verificado utilizando un LED). Si la frecuencia de la sinusoide es modificada fuera del ancho de banda configurado para el detector de tono, el LED se apaga.

Luego de verificar el correcto funcionamiento de las tres etapas por separado (micrófono, amplificador, detector de tono) se procedió a integrar las tres etapas. No se logró el correcto funcionamiento de las tres etapas juntas, ya que al conectar la salida del micrófono al opamp se pudo observar que la señal se contamina con ruido, por lo que no es detectada por el LM567.

Para solucionar este problema se soldó la etapa del micrófono a una placa universal de pertinax. Se pretendía que el ruido eléctrico fuera menor que en el ProjectBoard pero no se logró este resultado. Como segunda solución se decidió implementar un filtro activo para filtrar la salida del micrófono.

Utilizando la aplicación FilterPro de Texas Instrument se diseñó un filtro activo pasa banda de primer orden, centrado en 1 kHz, factor de calidad 10 y ganancia 2, obteniéndose el circuito que se muestra en la Figura 4.23.

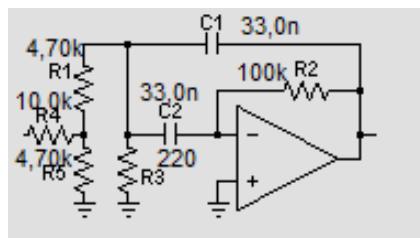


Figura 4.23 - Filtro activo de primer orden



En la Figura 4.24 se muestra la respuesta en frecuencia (amplitud en verde; fase en rojo) del filtro diseñado.

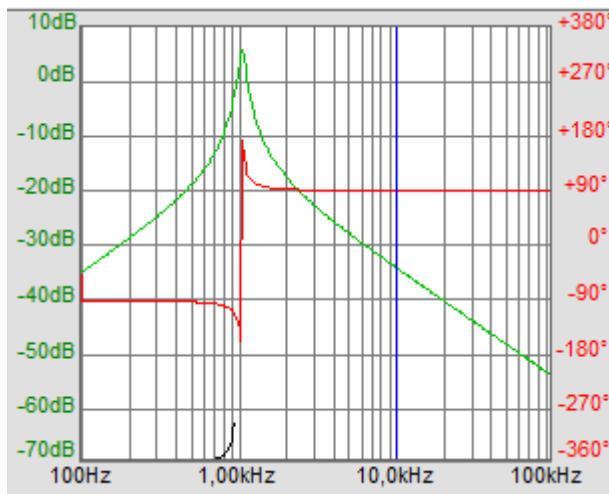


Figura 4.24 - Respuesta en frecuencia filtro activo

Este filtro tampoco logró resolver los problemas que se estaban experimentando con el sensor de sonido.

4.8.4 Sensor infrarrojo

Luego de múltiples pruebas y modificaciones sobre el sensor de sonido sin lograr el funcionamiento esperado, se evaluó la posibilidad de simular el sensor de vida de otra forma. Se tuvo en cuenta que el ambiente en el que habría de trabajar el sensor de sonido era muy ruidoso (debido a la presencia de los motores).

Se decidió que lo más adecuado a la aplicación era simular el sensor de vida con un sensor infrarrojo (IR), el cual activa una salida al detectar una luz IR de una frecuencia determinada.

Para lograr dicho objetivo, solo es necesario modificar algunos componentes del sensor de sonido que se había diseñado (el micrófono y los valores de algunas resistencias y condensadores). El micrófono fue sustituido por un receptor IR y el condensador a la salida del receptor IR se cambió por un condensador de $1 \mu\text{F}$.



Una vez implementado el sensor, se observó la señal a la salida del fototransistor y se concluyó que la etapa de amplificación no era necesaria, por lo que se eliminó la misma del circuito.

El LM567 entrega un nivel alto en todo momento, excepto cuando detecta una señal de la frecuencia configurada. La salida del LM567 debe interrumpir al PIC por lo que es necesario invertir la lógica y además bajar el voltaje de 9 V a 5 V.

Se hicieron pruebas con el relé a la salida del LM567 y no se logró el funcionamiento deseado, por lo que se decidió utilizar una compuerta NOT, la cual permite que su entrada sea de 9 V.

En la Figura 4.25 se muestra un esquema del sensor IR diseñado.

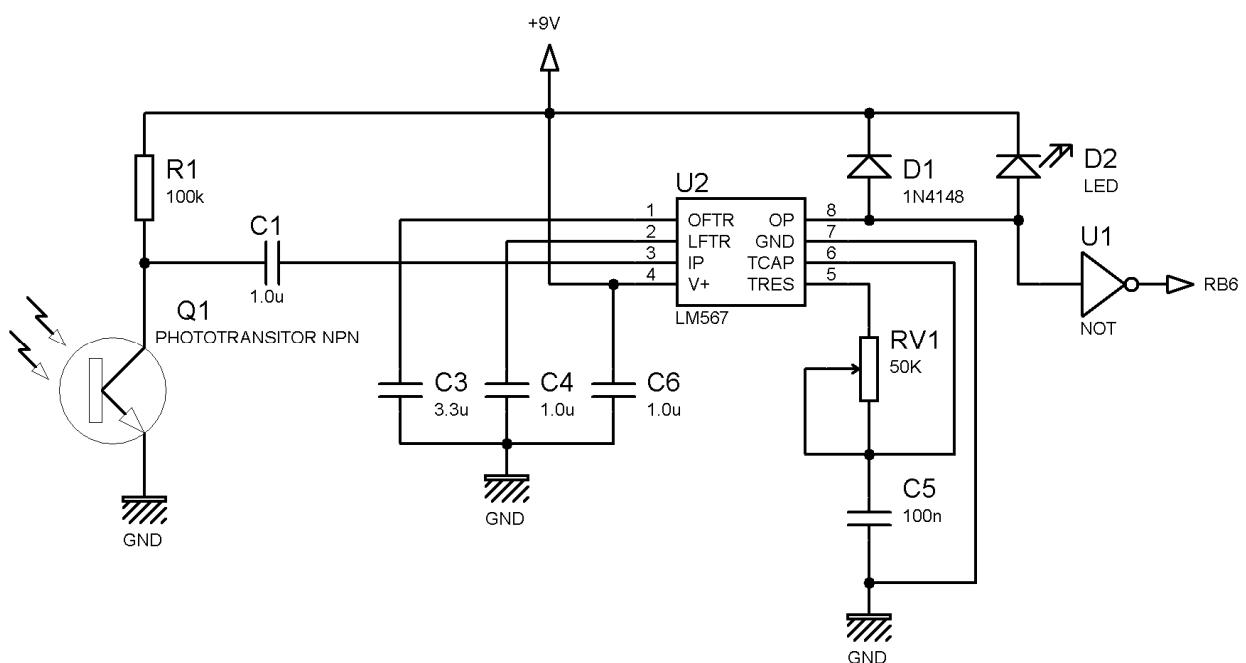


Figura 4.25 - Sensor IR

El funcionamiento de este sensor es análogo al funcionamiento del sensor de sonido que fue explicado en las secciones anteriores.

Además, se diseñó un emisor IR, el cual emite una luz IR a una frecuencia determinada. Para lograr que el LED IR emita a una frecuencia determinada es necesario aplicar una



onda cuadrada en sus bornes. Se utilizó un transistor PNP (2N2907), el cual funciona como oscilador al conmutar entre su región de corte y su región de saturación. También se utilizó un transistor NPN (2N3904) para amplificar la señal.

En la Figura 4.26 se muestra el esquema del emisor IR.

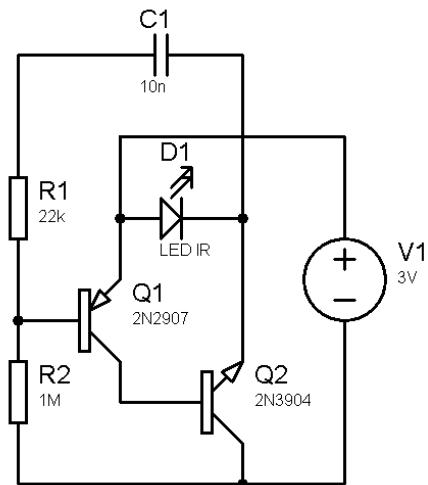


Figura 4.26 - Esquema emisor IR



4.9 Adaptación e integración de Módulo de Detección de obstáculos

4.9.1 Estudio de integración de sensores de obstáculos

Para implementar los sensores de obstáculos se utilizan microswitches reciclados de un mouse de computadora, cuyo funcionamiento fue explicado en el capítulo 2.

El vehículo fue diseñado con dos paragolpes para permitir distinguir entre un choque del lado derecho y un choque del lado izquierdo. Estos paragolpes, al ser empujados, mueven unas cintas metálicas que accionan los microswitches. Estos últimos, al ser accionados generan un “1” lógico que interrumpe al PIC y lo hace despertarse del modo SLEEP.

En la rutina de atención de interrupción, al detectar que la interrupción fue debida a un choque, el PIC notifica a la placa BB enviándole en “1” lógico en el pin correspondiente (RD4 si el choque fue del lado derecho, RD5 si el choque fue del lado izquierdo).

4.9.2 Desarrollo del código para evasión de obstáculos y movimiento automatizado

Uno de los objetivos del proyecto era lograr que el vehículo pudiera moverse de forma automatizada, sin intervención del usuario final, esquivando los obstáculos que detectara.

Diversas dificultades llevaron a que el cronograma del proyecto no pudiera ser cumplido tal cual había sido planificado. A poco más de un mes de la fecha límite de entrega de este proyecto, aún no se habían comenzado a ejecutar las tareas necesarias para cumplir con el objetivo antes mencionado.

El equipo de proyecto estimó que el desarrollo del módulo que cumpliría con la función de evasión de obstáculos y movimiento automatizado requeriría al menos un mes. Teniendo en cuenta que también había que disponer tiempo para pruebas y ajustes y que podrían surgir imprevistos, se corría el riesgo de no poder cumplir con la fecha límite de entrega.

Por lo tanto, se solicitó al Consejo un cambio en el alcance del proyecto. Este cambio fue aprobado, por lo que no se desarrolló el módulo de referencia, postergándose esta tarea para una posible futura ampliación de este proyecto.



4.10 Implementación del Módulo de Posicionamiento Global

Para poder integrar al proyecto un receptor GPS que permitiera obtener la posición del móvil en todo momento, fue necesario investigar sobre el sistema GPS en sí y las distintas formas existentes para lograr la interacción entre un receptor GPS y un sistema operativo. Por otro lado, fue necesario desarrollar código que permitiera obtener la información de posición y que la convirtiera al Datum utilizado por la cartografía uruguaya. Por último, fue necesario realizar un testeo de este código y de su integración con el módulo de movimiento.

4.10.1 Investigación de interacción con GPS

Durante la tarea de investigación sobre la forma en que el sistema de este proyecto podía interactuar con un receptor GPS, se recopiló información teórica a partir de la cual se redactaron las secciones correspondientes del capítulo 2.

En principio se investigó sobre el sistema GPS para comprender su funcionamiento. Luego se procedió a investigar sobre el estándar NMEA utilizado para la transmisión de datos entre receptores GPS o entre un receptor GPS y otro dispositivo.

Durante la investigación de estos temas se tomó conocimiento de la existencia de módulos GPS, los cuales son pequeños circuitos, algunos de ellos con antena incorporada, que prescinden de periféricos innecesarios para este proyecto como ser la interfaz de usuario (pantalla, botones, etc.). Se consideró que podrían servir para el proyecto por su portabilidad pero luego, esta idea fue descartada debido al alto costo de estos módulos, teniendo en cuenta que los costos del proyecto se deseaban mantener lo más bajos posible. En su lugar, se utiliza un receptor GPS comercial de mano.

Durante esta primera etapa de investigación, también se dedicó cierto tiempo a capacitación en lenguaje C (lectura de tutoriales, programas de ejemplo, etc.) debido a que éste fue el lenguaje que se seleccionó para el desarrollo de los módulos que habrían de ejecutarse en la placa BB.

Cabe aclarar que además de las restricciones mencionadas en el Plan de Proyecto, existen ciertas restricciones asociadas al sistema GPS. En primer lugar, en el hemisferio Sur no existe la posibilidad de hacer uso del sistema GPS Diferencial. Por otro lado, la



disponibilidad de las coordenadas de posición del móvil y la precisión de estas coordenadas dependen de que el receptor GPS tenga una buena recepción de las señales satelitales.

4.10.2 Obtención de la posición del receptor GPS desde un PC de pruebas

Para realizar las primeras pruebas y decidir las configuraciones necesarias para obtener la posición del receptor GPS, se instaló en un PC de pruebas el mismo sistema operativo que se encuentra instalado en la placa BB, es decir, Linux Ubuntu 9.04 (Jaunty Jackalope). Esto permitió poder simular el entorno definitivo sin necesidad de acceder a la placa BB y de esta forma los integrantes del equipo de proyecto pudieron seguir avanzando en el proyecto de forma independiente.

4.10.2.1 Instalación y configuración de GPS Daemon

Para simplificar el proceso de instalación se utilizó el gestor de paquetes “apt-get”, el cual fue descripto en el capítulo 2 de este documento. Por lo tanto, el proceso de instalación de GPSD se redujo a ejecutar el siguiente comando:

```
apt-get install gpsd
```

Luego, fue necesario copiar el archivo 40-gpsd.rules desde el directorio /lib/udev/rules.d hacia el directorio /etc/udev/rules.d para activar las reglas contenidas en ese archivo. Estas reglas especifican los dispositivos que pueden ser receptores GPS y establecen que al detectarse uno de estos dispositivos, se ejecute GPSD.

4.10.2.2 Detección del receptor GPS en el PC de pruebas

Se comenzó realizando pruebas con dos receptores marca Garmin que se obtuvieron en préstamo, uno de ellos modelo eTrex Venture HC y el otro, modelo eTrex Vista Cx. Estos modelos cuentan con una interfaz USB para ser conectados a un host.

Surgió el primer problema al conectar uno de los receptores GPS y ver que, si bien éste era reconocido por el sistema operativo, el nodo correspondiente no era creado en el directorio /dev.



Se comprobó que el dispositivo era reconocido por el sistema operativo ejecutando los comandos `lsusb` y `dmesg | tail`, cuyas salidas se muestran a continuación:

```
root@ubuntu:/home/sofia# lsusb
Bus 002 Device 004: ID 0bda:8197 Realtek Semiconductor Corp. RTL8187B Wireless Adapter
Bus 002 Device 003: ID 0930:6545 Toshiba Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 003: ID 091e:0003 Garmin International GPSmap (various models)
Bus 005 Device 002: ID 046d:c50e Logitech, Inc. MX-1000 Cordless Mouse Receiver
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

```
root@ubuntu:/home/sofia# dmesg | tail
(...)
[ 151.424033] usb 5-2: new full speed USB device using uhci_hcd and address 3
[ 151.582328] usb 5-2: configuration #1 chosen from 1 choice
```

Sin embargo, al ejecutar el comando `ls -l /dev | grep ttyUSB*` se pudo comprobar que, contrario a lo que se esperaba, el dispositivo `/dev/ttyUSB0` no era creado por udev.

Se intentó ejecutar GPSD pasándole como dispositivo `/dev/usbdev5.2_ep81` debido a que el mensaje que se puede ver en la salida del comando `dmesg` muestra que el receptor GPS está conectado en usb 5-2. Sin embargo, esto tampoco dio resultado, obteniéndose el mensaje: “`gpsd: GPS device /dev/usbdev5.2_ep81 nonexistent or can't be read`”.

Investigando en distintos foros de Ubuntu, se tomó conocimiento de que, a partir de Ubuntu 8.04, el driver `garmin_gps` fue puesto en la lista negra (blacklist). Este driver hace accesible un dispositivo Garmin USB a través de un puerto serial (`ttyUSB*`) y, por lo tanto, es necesario para que Ubuntu pueda manejar correctamente los receptores GPS Garmin con interfaz USB.

Luego de comentar la línea que coloca al driver `garmin_gps` en la lista negra (editando el archivo `/etc/modprobe.d/blacklist.conf`), se logró que al conectar el GPS, éste fuera reconocido correctamente por el sistema operativo y que el dispositivo `ttyUSB0` fuera creado en el directorio `/dev`.



A continuación se muestra la salida del comando *dmesg | tail*.

```
root@ubuntu:/home/sofia# dmesg | tail
(...)
[ 384.064024] usb 5-2: new full speed USB device using uhci_hcd and address 3
[ 384.221970] usb 5-2: configuration #1 chosen from 1 choice
[ 384.260175] USB Serial support registered for Garmin GPS usb/tty
[ 384.260193] garmin_gps 5-2:1.0: Garmin GPS usb/tty converter detected
[ 384.260267] usb 5-2: Garmin GPS usb/tty converter now attached to ttyUSB0
[ 384.260277] usbcore: registered new interface driver garmin_gps
[ 384.260279] garmin_gps: v0.31:garmin gps driver
```

Como se puede observar, se detecta un conversor USB/tty. Esto se debe a que los receptores GPS USB emulan una comunicación serial RS 232 utilizando chips conversores. Es por este motivo que fue necesario permitir que el driver *garmin_gps* se cargara. Este driver es el que dispone de la lógica para manejar el conversor USB/tty utilizado por los receptores GPS Garmin con interfaz USB.

Luego de investigar un poco más, se decidió que era mejor dejar el driver *garmin_gps* en la lista negra (esto es lo estándar) para que no se cargue durante el arranque del sistema operativo y cargarlo manualmente cuando fuera necesario. Para cargar manualmente un driver es necesario ejecutar el comando *modprobe*. En este caso se ejecuta *modprobe garmin_gps*.

4.10.2.3 Ejecución de GPSD en el PC de pruebas

El programa GPSD se ejecuta automáticamente al conectar el receptor GPS a la computadora. Se ejecutó el comando *killall gpsd* para poder ejecutarlo con ciertas opciones.

Se ejecutó GPSD con las opciones b, n, N, D y pasándole el dispositivo que corresponde al receptor GPS, es decir, se ejecutó: *gpsd -b -n -N -D 9 /dev/ttyUSB0*. La opción “-b” hace que GPSD se ejecute en modo read-only, es decir, que no le envíe comandos al receptor GPS. La opción “-n” hace que GPSD se ejecute sin esperar que se conecte una aplicación cliente para sondear el GPS. La opción “-N” hace que GPSD se ejecute en foreground, es decir, que no se ejecute en background y de esa forma se pueda ver la salida. La opción “-D 9” establece el nivel de debug en 9. Y por último se especifica cuál es el dispositivo GPS.



Luego de ejecutar GPSD como se explicó anteriormente se obtuvo la siguiente salida:

```
root@ubuntu:/home/sofia# gpsd -b -n -N -D 9 /dev/ttyUSB0
gpsd: launching (Version 2.39)
gpsd: listening on port gpsd
gpsd: Priority setting failed.
gpsd: shmat(0,0,0) succeeded
gpsd: shmat(32769,0,0) succeeded
gpsd: shmat(65538,0,0) succeeded
gpsd: shmat(98307,0,0) succeeded
gpsd: successfully connected to the DBUS system bus
gpsd: running with effective group ID 0
gpsd: running with effective user ID 0
gpsd: opening read-only GPS data source at '/dev/ttyUSB0'
gpsd: speed 9600, 8N1
gpsd: Can't open /proc/bus/usb/devices
gpsd: no probe matched...
gpsd: gpsd_activate(1): opened GPS (5)
gpsd: select waits
```

GPSD intenta abrir el directorio `/proc/bus/usb/devices`, debido a que el módulo `garmin_gps` utiliza este directorio. Como se puede observar, GPSD no puede abrir el directorio. Esto se debe a que dicho directorio ya no es soportado por temas de seguridad del sistema de archivos `usbfs`.

Se solucionó este problema montando el sistema de archivos `usbfs` en el directorio `/proc/bus/usb`. Lo cual se logra ejecutando el comando: `mount -t usbfs none /proc/bus/usb`. Como esto va a ser necesario siempre, se editó el archivo `/etc/fstab` para que se monte el sistema de archivos `usbfs` durante el arranque. En el archivo `/etc/fstab` se agregó la línea: `none /proc/bus/usb usbfs defaults 0 0`.

Por otro lado, durante la investigación para solucionar este problema se encontró que para que GPSD pueda leer los datos del GPS es necesario crear una regla de udev para que se le apliquen los permisos correctos al dispositivo.

Por este motivo se creó el archivo `/etc/udev/rules.d/51-garmin.rules` con el contenido que se muestra a continuación:

```
SUBSYSTEM!="usb_device", GOTO="garmin_rules_end" ACTION!="add",
GOTO="garmin_rules_end" ATTRS{idVendor}=="091e", ATTRS{idProduct}=="0003",
MODE="0660", GROUP="plugdev" LABEL="garmin_rules_end"
```



Luego de solucionar estos inconvenientes se volvió a ejecutar GPSD obteniéndose la salida que se muestra a continuación:

```
root@ubuntu:/home/sofia# gpsd -b -n -N -D 9 /dev/gps0
gpsd: launching (Version 2.39)
gpsd: listening on port gpsd
gpsd: Priority setting failed.
gpsd: shmat(0,0,0) succeeded
gpsd: shmat(32769,0,0) succeeded
gpsd: shmat(65538,0,0) succeeded
gpsd: shmat(98307,0,0) succeeded
gpsd: successfully connected to the DBUS system bus
gpsd: running with effective group ID 0
gpsd: running with effective user ID 0
gpsd: opening read-only GPS data source at '/dev/ttyUSB0'
gpsd: speed 9600, 8N1
gpsd: Set garmin_gps driver mode = 0
gpsd: PrintUSBPacket()
gpsd: Private, Set Mode: 1
gpsd: SendPacket(), wrote 0 bytes
gpsd: probe found Garmin USB binary driver...
gpsd: ntpd_link_activate: 0
gpsd: Create Thread gpsd_ppsmonitor
gpsd: gpsd_activate(1): opened GPS (5)
gpsd: select waits
gpsd: select waits
```

Como se puede observar, GPSD queda a la espera de que algún programa cliente se conecte al socket y solicite información. Por lo tanto, para verificar si GPSD está obteniendo la información del GPS, en otra terminal se ejecuta un telnet al puerto 2947 y luego se envía algún comando. A continuación se muestran los resultados de esta prueba:

```
root@ubuntu:/home/sofia# telnet localhost 2947
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^].
w
GPSD,X=1264280971.447764,I=Garmin USB binary
GPSD,W=1
r
GPSD,R=1
p
GPSD,P=?
```

Con esta prueba se pudo concluir que GPSD no estaba obteniendo la información de posición del GPS, debido que al consultar por la posición (con el comando “p”) se obtuvo como respuesta “P=?”.



Se consideró que una de las posibilidades podía ser que el modelo que se estaba utilizando no fuera compatible con GPSD. Sin embargo, se descartó esta posibilidad luego de consultar la lista de hardware soportado y confirmar que ambos modelos (eTrex Venture HC y eTrex Vista Cx) estaban en dicha lista.

Se consultaron diversos foros y se halló que existen multitudes de usuarios de GPSD que han tenido inconvenientes con los receptores que disponen de interfaz USB. Pensando que, tal vez, algunos problemas hubieran sido solucionados en las últimas versiones de GPSD y de las bibliotecas utilizadas, se procedió a realizar una actualización completa.

Se actualizaron los siguientes paquetes:

- Gpsd
- Gpsd-clients (Clientes de ejemplo)
- Udev
- Libusb
- Libusb-dev

Estas actualizaciones tampoco solucionaron el problema. Por este motivo, se empezó a dudar de que el receptor GPS estuviera enviando la información. Para verificar esto se realizaron pruebas con otro programa: GPSBabel.

Se ejecutó GPSBabel con las siguientes opciones: “-T” (real-time tracking), para obtener información en tiempo real; “-i garmin,get_posn”, “garmin” para establecer que la entrada es un dispositivo marca Garmin y “get_posn” para solicitar la posición; y “-f /dev/ttyUSB0”, para especificar cuál es el dispositivo que se debe leer. Como se muestra a continuación, se logró obtener la información de posición:

```
root@ubuntu:/home/sofia# gpsbabel -T -i garmin,get_posn -f /dev/ttyUSB0
Tue Jan 26 16:41:15 2010
34.881962S 56.163913W 88.241898
Tue Jan 26 16:41:16 2010
34.881956S 56.163913W 87.757500
Tue Jan 26 16:41:17 2010
34.881947S 56.163912W 87.103218
```

En esta etapa, surgió un comportamiento extraño. Estando GPSD ejecutándose en foreground en un terminal, al ejecutar GPSBabel en otro terminal y luego matar el proceso presionando Ctrl+C, se logró obtener información de posición en GPSD.



GPSBabel también se comportaba de forma extraña, funcionando correctamente o dejando de funcionar de forma aleatoria en apariencia.

Investigando se descubrió que pasándole a GPSBabel *usb*: como dispositivo y no */dev/ttyUSB0* se logra que GPSBabel utilice libusb para acceder al dispositivo y no utilice el driver *garmin_gps*. Si bien esto no representaba una solución al problema, dio la pauta sobre por dónde continuar la investigación.

Se pudo concluir que el mal funcionamiento de GPSD se debía exclusivamente a que el driver *garmin_gps* no funciona correctamente. Leyendo en foros se descubrió que, por parte de los desarrolladores y encargados del mantenimiento de Ubuntu, no existe intención de arreglar este driver debido a que las aplicaciones más recientes acceden a los receptores GPS con interfaz USB directamente a través de un puerto USB nativo (utilizando la biblioteca libusb) y no emulan un puerto serial. Un ejemplo de estas aplicaciones es GPSBabel.

GPSD solo utiliza el protocolo serial. Aparentemente, esto fue diseñado así explícitamente para no depender de la biblioteca libusb. Esto significa que para utilizar GPSD con dispositivos Garmin con interfaz USB es necesario utilizar el driver *garmin_gps*.

4.10.2.4 GPSD vs. GPSBabel

Teniendo en cuenta los resultados obtenidos hasta este punto, fue necesario decidir cómo proceder. Las dos alternativas existentes eran las siguientes:

1. Conseguir otro receptor GPS que tuviera interfaz serial para poder utilizar GPSD sin necesidad de cargar el módulo *garmin_gps*.
2. Utilizar GPSBabel en lugar de GPSD.

A simple vista parecería que la mejor opción es la segunda. Sin embargo, en el momento de tomar esta decisión ya se había desarrollado código para conectarse al socket creado por GPSD y obtener de éste la información de posición.

GPSBabel no dispone de la opción de enviar la información obtenida del receptor GPS a un socket, sino que permite que ésta sea enviada a un archivo. Por este motivo, la



elección de la segunda opción hubiera implicado desarrollar de cero una aplicación que leyera los archivos creados por GPSBabel.

Por otro lado, poco después de plantearse la necesidad de tomar esta decisión se consiguió un receptor GPS con interfaz serial, por lo que se decidió llevar a cabo la primera alternativa.

4.10.2.5 Receptor GPS serial

Afortunadamente se logró conseguir en préstamo un receptor GPS con interfaz serial (Garmin 12XL), por lo que se pudo comenzar a realizar pruebas con este receptor.

Debido a que la notebook utilizada para las pruebas no cuenta con un puerto serial fue necesario utilizar un adaptador serial/USB.

Para poder conectar el receptor GPS al adaptador serie/USB es necesario un cable de datos. Como no se disponía de este cable, se procedió a investigar la salida de datos del receptor GPS para poder determinar las funciones de los pines.

Investigando en Internet, se consiguió un esquema que muestra cómo deben conectarse los pines de la salida de datos del receptor GPS a los pines del conector RS 232.

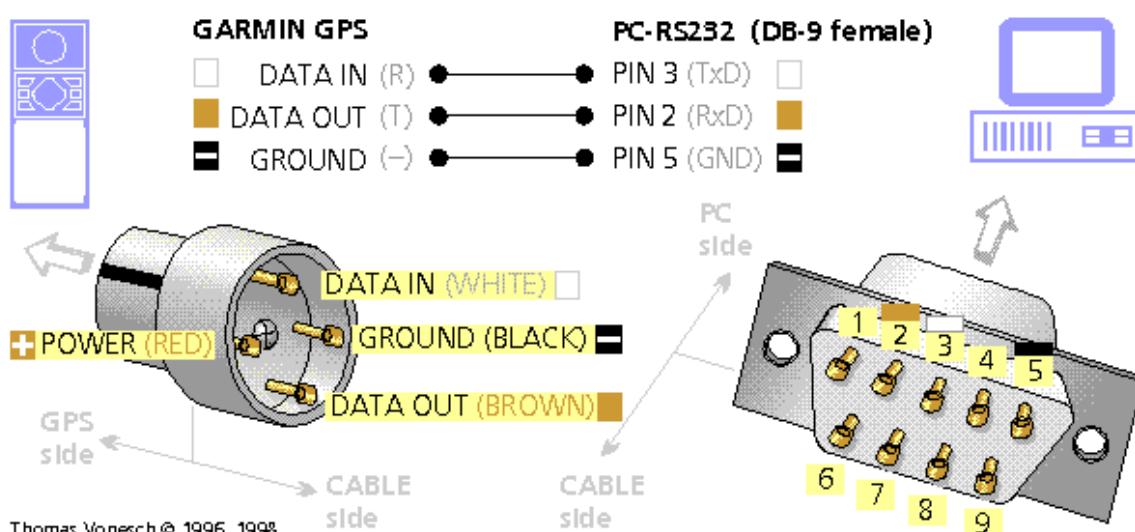


Figura 4.27 - Pines de la salida de datos del receptor GPS



En base a este esquema se adaptó un cable que permitió conectar el GPS a la notebook de pruebas a través del adaptador serie/USB. Luego, se configuró la interfaz del receptor GPS para transmitir datos en formato NMEA compatibles con NMEA 0183 2.0 a 4800bps.

Para verificar que el GPS estaba enviando información se ejecutó el comando `cat /dev/ttyUSB0` obteniéndose la salida que se muestra a continuación:

```
root@ubuntu:/home/sofia# cat /dev/ttyUSB0
$GPGSV,3,3,10,27,87,196,31,30,15,258,00,,,,,,,*76
$PGRME,21.5,M,32.2,M,38.9,M*19
$GPGLL,3452.900,S,05609.823,W,185426,A*23
$PGRMZ,202,f,3*1B
^C
```

Al conectar el GPS, GPSD se ejecuta automáticamente. Esto se verificó ejecutando el comando `ps -ef | grep gpsd`, cuya salida se muestra a continuación:

```
root@ubuntu:/home/sofia# ps -ef | grep gpsd
nobody  4404   1  0 16:30 ?    00:00:00 gpsd -F /var/run/gpsd.sock
root    4497  3877  0 16:38 pts/2  00:00:00 grep gpsd
```

Para consultar la información procesada por GPSD se hizo un telnet al puerto 2947, con lo cual se estableció una comunicación con el socket de control de GPSD y se pudieron enviar comandos solicitándole a GPSD distinta información. La interacción que se llevó a cabo se muestra a continuación:

```
root@ubuntu:/home/sofia# telnet localhost 2947
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^].
{"class":"VERSION","release":"2.90","rev":"svnexportado","proto_major":3,"proto_minor":1}
GPSD
i
GPSD,I=Generic NMEA
w
GPSD,W=1
GPSD,O=RMC 1264790545.000 0.005 -34.881633333 -56.163766667 ? ? ? 360.0000 0.000 ? ?
? ? 2
rGPSD,O=GGA 1264790545.000 0.005 -34.881633333 -56.163766667 57.600 ? 25.300
360.0000 0.000 ? ? ? 3
GPSD,R=1
$GPGSV,3,1,10,02,21,027,00,04,31,072,36,09,69,213,33,12,38,249,00*79
GPSD,Y=GSV ? 10:2 21 27 0 0:4 31 72 36 1:9 69 213 33 1:12 38 249 0 0:14 2 210 0 0:15 29 336
0 0:17 34 135 48 1:26 20 225 36 1:27 80 202 30 1:30 12 263 0 0:
r$GPGSA,A,3,,04,09,,,17,26,27,,,2.3,2.1,1.0*3B
GPSD,R=0
p
GPSD,P=-34.881633333 -56.163783333
```



Se le envió a GPSD el comando “i” para verificar que estaba detectando correctamente el receptor GPS. Luego se le envió el comando “w” (watch) para observar la salida procesada por GPSD. También se envió el comando “r” (raw) para observar la salida sin procesar y por último, se envió el comando “p” para obtener la posición.

4.10.3 Desarrollo del código de posicionamiento

Inicialmente se comenzó a desarrollar código para parsear las sentencias NMEA enviadas por el GPS. Sin embargo, al descubrir la existencia de GPS Daemon (GPSD), se decidió utilizar la información del GPS procesada por dicho programa.

Esta decisión se basó en que el estándar NMEA no está muy bien especificado y existen distintas implementaciones por lo que no es sencillo interpretar algunos campos de los mensajes NMEA.

El código del módulo de posicionamiento se divide, a grandes rasgos, en dos partes:

1. Obtención de las coordenadas de posición.
2. Conversión de las coordenadas del Datum WGS84 al Datum Yacaré.

En el “Anexo V – Código del módulo de posicionamiento” se adjunta el código desarrollado.

4.10.3.1 Obtención de coordenadas

Como se explicó en el capítulo 2, GPSD ofrece toda la información de posición a través del puerto TCP 2947. Por lo tanto, se debe establecer una conexión con ese puerto (socket) para obtener las coordenadas.

Antes que nada, es necesario aclarar que para trabajar con sockets hay que incluir la biblioteca `sys/socket.h`, la cual existe independientemente del sistema operativo. Esta biblioteca contiene las funciones necesarias para crear sockets, asignarles una dirección, establecer una conexión, etc.

En primera instancia, es necesario que la aplicación cliente cree un socket. Esto se hace invocando la función `int socket(family, type, protocol)`. En el caso de este proyecto, se



deseaba establecer una comunicación con un puerto TCP, por lo que correspondía la familia PF_INET, el tipo SOCK_STREAM y el protocolo IPPROTO_TCP.

Teniendo esto en cuenta, se invocó la función como se muestra a continuación:

```
socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)
```

Luego es necesario configurar el socket para que se permita reutilizar las direcciones locales. Esto se logra seteando la opción a nivel de socket SO_REUSEADDR. Para setear dicha opción se debe invocar la función *int setsockopt (int socket, int level, int optname, void *optval, size_t optlen)*.

En este caso, se invocó la función como se muestra a continuación:

```
setsockopt(gpsd, SOL_SOCKET, SO_REUSEADDR, (char *)&one, sizeof(one))
```

Donde **gpsd** es el socket que se creó con la función *socket*, como se explicó anteriormente; **SOL_SOCKET** especifica que se referirá a una opción a nivel de socket; **SO_REUSEADDR** es el nombre de la opción que se desea configurar y **one** es una variable de tipo *int* cuyo valor es uno.

El último paso para conectarse con GPSD es establecer la conexión entre el socket creado por la aplicación cliente y el socket de GPSD. Para esto se debe invocar la función *int connect(int socket, struct sockaddr *addr, size_t length)*, la cual inicia una conexión desde el socket cuyo descriptor de archivo es *socket* hacia el socket cuya dirección se especifica a través de la estructura *addr* con largo *length*.

Para pasárselo a este método la dirección del socket del servidor (GPSD) es necesario utilizar una estructura *sockaddr*. En la práctica, la dirección se encuentra en una estructura de algún otro tipo apropiado al formato de dirección que se esté utilizando, pero luego se castea a *struct sockaddr ** cuando se le pasa como parámetro a la función *connect*.

En el caso de este proyecto, se establece una conexión TCP/IP, por lo que se trabaja en el espacio de nombres de Internet (familia de protocolos PF_INET). En este espacio de nombres, una dirección de socket consiste en una dirección de host y un puerto en ese host. Para representar direcciones de socket en el espacio de nombres de Internet se utiliza el tipo de datos **struct sockaddr_in**.



En el caso de este proyecto, la función connect se invocó como se muestra a continuación:

```
connect(gpsd, (struct sockaddr *) &sin, sizeof(sin))
```

Donde **gpsd** es el socket que se creó con la función socket y **sin** es una estructura que contiene la dirección IP y el puerto con el cual se desea establecer la conexión.

Una vez establecida la conexión es posible transferir datos utilizando las funciones *write* y *read*. Para obtener las coordenadas de posición del móvil por primera vez, es necesario que GPSD ingrese en el modo *Watch*, por lo que se envía el comando “w” como se muestra a continuación:

```
write(gpsd, "w\n", strlen("w\n"))
```

Donde **gpsd** es el socket, “**w\n**” es el comando que se desea enviar (\n es el salto de página y actúa como la tecla Enter, sirviendo de confirmación del comando) y, por último, **strlen(“w\n”)** es el largo del comando.

Aunque no se vaya a utilizar, es necesario leer la salida de este comando para que no quede en el buffer y luego se puedan leer correctamente los datos que se habrán de utilizar. Esto se logra invocando a la función *read* como se muestra a continuación:

```
read(gpsd, descarte, sizeof(descarte))
```

Donde **gpsd** es el socket y **descarte** es un variable auxiliar en la que se almacena la salida que se desea descartar.

Luego, se vuelve a enviar el comando “w” para salir del modo Watch.

Por último, para solicitar la posición, es necesario enviar el comando “p”, el cual devolverá la posición en el formato “GPSD, P=%f %f”, con los números en grados y la latitud en primer lugar. Se envía este comando invocando la función *write* de forma análoga a como se explicó anteriormente.

Luego de enviar el comando “p”, se invoca la función *read* y se guarda lo leído en una variable para poder analizar la salida del comando.

Si GPSD aún no dispone de una lectura de posición, la salida será “GPSD,P=?”. En ese caso, se esperan 0.4 segundos y se vuelve a intentar. Si luego de 5 veces de reintentar,



aún no se dispone de una lectura de posición, la función termina informando de la situación.

Si se logró obtener una lectura de posición, se procede a desglosar la cadena de caracteres para poder almacenar la latitud y la longitud en variables separadas. Este desglose es posible a través de varias invocaciones a la función `char * strtok (char *newstring, const char *delimiters)`, lo cual permite dividir una cadena de caracteres en tokens. El parámetro `delimiters` especifica el conjunto de caracteres que rodean a los tokens que se desean extraer. En el primer llamado a la función `strtok`, el parámetro `newstring` es la cadena de caracteres que se desea dividir. Este primer llamado a la función `strtok` devuelve el primer token. En los llamados subsiguientes se debe pasar `NULL` como primer parámetro, para indicar que se continuará trabajando con la misma cadena de caracteres.

A continuación se muestra la porción de código en la que se realiza la división de la cadena de caracteres que contiene las coordenadas de posición:

```
char *aux;
const char delimiters[] = " ,=";

aux = strtok(buf, delimiters);
aux = strtok(NULL, delimiters);
*latitude = strtod(strtok(NULL, delimiters),NULL);
*longitude = strtod(strtok(NULL, delimiters),NULL);
```

Como se puede observar, también se hace uso de la función `strtod` (“string to double”), la cual convierte la parte inicial de la cadena de caracteres que se pasa como parámetro en un número de punto flotante. Este número de punto flotante se devuelve como un valor de tipo `double`.

De esta forma se obtienen dos variables de tipo `double`, **latitude** y **longitude**, cargadas con los correspondientes valores de las coordenadas de posición del móvil.

4.10.3.2 Conversión de coordenadas al Datum Yacaré

Como se mencionó anteriormente, el receptor GPS entrega las coordenadas de posición en el Datum WGS84. Estas coordenadas serán utilizadas en GoogleMaps. Sin embargo, para calcular distancias es mejor disponer de coordenadas en unidades métricas, por lo que se decidió utilizar las coordenadas en el Datum Yacaré.



Para realizar esta conversión se obtuvo código de la aplicación Geotrans, producto de la Agencia Nacional de Inteligencia Geoespacial (NGA) de los Estados Unidos y del Centro de Investigación y Desarrollo del Ejército de los Estados Unidos, cuya licencia permite su reutilización. Fue necesario adaptar este código, ya que la aplicación Geotrans es completamente genérica, permitiendo realizar conversiones entre diversos Datums y sistemas de coordenadas.

El método que realiza la conversión en primera instancia setea los parámetros adecuados, para luego convertir las coordenadas geodésicas al sistema Mercator Tranverso.

En el primer paso se configuran los parámetros del elipsoide utilizado por el Datum Yacaré (Internacional 1924) y los parámetros de la proyección Mercator Transverso (proyección en la que se basa el Datum Yacaré). Es decir, se les da el valor adecuado a las variables correspondientes al semi-eje mayor y al achatamiento del elipsoide y a la latitud de origen, al meridiano central, al este falso, al norte falso y al factor de escala de la proyección. A partir de los valores anteriores se calculan otros parámetros del elipsoide y de la proyección, necesarios para realizar la conversión.

En el segundo paso se aplica un algoritmo matemático que realiza la conversión basándose en los parámetros configurados.

4.10.4 Compilación del código de posicionamiento

Debido a que en este código se utilizan funciones de la biblioteca math.h, es necesario enlazar dicha biblioteca al compilar. Por este motivo, para compilar el código se invocó la función *gcc* como se muestra a continuación:

```
gcc ProgramaGPS.c -o ProgGPS -lm
```

4.10.5 Testeo del módulo de posicionamiento en el PC de pruebas

Para testear el funcionamiento del código se programó un método main en el cual se invocan las funciones principales del código.



En primera instancia se verificó el funcionamiento del método que establece la conexión con GPSD y obtiene las coordenadas de posición del móvil.

Todas las funciones de la biblioteca sys/sockets.h que fueron utilizadas, devuelven un código de error en caso de no tener éxito en su ejecución. Por lo tanto, se introdujeron controles inmediatamente después de todas las invocaciones a dichas funciones, imprimiendo los errores que pudieran surgir.

También, como parte del testeo, en diversas partes del programa se imprimieron valores de variables significativas para controlar el correcto desarrollo del método.

Una vez que se logró el correcto funcionamiento de este método, se procedió a testear el método responsable de convertir las coordenadas del Datum WGS84 al Datum Yacaré. Para este testeo se obtuvieron a partir del receptor GPS, las coordenadas de posición del lugar de pruebas, tanto en un Datum como en el otro. Estos valores se utilizaron para comparar los resultados de la ejecución del programa y verificar que los mismos fueran correctos.

Para poder prescindir del receptor GPS durante este testeo, se modificó el método main, comentando la línea en la que se llama al método que se conecta a GPSD y obtiene la posición. En lugar de obtener las coordenadas de posición a partir del receptor GPS, éstas fueron pasadas de forma manual al método que realiza la conversión.

Se experimentaron diversas dificultades debido a que este método se obtuvo a partir de la adaptación de código de una aplicación de terceros. Por este motivo, inicialmente no se estaba familiarizado con el funcionamiento de dicho código, lo cual llevó a errores.

Un error que se detectó fue que se estaban utilizando las coordenadas geodésicas en grados, pero los métodos utilizados esperaban recibir los parámetros en radianes, por lo que fue necesario realizar la conversión entre grados y radianes.

También se detectaron errores debido al rango de ángulos donde debían estar contenidas las coordenadas. Inicialmente se realizaba la conversión a radianes controlando que el resultado estuviera entre 0 y 2π , pero luego se detectó que esto ocasionaba un error. Por lo tanto, se modificó el método de conversión a radianes para que el resultado estuviera entre $-\pi$ y π .



Una vez salvadas estas dificultades se logró obtener las coordenadas de posición del lugar de prueba en Datum WGS84 y convertirlas al Datum Yacaré.

4.10.6 Obtención de la posición del receptor GPS desde la placa BeagleBoard

Para obtener la posición del receptor GPS desde la placa BB se procedió a instalar GPSD de forma análoga a la que se explicó en la sección correspondiente a la instalación en el PC de pruebas, por lo tanto no se volverá a explicar ese procedimiento en esta sección.

Se experimentaron diversas dificultades en las etapas de detección del receptor GPS y ejecución de GPSD por lo que a continuación se detallan estas etapas.

4.10.6.1 Detección del receptor GPS en la placa BB

Se conectó el receptor GPS a la placa BB a través de un adaptador serie/USB. Si bien udev detectaba el dispositivo, no creaba el nodo correspondiente en el directorio /dev. Se ejecutó el comando `lsusb` para verificar que el adaptador serie/USB estuviera siendo reconocido y se obtuvo la siguiente salida:

```
root@beagleboard:~# lsusb
Bus 002 Device 005: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
Bus 002 Device 004: ID 07a6:8513 ADMtek, Inc. AN8513 Ethernet
Bus 002 Device 002: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Como se puede observar, el adaptador serie/USB es reconocido correctamente, “067b” es el código asociado al fabricante del adaptador (vendor) y “2303” es el código asociado al producto en sí (product). Sin embargo, al ejecutar el comando `dmesg` se pudo detectar que el driver correspondiente a este adaptador no estaba siendo cargado correctamente. La salida de este comando se muestra a continuación:

```
root@beagleboard:~# dmesg
...
usb 2-1.1: new full speed USB device using musb_hdrc and address 5
usb 2-1.1: device v067b p2303 is not supported
usbserial: disagrees about version of symbol module_layout
usbserial: disagrees about version of symbol module_layout
```



Como se puede observar, se obtiene un mensaje que indica que el adaptador serie/USB que se está utilizando no es soportado.

Para analizar en mayor detalle el problema, se configuró la prioridad de log de udev a modo debug ejecutando el comando `udevadm control --log-priority=debug`. A continuación se desconectó y se volvió a conectar el receptor GPS y se analizó el log abriendo el archivo `/var/log/syslog`. De esta forma se pudo detectar que udev encontraba un error al intentar cargar el módulo `pl2303`, el cual es necesario para manejar el conversor serie/USB.

El mensaje de error que udev recibía al intentar cargar dicho módulo era “`Invalid module format`”. Se investigó sobre este error y se tomó conocimiento de que suele surgir cuando se actualiza la ulmage del sistema operativo, sin actualizar los módulos. Sin embargo, ésta no era la situación, por lo que se concluyó que era un problema de compatibilidad entre el módulo y la versión del kernel que se estaba utilizando.

Para cargar el módulo, udev invoca el comando `modprobe` con la opción “`-b`”, la cual especifica que debe verificarse si el módulo a cargar está en la blacklist. Por lo tanto, se decidió agregar el módulo `pl2303` a la blacklist (agregando la línea “`blacklist pl2303`” al archivo `/etc/modprobe.d/blacklist.conf`), esperando que al encontrar este módulo en la blacklist, udev intentara cargar otro módulo con el cual pudiera controlar el dispositivo. Linux cuenta con un módulo `usbserial`, el cual es genérico y permite controlar el adaptador serie/USB que se está utilizando.

Udev continuaba intentando cargar el módulo, por lo que se sustituyó la palabra “`pl2303`” por el código completo del módulo, tal cual lo muestra udev. Es decir, en la blacklist se sustituyó la línea “`blacklist pl2303`” por “`blacklist usb:v067Bp2303d0100dcFFdsc00dp00icFFiscFFipFF`”.

Si bien se logró que udev no obtuviera un error al intentar cargar el módulo, no se logró solucionar el problema ya que udev interrumpe el procedimiento de reconocimiento del dispositivo al detectar que el módulo `pl2303` se encuentra en la blacklist, por lo que no llega a crear el nodo correspondiente en el directorio `/dev`.

Se procedió a analizar las reglas utilizadas por udev al detectar el dispositivo. Se encontró que uno de los archivos de reglas que udev ejecuta es `/lib/udev/rules.d/80-drivers.rules`, el cual contiene una línea genérica que aplica para diversos dispositivos. En



esta línea se especifica que debe ejecutarse `/sbin/modprobe $env{MODALIAS}`, es decir que el nombre del módulo que debe cargarse se obtiene de una variable de entorno.

Se decidió agregar en este archivo una línea que aplicara solo al adaptador serie/USB que se está utilizando y que especifique que se cargue el módulo `usbserial`. De esta forma, se agregó en el archivo `/lib/udev/rules.d/80-drivers.rules` la línea:

```
PHYSDEVBUS=="usb", PRODUCT=="067b/2303/0100", RUN+="/sbin/modprobe usbserial  
vendor=0x067b product=0x2303"
```

Con esto se logró que se cargara el módulo `usbserial` correctamente y que se creara el nodo correspondiente en el directorio `/dev`, es decir, que se creara el nodo `/dev/ttyUSB2`.

Para verificar que se estuviera recibiendo la información que envía el receptor GPS, se ejecutó `cat /dev/ttyUSB2`. No se obtuvo ningún dato en la salida y el proceso `cat` dejó de responder, por lo que fue necesario interrumpir la ejecución de este proceso.

Para descubrir la causa de este problema se volvió a analizar la salida del comando `dmesg` al detectar el dispositivo, la cual se muestra a continuación:

```
root@beagleboard:~# dmesg  
...  
usb 2-1.3: new full speed USB device using musb_hdrc and address 4  
usb 2-1.3: device v067b p2303 is not supported  
usbserial_generic 2-1.3:1.0: Generic device with no bulk out, not allowed.  
usbserial_generic: probe of 2-1.3:1.0 failed with error -5  
usbserial_generic 2-1.3:1.1: generic converter detected  
usb 2-1.3: generic converter now attached to ttyUSB2
```

Se le prestó especial atención a la tercer línea: “Generic device with no bulk out, not allowed”. Se tomó conocimiento de que este error podía deberse a que el módulo genérico `usbserial` no controla correctamente el adaptador serie/USB que se está utilizando y ésta podría ser la causa por la cual no se puede establecer la comunicación con el receptor GPS.

Por lo tanto, se procedió a buscar el módulo `pl2303` en un kernel genérico para compilarlo para la placa BB y probar si esto solucionaba el problema. Se siguieron obteniendo errores al cargar este módulo, por lo que se decidió volver a compilar el kernel con soporte a todos los módulos que tuvieran que ver con comunicaciones seriales sobre USB.



Para verificar si el problema había sido resuelto se dejó el archivo de reglas 80-drivers.rules de udev como estaba originalmente y se eliminó la línea que se había agregado en el archivo blacklist.conf.

A continuación se conectó el receptor GPS al adaptador serie/USB y a su vez, el adaptador se conectó a la placa BB. El adaptador serie/USB fue reconocido correctamente y se creó el nodo correspondiente en el directorio /dev.

Se utilizó la aplicación Minicom para configurar la comunicación serial con el receptor GPS. Luego se verificó que se estaban recibiendo correctamente los datos enviados por el GPS, lo cual se muestra en la siguiente pantalla:

```
root@beagleboard: ~
Welcome to minicom 2.3

OPTIONS: I18n
Compiled on Nov 16 2008, 16:31:55.
Port /dev/ttyUSB2

      Press CTRL-A Z for help on special keys

,f,3*15
$PGRMM,WGS 84*06
$GPBOD,,T,,M,,*47
$GPRTE,1,1,c,0*07
$GPRMC,121957,A,3454.248,S,05609.778,W,000.0,360.0,160310,007.2,W*6D
$GPRMB,A,,,,,,,,,V*71
$GPGGA,121957,3454.248,S,05609.778,W,1,07,1.1,57.2,M,10.0,M,,*78
$GPGSA,A,3,05,07,08,,10,15,17,,28,,,2.4,1.1,2.1*34
$GPGSV,3,1,09,05,31,301,41,07,17,098,41,08,45,124,38,09,02,257,00*79
$GPGSV,3,2,09,10,25,326,34,15,33,225,42,17,29,024,42,27,03,255,00*79
$GPGSV,3,3,09,28,73,146,43,,,*4A
$PGRME,4.1,M,8.3,M,9.2,M*2B
$GPGLL,3454.248,S,05609.778,W,122000,A*2E

CTRL-A Z for help | 4800 8N1 | NOR | Minicom 2.3 | VT102 | Offline
```

Figura 4.28 – Salida minicom



Como se puede observar, el nodo que corresponde al receptor GPS es /dev/ttyUSB2 y la conexión serial está configurada a 4800 baudios, con 8 bits de datos, sin bit de paridad y 1 bit de parada.

4.10.6.2 Ejecución de gpsd en la placa BB

A pesar de haber logrado que el receptor GPS fuera reconocido correctamente y que udev creara el nodo correspondiente en el directorio /dev, se observó que udev no ejecutaba GPSD al detectar el receptor GPS. Analizando el archivo /var/log/syslog se pudo ver que udev ejecutaba la regla correspondiente del archivo 40-gpsd.rules, invocando al script gpsd.hotplug.wrapper pero aún así GPSD no llegaba a ejecutarse. Se intentó ejecutar gpsd manualmente y se obtuvo el siguiente error:

```
root@beagleboard:~# gpsd -b -n -N -D 9 /dev/ttyUSB2
gpsd: launching (Version 2.38)
gpsd: Can't bind to port gpsd
gpsd: command socket create failed, netlib error -1
```

Este error suele presentarse cuando ya se está ejecutando una instancia de GPSD y, por lo tanto, el puerto que trata de abrir la nueva instancia ya está ocupado. Sin embargo, al invocar el comando *ps -ef | grep gpsd* no se obtuvo ningún resultado, lo cual significa que no existe otra instancia de GPSD ejecutándose.

Otra situación posible era que otra aplicación estuviera utilizando el puerto que GPSD intenta abrir (2947). Ejecutando el comando *netstat -na* fue posible ver que éste no era el caso.

Habiendo descartado las posibilidades anteriores, se analizó el código fuente de GPSD para determinar en qué interfaz intentaba abrir el puerto. Se descubrió que GPSD puede ejecutarse con la opción “-G”, la cual hace que el puerto 2947 esté disponible en todas las direcciones IP. Sin embargo, como GPSD no se estaba ejecutando con esta opción, intentaba abrir el puerto 2947 en la interfaz de loopback, la cual no se encontraba definida.

Por lo tanto, se invocó el comando “ifconfig lo up” con el cual se logró levantar la interfaz de loopback manualmente. Se volvió a intentar ejecutar GPSD y esta vez funcionó correctamente. Para verificar que GPSD estaba procesando los datos de posición de forma adecuada se ejecutó el comando “telnet localhost 2947” y se enviaron algunos



comandos de prueba. Se obtuvo respuesta de GPSD con datos de posición, lo cual permitió verificar que todo estaba funcionando correctamente.

Para lograr que la interfaz de loopback se habilite en el arranque y que no sea necesario cargarla manualmente, primero fue necesario definirla en el archivo /etc/network/interfaces. Se editó el mismo insertando las siguientes líneas al comienzo:

```
auto lo
iface lo inet loopback
```

Luego fue necesario crear un archivo en el directorio /etc/init.d que se encargara de levantar la interfaz. Se creó el archivo /etc/init.d/loopback con el siguiente contenido:

```
#!/bin/sh -e
#
# loopback - brings up the loopback (127.0.0.1) network device so that
#           DHCP and other such things will work
#
# Check the package is still installed
[ -x /sbin/ifup ] || exit 0

# Get LSB functions
. /lib/lsb/init-functions
. /etc/default/rcS

case "$1" in
    start)
        [ -d /var/run/network ] || mkdir /var/run/network

        log_begin_msg "Starting basic networking..."
        if ifup -v --allow auto lo; then
            if ifup -v auto lo; then
                log_end_msg 0
            else
                log_end_msg $?
            fi
        ;;
    stop)
        log_begin_msg "Stopping basic networking..."
        if ifdown lo; then
            log_end_msg 0
        else
            log_end_msg $?
        fi
        ;;
    restart|force-reload)
        exit 0
        ;;
    *)
        echo "Usage: /etc/init.d/loopback {start|stop|restart|force-reload}"
        exit 1
        ;;
esac
exit 0
```



Una vez hechos estos cambios se reinició la placa BB para verificar que la interfaz de loopback hubiera sido definida y habilitada correctamente. Una vez completado el proceso de arranque se invocó el comando “ifconfig”, el cual devolvió la siguiente salida:

```
root@beagleboard:~# ifconfig
eth0    Link encap:Ethernet HWaddr 00:60:6e:00:03:2b
        inet addr:192.168.1.253 Bcast:192.168.1.255 Mask:255.255.255.0
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:50 errors:0 dropped:0 overruns:0 frame:0
              TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:6521 (6.5 KB) TX bytes:6921 (6.9 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
              UP LOOPBACK RUNNING MTU:16436 Metric:1
              RX packets:10 errors:0 dropped:0 overruns:0 frame:0
              TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:924 (924.0 B) TX bytes:924 (924.0 B)
```

Como se puede observar, la interfaz de loopback se encuentra activa.

4.10.7 Construcción de biblioteca estática

Para que las funciones del módulo de posicionamiento pudieran ser llamadas desde el código de la interfaz Web, se elaboró una biblioteca estática a partir del código desarrollado.

Los pasos que se deben seguir para construir, compilar y utilizar librerías estáticas fueron explicados en el capítulo 2 de este documento.

Como primer paso se creó el archivo de cabecera *ProgGPS.h* contenido los macros *include*, los macros *define*, las definiciones de variables, las definiciones de estructuras y los prototipos de funciones que es necesario que tengan visibilidad global.

Luego, se creó el archivo fuente *ProgGPS.c* contenido los macros *define*, los macros *include*, las definiciones de variables y las definiciones de estructuras que deben tener visibilidad local. Además, se incluyó en este archivo la implementación de todas las funciones cuyos prototipos se incluyeron en el archivo de cabecera y la implementación de otras funciones auxiliares.



Por último se creó el archivo *pruebaGPS.c*, el cual utiliza el macro *include* para incluir el archivo de cabecera *ProgGPS.h* y contiene el método main que permitirá probar si la invocación de funciones de la biblioteca construida funciona correctamente.

Estos tres archivos fueron creados a partir del código del módulo de posicionamiento que ya se había desarrollado como fue explicado en las secciones anteriores.

Una vez creados los tres archivos se los colocó en el mismo directorio y se procedió con los siguientes pasos:

1. Obtener el archivo objeto del archivo fuente *ProgGPS.c*.

```
root@beagleboard:/home/ubuntu/UMix# gcc -c -o ProgGPS.o ProgGPS.c
```

2. Crear la biblioteca *libGPS.a* empaquetando el archivo objeto obtenido en el paso anterior.

```
root@beagleboard:/home/ubuntu/UMix# ar rcs libGPS.a ProgGPS.o
```

3. Obtener el archivo objeto del archivo fuente del programa de prueba (*pruebaGPS.c*), especificando que el archivo de cabecera que se incluyó se encuentra en el directorio actual.

```
root@beagleboard:/home/ubuntu/UMix# gcc -c -o pruebaGPS.o pruebaGPS.c -I./
```

4. Compilar el archivo objeto *pruebaGPS.o* indicando que la librería se encuentra en el directorio actual (-L./) y que se van a utilizar las librerías *libGPS.a* (-IGPS) y *math.h* (-lm).

```
root@beagleboard:/home/ubuntu/UMix# gcc -o pruebaGPS pruebaGPS.o -L./ -IGPS -lm
```

4.10.8 Testeo del módulo de posicionamiento en la placa BB

Para poder testear el módulo de posicionamiento en la placa BB, primero fue necesario copiar el archivo con el código en un directorio en la placa BB destinado al proyecto. Para realizar esta copia a través de la red, se utilizó el programa WinSCP.

WinSCP es un cliente abierto SFTP (Secure File Transfer Protocol) gráfico para Windows que emplea SSH (Secure Shell). Su función principal es facilitar la transferencia segura



de archivos entre dos sistemas informáticos, el sistema local y uno remoto que ofrezca servicios SSH.

Luego de copiar el archivo con el código, se compiló el código en la placa BB y se realizaron las pruebas correspondientes, las cuales se describen a continuación.

Se encendió el receptor GPS y se conectó a la placa BB a través del adaptador serie/USB. Se verificó con el comando `ps -ef | grep gpsd` que GPSD se estuviera ejecutando. Luego, se ejecutó el script de prueba (*pruebaGPS*), el cual imprimió en pantalla la información de hora y posición de la forma esperada.



4.11 Implementación del Módulo de Persistencia

Otro módulo que se consideró necesario para este proyecto, era uno que permitiera almacenar los puntos por los pase el móvil. De esta forma, una vez recorrida un área sería posible obtener las coordenadas de los puntos por los que se pasó, el momento en que se pasó por cada punto y si se detectaron supervivientes en ese punto.

Para implementar este módulo se pasó por dos etapas. En la primera se instaló LAMPP y se creó la base de datos en el PC de prueba. Luego se desarrolló el código de este módulo y se compiló y testeó en dicho PC. En la segunda etapa se importó la base de datos en el servidor definitivo y se realizaron pruebas accediendo a la base de datos desde la placa BB.

4.11.1 Diseño de la base de datos

La base de datos a utilizar por este módulo es muy sencilla dado que cuenta con una única tabla, la tabla Puntos, la cual tiene los campos que se explican a continuación:

- *Identificador*

Es un campo del tipo *int* (entero) que representa la clave primaria de la tabla, es decir que identifica de forma única a cada punto. Se configuró como AUTO INCREMENT por lo que no es necesario determinar su valor al insertar un registro, sino que el motor de base de datos se encargará de asignarle el valor que corresponda.

- *Latitud*

Es un campo de tipo *double* que contiene la coordenada de latitud del punto. Si el número es negativo quiere decir que se refiere a una posición al sur del Ecuador.

- *Longitud*

Es un campo de tipo *double* que contiene la coordenada de longitud del punto. Si el número es negativo quiere decir que se refiere a una posición al oeste del meridiano de Greenwich.

- *Timestamp*

Es un campo de tipo *datetime* que contiene la fecha y la hora en que el móvil pasó por ese punto. El formato de este campo es AAAA-MM-DD hh:mm:ss. Por ejemplo, 21 de diciembre de 2010 a las 21 horas sería 2010-12-21 21:00:00.



- *Encontro*

Es un campo del tipo *tinyint* que vale 1 si se detectó un superviviente en ese punto o 0 si no se detectó nada.

4.11.2 Instalación de LAMPP en el PC de pruebas

Para simplificar la instalación del servidor MySQL y la creación de la base de datos en el PC de prueba, se decidió utilizar LAMPP. LAMPP, además de proveer un servidor MySQL, también provee un componente denominado phpMyAdmin que ofrece una interfaz gráfica para crear y administrar bases de datos de forma sencilla.

La instalación de LAMPP consiste simplemente en descargar un archivo .tar.gz de la página www.apachefriends.org y descomprimirlo en el directorio /opt. Puede ser necesario modificar algún archivo de configuración si se desea personalizar el comportamiento de LAMPP, pero en el caso de este proyecto no fue necesario.

4.11.3 Creación de la base de datos en el PC de pruebas

Aprovechando la facilidad para crear bases de datos que provee el componente phpMyAdmin, se utilizó este componente para crear la base de datos UMix en el PC de pruebas.

Para acceder a phpMyAdmin es necesario ingresar la URL: <http://localhost/phpmyadmin/> en la barra de dirección de un navegador Web. Se podrá ver una pantalla similar a la que se muestra en la Figura 4.29.

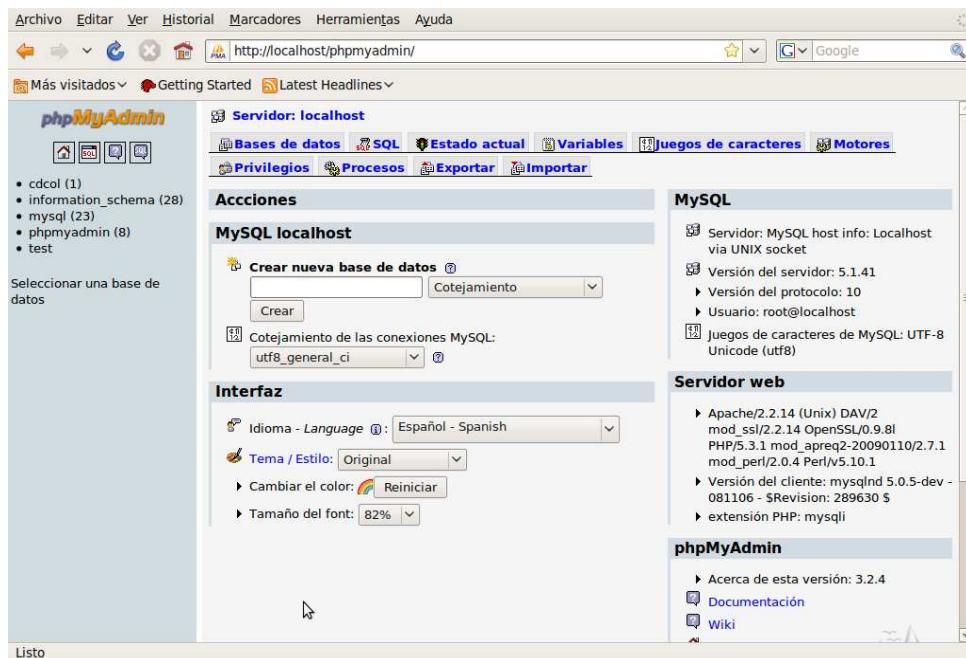


Figura 4.29 - Pantalla principal phpMyAdmin

4.11.4 Obtención del script de creación de la base de datos

Para poder importar la base de datos creada al servidor definitivo de forma sencilla, es necesario obtener el script de creación de la base de datos. Esta opción está disponible en el tab “Exportar” de la página principal de la interfaz Web de phpMyAdmin. Teniendo en cuenta que solo se desea exportar la estructura de la base de datos, al exportar la misma se deshabilitó la sección “Datos”.

En el “Anexo VI – Procedimientos de LAMPP” se adjunta el script que se obtuvo.

4.11.5 Creación de usuario en el PC de pruebas

Por cuestiones de seguridad, para no tener que utilizar el usuario root, fue necesario crear un usuario para trabajar sobre la tabla del proyecto.

Accediendo al tab “Privilegios” en la pantalla principal es posible agregar un nuevo usuario con privilegios específicos para trabajar sobre una base de datos en particular.

En el caso de este proyecto se creó el usuario UMixUser, que va a trabajar desde el servidor local y cuya contraseña es “umixpass”. No se le asignan privilegios globales sino que se le asignaron únicamente privilegios específicos para trabajar sobre la base de datos del proyecto. *Nota: El usuario definitivo se creó con permisos para acceder desde cualquier servidor.*



Teniendo en cuenta que este usuario es creado para acceder a la base de datos desde la aplicación desarrollada para este proyecto, el usuario solo necesitará privilegios para hacer consultas y modificaciones de datos. No necesitará privilegios para hacer cambios en la estructura ni necesitará privilegios para administrar la base de datos. Por lo tanto, solo se le otorgan privilegios para SELECT, INSERT, UPDATE y DELETE.

En el Anexo VI – Procedimientos en LAMPP se explican con mayor detalle los procedimientos para instalar LAMPP, crear la base de datos y el usuario y obtener el script de creación de la base de datos utilizando LAMPP.

4.11.6 Creación de la base de datos en el servidor definitivo

Para simplificar la creación de la base de datos en el servidor definitivo, se decidió exportar el script del PC de pruebas para luego importarlo en el servidor definitivo. El procedimiento de exportar el script de creación de la base de datos ya fue explicado en la sección “Obtención del script de creación de la base de datos”. Para poder importar dicho script es necesario ejecutar un cliente MySQL que se conecte al servidor MySQL que fue previamente instalado.

Para poder ejecutar el cliente MySQL es necesario estar loggeado con el usuario root, para lo cual se ejecutó el comando *su*. Luego se invocó al comando *mysql* como se muestra a continuación:

```
root@ubuntu-pc:/home/ssh-user# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.1.37-1ubuntu5.1 (Ubuntu)
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Para ejecutar el script que se exportó de la base de datos creada en el PC de pruebas se debe ejecutar el comando *source* en el cliente MySQL como se muestra a continuación:

```
mysql> source /home/ssh-user/BaseDeDatos/scriptUMix.sql
Query OK, 0 rows affected (0,00 sec)
Query OK, 1 row affected (0,02 sec)
Database changed
Query OK, 0 rows affected (0,00 sec)
```



Para verificar que la base de datos se creó correctamente se utilizó el comando **SHOW DATABASES**:

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| UMix          |
| mysql          |
| test           |
+-----+
4 rows in set (0,00 sec)
```

Como se puede observar, la base de datos UMix fue creada correctamente. Sin embargo, también se desea verificar que la tabla Puntos haya sido creada y que tenga la estructura correcta.

En primera instancia, se seleccionó la base de datos UMix, utilizando el comando **USE**:

```
mysql> USE UMix;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

Para verificar que la tabla Puntos se creó correctamente, se utilizó el comando **SHOW TABLES**:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_UMix |
+-----+
| Puntos          |
+-----+
1 row in set (0,00 sec)
```

Para verificar que la tabla Puntos tiene la estructura correcta se utilizó el comando **DESCRIBE**.

```
mysql> DESCRIBE Puntos;
+-----+-----+-----+-----+-----+
| Field    | Type     | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| Identificador | int(11) | NO  | PRI | NULL   | auto_increment |
| Latitud    | double   | NO  |     | NULL    |             |
| Longitud   | double   | NO  |     | NULL    |             |
| Timestamp  | datetime | NO  |     | NULL    |             |
| Encontro   | tinyint(4)| NO  |     | NULL    |             |
+-----+-----+-----+-----+-----+
5 rows in set (0,02 sec)
```



4.11.7 Creación de usuario en el servidor definitivo

Para crear el usuario con el cual se habrá de establecer la conexión se ejecutaron las siguientes sentencias SQL:

```
CREATE USER 'UMixUser'@'%' IDENTIFIED BY 'umixpass';
GRANT USAGE ON * . * TO 'UMixUser'@'%' IDENTIFIED BY 'umixpass' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0 ;
GRANT SELECT , INSERT , UPDATE , DELETE ON `UMix` . * TO 'UMixUser'@'%';
```

Como se puede observar, en primer lugar se crea el usuario. Como se mencionó anteriormente, el usuario definitivo puede acceder desde cualquier servidor ('UMixUser'@'%'). Luego se establecen el máximo de consultas por hora, el máximo de conexiones por hora, el máximo de actualizaciones por hora y el máximo de conexiones de usuario en cero. El hecho de setear estos parámetros en cero implica que no hay límite. Por último se le otorgan al usuario los permisos para hacer consultas, insertar, actualizar y borrar registros de cualquier tabla de la base de datos UMix.

4.11.8 Integración del módulo de persistencia^{xxxiii, xxxiv, xxxv}

La base de datos del proyecto se integra con el módulo de posicionamiento y con la interfaz Web del proyecto. Para integrar la base de datos con el módulo de posicionamiento fue necesario desarrollar código en lenguaje C que pudiera interactuar con la base de datos y con dicho módulo. La integración de la base de datos con la interfaz Web se explica en las secciones correspondientes al desarrollo de dicha interfaz.

Inicialmente se desarrollaron funciones para almacenar puntos en la base de datos, obtener puntos en los que se detectaron sobrevivientes y eliminar puntos de la base de datos. Sin embargo, durante el desarrollo de la interfaz Web se detectó que era más sencillo y eficiente que las tareas de obtención de puntos almacenados y de eliminación de puntos fueran realizadas directamente desde la interfaz Web.

4.11.8.1 Desarrollo del código del módulo de persistencia

Para desarrollar el código de este módulo se comenzó por investigar la API C proporcionada por MySQL. Los resultados de esta investigación se resumen en la sección correspondiente del capítulo 2.



Para manejar de forma sencilla los datos de los puntos se definieron dos estructuras:

1. La estructura `Point_Data` representa un punto y contiene cuatro variables, dos de ellas de tipo `double`, `latitude` y `longitude`, una variable de tipo `string`, `datetime`, y una variable de tipo `int`, `found`.
2. La estructura `Point_Data_Array` representa un arreglo de puntos y contiene dos variables: un array de estructuras `Point_Data`, `pointsArray`, y una variable de tipo `size_t`, `length` cuyo valor es la cantidad de puntos almacenados en `pointsArray`.

La definición de esta última estructura fue necesaria debido a que en el lenguaje C no es posible obtener la cantidad de elementos que tiene un arreglo, si éste no fue inicializado dentro de la misma función en la que se quiere conocer su tamaño. Es decir que si una función recibe el arreglo como parámetro, no es posible conocer la cantidad de elementos que éste contiene, a menos que esta cantidad sea pasada como un parámetro más de la función o bien esté contenida dentro de cierta estructura.

Se declararon como variables locales del programa arreglos conteniendo la información necesaria para establecer la conexión con el servidor de bases de datos, es decir, host, usuario, password y base de datos.

Luego se definieron e implementaron las siguientes funciones:

- `void setServer()`

Esta función es invocada antes de cualquier interacción con la base de datos. Se encarga de leer un archivo para obtener la dirección del servidor de base de datos y cargarle el valor leído a la variable `server`.

- `Point_Data_Array getSurvivorPoints()`

Esta función no recibe parámetros. Ejecuta una consulta a la base de datos y devuelve una estructura conteniendo un arreglo con todos los puntos en los que se detectaron sobrevivientes y la cantidad de puntos contenida en dicho arreglo.

- `Point_Data loadPointStruct(double latitude, double longitude, char *datetime, int found)`

Dados los datos de un punto, los carga en una estructura `Point_Data` y devuelve la misma.



- void storePoint(Point_Data point)

Dada una estructura Point_Data, almacena la información del punto correspondiente en la base de datos.

- int deleteAllPoints()

Esta función no recibe parámetros. Elimina todos los puntos almacenados en la base de datos.

- void recorrerArrayPuntos(Point_Data_Array points)

Dada una estructura Point_Data_Array, recorre el arreglo de puntos que ésta contiene, imprimiendo la información de cada uno de los puntos. Esta función se implementó para testear la función getSurvivorPoints() y asegurarse de que ésta carga los datos correctamente.

Llegado el momento de la integración de todos los módulos, solo se utilizaron las funciones *storePoint* y *setServer* debido a que, como se explicó anteriormente, las otras funciones fueron implementadas directamente en el código de la interfaz Web. En la sección “Desarrollo de la interfaz Web” se presentan diagramas explicando el comportamiento de la interfaz Web, incluyendo los accesos a la base de datos.

En todas las funciones que se implementaron para este módulo se realizó un control de errores, chequeando el valor de retorno de todas las funciones de la API C de MySQL que fueron invocadas. En caso de surgir un error, se imprime en pantalla el mensaje correspondiente.

En el Anexo VIII - Código del módulo de persistencia se adjunta el código desarrollado para este módulo.

4.11.8.2 Compilación del código del módulo de persistencia

MySQL trae un script especial denominado *mysql_config*, el cual provee información útil para compilar el código desarrollado para conectarse a la base de datos MySQL e interactuar con ella.

Si se invoca dicho script con la opción **--libs** es posible obtener las bibliotecas y opciones que es necesario enlazar con el programa. Si se invoca el script *mysql_config* con la



opción **--cflags** es posible obtener los flags que necesita el compilador para encontrar los archivos que se incluyen en el código.

La información obtenida de invocar el script *mysql_config* es necesario pasársela a *gcc*. Por lo tanto, para compilar el código desarrollado, se invocó el comando *gcc* como se muestra a continuación:

```
gcc -o UMixDBAccess.o $(mysql_config --cflags) UMixDBAccess.c $(mysql_config --libs)
```

4.11.8.3 Testeo del funcionamiento del código en el PC de pruebas

El testeo de este módulo se realizó en forma similar al testeo del módulo de posicionamiento. Se elaboró un método *main()* en el que se invocan las funciones implementadas, imprimiendo sus salidas para verificar que las funciones se comportan como es esperado.

Este testeo primero se realizó en el PC de pruebas para luego realizarlo en la placa BB.

A continuación se muestra el método *main()* elaborado:

```
void main(){

    double lat = -33;
    double lon = -57.4;
    char *fechahora = "2010-02-20 21:00:00";
    int flag = 1;

    Point_Data testPoint;
    testPoint = loadPointStruct(lat, lon, fechahora, flag);
    storePoint(testPoint);
    Point_Data_Array points;
    points = getSurvivorPoints();
    int deleted = deleteAllPoints();
    printf("Se borraron %d puntos\n", deleted);
    recorrerArrayPuntos(points);
}
```

A continuación se explican las líneas anteriores:

1. Se declaran e inicializan variables que contienen los datos de un punto.
2. Se declara un *Point_Data* *testPoint*.
3. Se inicializa *testPoint* asignándole la salida de la función *loadPointStruct*.
4. Se invoca a la función *storePoint* para almacenar el punto en la base de datos.
5. Se declara un *Point_Data_Array* *points*.



6. Se inicializa *points* asignándole la salida de la función *getSurvivorPoints*. Esta función va imprimiendo en pantalla los datos de todos los puntos que devuelve la consulta. Debido a que el flag del punto que se insertó en el paso 4 vale 1, la información de este punto debería imprimirse en este paso (además de otros puntos en la base de datos cuyo flag valga 1). Esto permite verificar que la inserción en la base de datos fue exitosa.
7. Se declara e inicializa una variable int *deleted* cuyo valor es la cantidad de filas borradas por la función *deleteAllPoints*.
8. Se imprime un mensaje con la cantidad de puntos borrados.
9. Se invoca a la función *recorrerArrayPuntos*, la cual va imprimiendo los datos de cada punto contenido en la estructura *points* que recibe como parámetro. Como se mencionó anteriormente, esta función fue elaborada con el único fin de verificar que la función *getSurvivorPoints* carga la estructura Point_Data_Array correctamente.

Una vez compilado el código, se procedió a ejecutarlo para verificar que funcionara correctamente. Como era de esperar, no se logró que funcionara en la primera prueba. Al intentar ejecutarlo se obtuvo el error: “*Error en connect: Can't connect to local MySQL server through socket '/var/run/mysql/mysqld.sock' (2)*”.

Se investigó sobre el socket utilizado por el servidor MySQL que se instala con LAMPP y se tomó conocimiento de que éste utiliza /opt/lampp/var/mysql/mysql.sock y no /var/run/mysql/mysqld.sock. Este último es el socket que utiliza el demonio mysql (mysqld), el cual es el servidor MySQL por defecto.

Para solucionar este problema se creó el directorio /var/run/mysqld y luego se creó un enlace simbólico desde /var/run/mysqld/mysqld.sock apuntando a /opt/lampp/var/mysql/mysql.sock de la forma en que se muestra a continuación:

```
root@ubuntu:/home/sofia/Escritorio/BD# mkdir /var/run/mysqld
root@ubuntu:/home/sofia/Escritorio/BD# ln -s /opt/lampp/var/mysql/mysql.sock
/var/run/mysqld/mysqld.sock
root@ubuntu:/home/sofia/Escritorio/BD# ls -l /var/run/mysqld
total 0
lrwxrwxrwx 1 root root 31 2010-02-22 17:10 mysqld.sock -> /opt/lampp/var/mysql/mysql.sock
```

Al volver a ejecutar el programa, éste logró establecer la conexión con el servidor MySQL correctamente. Sin embargo, al reiniciar el sistema se notó que el archivo mysqld.sock creado ya no existía. Por lo tanto, se decidió que esta no era una buena solución.



Era necesario determinar en dónde se especifica el path donde buscar el socket, por lo tanto se procedió a investigar la función que se utiliza para establecer la conexión con el servidor MySQL (*mysql_real_connect*). Esta función ya fue explicada en el capítulo 2. Sin embargo, en esta sección es importante repetir que el parámetro *const char *unix_socket* especifica el socket que se debe utilizar para establecer la conexión con el servidor MySQL.

Hasta el momento, este parámetro se pasaba como NULL, por lo que el sistema utilizaba el socket por defecto (/var/run/mysqld/mysqld.sock). Se solucionó el error que surgía al intentar conectarse al servidor MySQL pasando en el parámetro *unix_socket*, el path del socket utilizado por LAMPP (/opt/lampp/var/mysql/mysql.sock).

Durante el testeo también surgieron errores relacionados a la declaración e inicialización de arreglos. Estos se debieron a la poca experiencia que se poseía en el uso de lenguajes en los que es necesario administrar la memoria, como es el caso del lenguaje C.

Luego de investigar el uso de las funciones necesarias para reservar, asignar y liberar memoria, fue posible solucionar dichos errores. En el capítulo 2 se resumen los resultados de la investigación sobre este tema.

4.11.8.4 Migración del módulo de persistencia a la placa BB

Para poder integrar el módulo de persistencia con el resto del sistema se procedió de forma similar a la migración del módulo de posicionamiento. Primero se copió el archivo con el código de este módulo en un directorio de la placa BB utilizando WinSCP. Luego se compiló el código en la placa BB y se realizaron las pruebas correspondientes, las cuales se describen en la sección “Testeo del funcionamiento del código en la placa BB” correspondiente a este módulo.

Fue necesario modificar el código para que se establezca una conexión con el servidor MySQL remoto y no con un servidor MySQL local. Además, fue necesario modificar el parámetro que establece el socket que se utiliza para la conexión. A este parámetro se le dio valor “NULL” ya que para establecer una conexión con un servidor MySQL remoto se utiliza un puerto TCP/IP y no un socket de Unix.



En resumen, se modificó el valor de la variable `server` asignándole la dirección IP del servidor MySQL y, en todas las llamadas al método `mysql_real_connect` de la API C de MySQL, se sustituyó el parámetro correspondiente al socket por el valor “NULL”.

4.11.8.5 Testeo del funcionamiento del código en la placa BB

Debido a que ya se habían realizado testeos en el PC de pruebas, se sabía que el código funcionaba correctamente al establecer una conexión con un servidor MySQL local. Sin embargo, era necesario verificar que la conexión con un servidor MySQL remoto también funcionaba.

En principio la conexión con el servidor MySQL remoto desde la placa BB no funcionó. Al intentar ejecutar el programa compilado se obtenía el mensaje que se muestra a continuación:

```
root@beagleboard:/home/ubuntu/BaseDeDatos# ./UMixDBAccess.o
Error en connect: Can't connect to MySQL server on '192.168.1.100' (111)
```

Para solucionar este problema se investigó sobre el establecimiento de conexiones entre un cliente y un servidor MySQL remoto. Investigando en Internet respecto de este tema, se tomó conocimiento de que era necesario modificar el valor del parámetro “bind-address” del servidor MySQL. Este parámetro por defecto está configurado como “localhost”, pero para permitir conexiones desde clientes remotos es necesario que su valor sea la dirección IP del servidor MySQL.

Se editó el archivo `/etc/mysql/my.cnf` en el servidor MySQL y se sustituyó la línea:

```
bind-address = localhost
```

Por la línea:

```
bind-address = 192.168.1.100
```

Luego, también se tomó conocimiento de que si la dirección IP del servidor MySQL es dinámica, es posible comentar la línea correspondiente a la bind-address y por defecto se configurará en la dirección actual del servidor MySQL.

Una vez realizada esta modificación se pudo ejecutar el programa compilado, el cual devolvió los resultados esperados.



4.11.8.6 Construcción de biblioteca estática

Siguiendo un procedimiento análogo al que se explicó para crear la biblioteca *libGPS.a*, se creó la biblioteca *libDB.a*.

Se creó el archivo de cabecera *DBAccess.h* conteniendo los macros *include*, las definiciones de estructuras y los prototipos de funciones que es necesario que tengan visibilidad global. Es importante aclarar que se utilizó un macro *include* para incluir el archivo de cabecera *ProgGPS.h* debido a que el código de acceso a la base de datos utiliza la estructura *Point_Data_Structure*, la cual se define en el mencionado archivo.

Luego se creó el archivo fuente *DBAccess.c* conteniendo los macros *include*, las definiciones de variables y las definiciones de estructuras que deben tener visibilidad local. Además, se incluyó en este archivo la implementación de todas las funciones cuyos prototipos se incluyeron en el archivo de cabecera.

Por último se creó el archivo *pruebaDB.c*, el cual utiliza el macro *include* para incluir el archivo de cabecera *DBAccess.h* y contiene el método *main* que permitirá probar si la invocación de funciones de la biblioteca construida funciona correctamente.

Estos tres archivos fueron creados a partir del código del módulo de respaldo que ya se había desarrollado como fue explicado en las secciones anteriores. Una vez creados los tres archivos se los colocó en el mismo directorio en donde estaban los archivos correspondientes al módulo de posicionamiento y se procedió con los siguientes pasos:

1. Obtener el archivo objeto del archivo fuente *DBAccess.c*.

```
root@beagleboard:/home/ubuntu/UMix# gcc -c -o DBAccess.o $(mysql_config --cflags)  
DBAccess.c $(mysql_config --libs)
```

2. Crear la biblioteca *libDB.a* empaquetando el archivo objeto obtenido en el paso anterior.

```
root@beagleboard:/home/ubuntu/UMix# ar rcs libDB.a DBAccess.o
```

3. Obtener el archivo objeto del archivo fuente del programa de prueba (*pruebaDB.c*), especificando que el archivo de cabecera que se incluyó se encuentra en el directorio actual.

```
root@beagleboard:/home/ubuntu/UMix# gcc -c -o pruebaDB.o $(mysql_config --cflags)  
pruebaDB.c $(mysql_config --libs) -I./
```



4. Compilar el archivo objeto *pruebaDB.o* indicando que la librería se encuentra en el directorio actual (-L./) y que se van a utilizar las librerías libDB.a (-IDB) y math.h (-Im).

```
root@beagleboard:/home/ubuntu/UMix# gcc -o pruebaDB $(mysql_config --cflags) pruebaDB.o $(mysql_config --libs) -L./ -IDB -Im
```

Nota: El motivo de que sea necesario utilizar los parámetros `$(mysql_config --cflags)` y `$(mysql_config --libs)` se explicó en la sección “Compilación del código del módulo de ”.



4.12 Diseño de la fuente de alimentación

Como se mencionó en el capítulo 2 de este documento, para obtener los voltajes requeridos por los componentes de los circuitos electrónicos diseñados para este proyecto, se utilizó una fuente de continua de 12 V y reguladores de voltaje de la familia LM78XX.

Se diseñó la fuente de alimentación de forma de que las entradas V_S de los drivers de los motores (L293D) (voltaje de alimentación para la etapa de salida de potencia) se conectarán directamente a la fuente de 12 V; la alimentación de 9 V requerida por el sensor IR se obtuviera de la salida de un regulador LM7809 y la alimentación de 5 V requerida por el PIC, los sensores de obstáculos, la alimentación de la lógica de los drivers de los motores, la placa BeagleBoard y el hub USB, se obtuviera de la salida de un regulador LM7805.

A continuación se muestra el circuito de la fuente de alimentación:

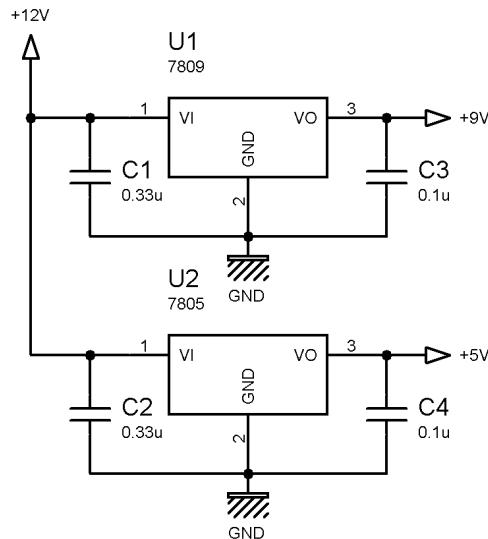


Figura 4.30 - Fuente de alimentación

Donde +12V representa el borne positivo de la fuente y GND, el borne negativo.

Para que el vehículo fuera inalámbrico, inicialmente se pretendía utilizar una batería de 12 V y 1.2 A, sin embargo, esta intensidad de corriente no era suficiente. Averiguando en el mercado local se tomó conocimiento de que el valor siguiente de intensidad de



corriente que se puede conseguir es de 4 A. Desafortunadamente, las dimensiones y el peso de estas baterías exceden las capacidades del vehículo que se construyó como prototipo de este proyecto.

Por lo tanto, se decidió que a los efectos de alimentar el prototipo se utilizaría una fuente de computadora. Si este proyecto se llevara a cabo fuera del ámbito académico, podría construirse un vehículo con las proporciones y los motores que permitieran cargar con la batería necesaria.

Luego de realizar algunas pruebas, se detectó que el regulador de 5 V estaba levantando mucha temperatura provocando que el sistema se comportara de forma aleatoria e incluso provocando que la placa BB se reiniciara en algunas ocasiones. Este exceso de temperatura en el regulador se debía a que su corriente máxima es de 1 A. Como se mencionó anteriormente, el PIC, la placa BB, el hub USB y la mayoría de los integrados se alimentan con 5 V, por lo que la corriente que se debe obtener del regulador de 5 V excede la corriente máxima del mismo.

Debido a esto y teniendo en cuenta que al momento de detectar esta dificultad no se contaba con el tiempo suficiente para realizar grandes modificaciones, se decidió que la alimentación de 5 V sería obtenida, al igual que la alimentación de 12 V, de la fuente de computadora.



4.13 *Compilación kernel Linux OMAP*

Si bien inicialmente se había compilado e instalado el kernel ARM EABI Ubuntu 9.04, en la etapa final del proyecto se decidió utilizar el kernel Linux OMAP. A continuación se explican los motivos por los cuales se tomó esta decisión, el impacto de este cambio y el procedimiento que se llevó a cabo.

4.13.1 Análisis del cambio de kernel

4.13.1.1 Justificación

El principal motivo por el cual se decidió aplicar un cambio de kernel en el proyecto fue la necesidad de disponer de las cabeceras del kernel para poder desarrollar drivers que permitan un acceso y un uso más simple y eficiente de los pines del puerto GPIO de la placa.

Por algún motivo las cabeceras del kernel nativo con el que se compiló el sistema operativo de la placa no estaban disponibles en Internet.

El segundo motivo por el cual se realizó el cambio fue que el kernel original incorporaba muchas funcionalidades que no eran necesarias y por lo tanto generaban un consumo de recursos adicional que reducía el rendimiento del sistema. La compilación de un nuevo kernel permitió seleccionar únicamente las características que eran indispensables para el funcionamiento del sistema.

4.13.1.2 Impacto

Es importante tener en cuenta que un cambio de estas características puede llevar a incompatibilidades con el trabajo ya realizado. Por este motivo, en una primera instancia el equipo de trabajo fue reticente a realizar este cambio teniendo en cuenta el tiempo que se podía perder en adaptaciones.

Sin embargo, luego de comprobar que el nuevo kernel estaba homologado por el equipo de desarrolladores de Debian y que la única diferencia con el anterior radicaba en la inclusión de nuevas funcionalidades y un código más adaptado al procesador y plataforma utilizados, se optó por intentar el cambio (haciendo los respaldos pertinentes).



4.13.2 Configuración del entorno

4.13.2.1 Fundamento

Para generar un kernel de Linux existen tres opciones. La primera es disponer en la máquina de desarrollo de un compilador y otra serie de herramientas compatibles con la plataforma donde se piensa utilizar el kernel, comúnmente conocidas como **cross-toolchain**. A esta opción se la conoce como compilación cruzada, ya que se compila el kernel en la máquina de desarrollo y luego se transfiere a la plataforma objetivo. La segunda posibilidad es compilar el kernel directamente en la plataforma donde se va a utilizar. Y, por último, existe la posibilidad de simular la plataforma objetivo en la máquina de desarrollo (como puede ser mediante la herramienta **qemu**).

Las últimas dos opciones mencionadas llevan mucho más tiempo que la primera, por lo que se optó por realizar una compilación cruzada.

4.13.2.2 Instalación de emdebian

Existen infinidad de cross-toolchains que permiten compilar kernels de Linux para un procesador ARM. Las más conocidas son las que ofrecen **Codesourcery** y **Emdebian**. La primera tiene el inconveniente de ser paga. Por lo tanto, finalmente se optó por utilizar la que provee **Emdebian** bajo licencia GPL.

A partir de una PC con Ubuntu, la instalación se realiza mediante los pasos que se enumeran a continuación:

1. Instalación de paquetes previos

Para instalar los paquetes requeridos se utilizó el comando *apt-get install* como se muestra a continuación:

```
$sudo apt-get install git-core kernel-package fakeroot build-essential curl libncurses-dev u-boot-mkimage
```

2. Instalación de los repositorios

Para configurar apt-get para que utilice los repositorios adecuados, se agregaron las siguientes líneas en el archivo **/etc/apt/sources.list**:

```
#debian embedded
deb http://www.emdebian.org/debian/ lenny main
```



3. Instalación de los compiladores

La instalación de los compiladores consiste simplemente en ejecutar los siguientes comandos en una terminal:

```
$sudo apt-get update  
$sudo apt-get install cpp-4.3-arm-linux-gnu g++-4.3-arm-linux-gnu gcc-4.3-arm-linux-gnu
```

4.13.3 Compilación

El proceso de compilación consiste en la obtención del código fuente, la configuración del kernel y de los módulos, la compilación propiamente dicha y, por último, la instalación. A continuación se explican estas etapas en mayor detalle.

4.13.3.1 Obtención del código fuente

Antes de poder compilar un kernel de Linux, es necesario disponer de los archivos fuente. En este caso en particular se utilizó el kernel recomendado para la placa BB, que es el desarrollado para los procesadores OMAP. Para obtener los archivos fuente, se debe abrir un terminal en la PC de desarrollo (que nuevamente se asume que dispone de un sistema operativo Ubuntu). Luego, es necesario posicionarse en el directorio donde se desea descargar los fuentes, y por último se debe ejecutar el siguiente comando:

```
$git clone git://git2.kernel.org/pub/scm/linux/kernel/git/tmlind/linux-omap-2.6.git
```

El comando *git clone* permite clonar un repositorio Git en un directorio local.

4.13.3.2 Configuración del kernel y los módulos

Dado que el código de Linux incluye drivers y herramientas para una gran cantidad de escenarios, para obtener un buen rendimiento del mismo es necesario establecer qué características serán finalmente abarcadas.

Linux posee, además, una cualidad que es ideal para desarrolladores, la capacidad de cargar y descargar módulos dinámicamente mientras se está ejecutando. Esto permite, por lo tanto, que para muchas de las características del kernel se pueda decidir si incluirlas directamente en el núcleo o si dejarlas como módulo y eventualmente cargarlas luego.



En esta etapa es muy recomendable tener un conocimiento profundo sobre la estructura y funcionamiento de los archivos **Makefile**.

La placa BB, por su parte, ya cuenta con una configuración por defecto incluida en el código fuente, pero es necesario ejecutar los siguientes comandos para establecerla como la configuración utilizada:

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnu- distclean  
$make ARCH=arm CROSS_COMPILE=arm-linux-gnu- omap3_beagle_defconfig
```

El primer comando se encarga de limpiar toda posible configuración establecida previamente de forma de comenzar con un kernel absolutamente “limpio”. Si bien puede parecer inútil, se recomienda ampliamente su utilización para evitar conflictos.

Para ejecutar correctamente dicho comando es necesario estar posicionado en la carpeta del código. Además, es necesario aclarar que el parámetro ARCH del comando es necesario para indicar la arquitectura de la plataforma objetivo y el parámetro CROSS_COMPILE indica qué cadena de herramientas utilizar para compilar el kernel. En este caso se utilizó la de **Emdebian**. En caso de utilizar otra cadena de herramientas, se deberá cambiar adecuadamente este parámetro.

Para realizar cambios sobre la configuración establecida existen varias herramientas gráficas especialmente diseñadas con ese propósito. En el proyecto se utilizó **Menuconfig** ya que es intuitiva, fácil de usar y está instalada por defecto en la mayoría de las distribuciones de Linux. La ejecución de esta herramienta se realiza mediante el siguiente comando:

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnu- menuconfig
```

4.13.3.3 Compilación

Una vez establecida la configuración deseada, es necesario generar la imagen comprimida del kernel **uImage** y los **módulos** correspondientes. Esto se realiza mediante los siguientes comandos:

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnu- uImage  
$make ARCH=arm CROSS_COMPILE=arm-linux-gnu- modules
```



El primer comando requiere que previamente se haya instalado la herramienta **mkimage**, la cual fue mencionada en la sección “Compilación e instalación de ARM EABI Ubuntu 9.04”.

La ejecución de cada uno de estos comandos demorará un tiempo considerable, dependiendo de la configuración establecida, ya que son los que efectivamente compilan el kernel.

4.13.3.4 Instalación

Finalmente es necesario colocar adecuadamente la imagen del kernel y los módulos dentro de la tarjeta SD que utiliza la placa BB.

La dificultad de instalar la imagen del kernel en la tarjeta radica en que **uBoot** (que es el cargador de arranque que utiliza la placa BB) no maneja adecuadamente cadenas de caracteres nulos dentro de la imagen. Es por esto que copiar y pegar la imagen del kernel dentro de la partición FAT32 de la SD podría ocasionar problemas y llevar a que el kernel no arranque adecuadamente.

Afortunadamente en Ubuntu existe el comando “**cat**” que puede mostrar el contenido de un archivo sin cadenas de caracteres nulos y es posible redirigir la salida de dicho comando a un archivo en particular. Teniendo esto en cuenta, el comando utilizado para mover la imagen del kernel resulta como se muestra a continuación:

```
$tar arch/arm/boot/ulmage > <path a la partición FAT32 de la SD>/ulmage
```

La instalación de los módulos, por su parte, ya cuenta con un script que automatiza su instalación y es el siguiente:

```
$sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnu- INSTALL_MOD_PATH=<path al sistema de archivos de la placa BB> modules_install
```

Es importante la utilización del parámetro **INSTALL_MOD_PATH** ya que de lo contrario se instalarían los módulos localmente y al estar generados para otra arquitectura podrían llevar a arruinar la instalación local del sistema operativo. Además, este comando se debe ejecutar como super usuario para evitar problemas de permisos sobre el sistema de archivos de la placa BB.



4.14 Adaptación y compilación de uBoot

A pesar de poder acceder y modificar determinadas señales GPIO a nivel de la capa de aplicación de Ubuntu, la señal eléctrica en el pin correspondiente del puerto de expansión, no verificaba los mismos niveles o no percibía los cambios en el caso de las entradas. Se investigó este problema y se tomó conocimiento que la manipulación de las señales GPIO a nivel de Linux no funciona adecuadamente, por lo que es necesario configurarlas a nivel del kernel a través del cargador de arranque (uBoot en este caso).

Por lo tanto, se compiló uBoot manualmente para configurar apropiadamente las señales del puerto de expansión de la placa BB que se deseaban utilizar. También fue necesario establecer la resistencia interna que se deseaba aplicar al pin asociado a cada una de dichas señales.

4.14.1 Cambios en el código de uBoot

Para configurar las señales del puerto de expansión fue necesario modificar el archivo **board/ti/Beagle.h** del código de uBoot, el cual puede ser descargado de la URL <http://www.denx.de/wiki/U-Boot/WebHome>.

En el caso del proyecto UMix es necesario asegurarse de que las señales correspondientes a los pines 6, 7, 8, 9, 10, 15, 23 y 24 correspondan al protocolo GPIO. Para lograr esto, se debe buscar el pin en el archivo de configuración mencionado utilizando el nombre de la señal. Se utiliza como nombre de la señal el nombre correspondiente a la opción “A”. Una vez encontrada la línea de código correspondiente al pin que se desea configurar se deben ajustar los parámetros de forma que:

- Se utilice el modo 4, que es el que corresponde al protocolo GPIO para estos pines (con el parámetro M4).
- Para todos los pines excepto el correspondiente al sensor IR, la resistencia de entrada esté habilitada (con el parámetro “enable” (EN)) y sea del tipo “pull type down” (con el parámetro PTD).
- Para el pin correspondiente al sensor IR, la resistencia de entrada esté deshabilitada (con el parámetro “disable” (DIS)).



- Sea de entrada y salida (con el parámetro IEN)

De esta forma resulta que el archivo **Board/ti/Beagle.h** contiene las siguientes líneas:

```
MUX_VAL(CP(MCBSP3_DR), (IEN | PTD | EN | M4))
MUX_VAL(CP(MMC2_DAT5), (IEN | PTD | EN | M4))
MUX_VAL(CP(MCBSP3_FSX), (IEN | PTD | EN | M4))
MUX_VAL(CP(MMC2_DAT4), (IEN | PTD | EN | M4))
MUX_VAL(CP(MCBSP3_DX), (IEN | PTD | EN | M4))
MUX_VAL(CP(MMC2_DAT1), (IEN | PTD | EN | M4))
MUX_VAL(CP(I2C2_SDA), (IEN | PTD | EN | M4))
MUX_VAL(CP(I2C2_SCL), (IEN | PTD | DIS | M4))
```

4.14.2 Toolchain

La compilación de uBoot para la plataforma objetivo requiere de un compilador y otra serie de herramientas que permitan generar código que pueda ser interpretado directamente por el procesador de la plataforma.

En este caso se utilizó la toolchain **arm-none-linux-gnueabi** que provee **Code Sourcery**. La misma puede ser descargada de <http://www.codesourcery.com>

4.14.3 Compilación de uBoot

La compilación de uBoot se realiza mediante los siguientes comandos:

```
make CROSS_COMPILE=arm-none-linux-gnueabi- distclean
make CROSS_COMPILE=arm-none-linux-gnueabi- omap3530beagle_config
make CROSS_COMPILE=arm-none-linux-gnueabi-
```

Luego de ejecutados estos comandos se debería obtener al menos un archivo llamado u-boot.bin, que es la imagen que se buscaba obtener.

4.14.4 Instalación de uBoot

Para culminar con la instalación de la nueva versión de uBoot se deben seguir los pasos detallados en la sección 12.3 (“SD Card Configuration”) del manual de referencia de la placa BB.



4.15 Desarrollo de la interfaz Web

Uno de los objetivos del proyecto UMix era que el sistema pudiera ser accedido desde cualquier PC conectada a Internet. Por este motivo se optó por incluir en la implementación del proyecto el desarrollo de una interfaz Web que permita la administración y utilización del móvil.

Esta interfaz ofrece las siguientes funcionalidades:

- Cambiar la configuración de red del móvil
- Conocer la ubicación del móvil
- Conocer el estado de los sensores de obstáculos
- Conocer el estado del sensor IR
- Enviar directivas de movimiento al vehículo
- Obtener un reporte con las posiciones en las que se detectaron supervivientes

4.15.1 Fundamentos de la elección del lenguaje

La primera gran decisión que hay que realizar a la hora de implementar una interfaz Web es el lenguaje que se utilizará para su implementación. Cada lenguaje tiene sus ventajas y desventajas, las cuales deben ser tenidas en cuenta a la hora de realizar esta elección.

En este proyecto, luego de haber considerado varias alternativas, se optó por utilizar HTML conjuntamente con algunas instrucciones específicas en PHP y Java Script. Estos lenguajes se describen de forma resumida en el capítulo 2.

Se tomó esta decisión principalmente por la facilidad que tienen estos lenguajes para interactuar con el sistema, por su poca utilización de recursos y debido a que no hay un gran interés en disponer de una interfaz Web con efectos visuales muy elaborados.



4.15.2 Desarrollo

4.15.2.1 Adaptaciones a la plataforma - Elevación de privilegios

Uno de los inconvenientes que se encontraron a la hora de realizar la interfaz Web fue la ejecución de comandos que requerían la elevación de privilegios. Si bien Linux ofrece el comando *sudo*, el cual permite elevar privilegios, y este comando puede ser invocado desde un script PHP, el problema radicaba en que este comando requiere una interacción con el usuario. Es decir, la contraseña debe ser introducida en tiempo de ejecución y no es posible pasarlala como parte del comando.

En un principio se pensó en cambiar los permisos del usuario que ejecuta el servidor Apache, o en cambiar los permisos de acceso a los comandos que se necesitaban. Sin embargo, cualquiera de las soluciones propuestas implicaba comprometer gravemente la seguridad del sistema.

La solución finalmente adoptada resultó ser mucho más eficiente y mucho más simple de implementar que las anteriores ya que permite habilitar a determinado usuario la ejecución de determinados comandos como usuario privilegiado sin solicitar una contraseña.

La implementación de dicha solución se realiza en base a la modificación adecuada del archivo */etc/sudoers* mediante el editor *visudo*. Es realmente importante utilizar este editor ya que, por un lado asegura que solo un usuario accederá a dicho archivo al momento de su edición y fundamentalmente porque una vez finalizada la configuración corrobora que la sintaxis utilizada sea la correcta. Un archivo */etc/sudoers* dañado puede dejar inutilizable todo el sistema.

A continuación se muestra un ejemplo del archivo */etc/sudoers*:

```
ubuntu@beagleboard:~$ sudo visudo
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#
Defaults    env_reset
```



```
# Host alias specification

# User alias specification
User_Alias APACHE = www-data
# Cmnd alias specification
Cmnd_Alias NETWORK = /etc/init.d/networking
# User privilege specification
root ALL=(ALL) ALL
APACHE ALL=(ALL) NOPASSWD: NETWORK

# Uncomment to allow members of group sudo to not need a password
# (Note that later entries override this, so you might need to move
# it further down)
# %sudo ALL=NOPASSWD: ALL
%admin ALL=(ALL) ALL
```

Casi intuitivamente se puede apreciar que lo primero que se realiza es establecer un alias para el usuario que ejecuta el servidor Apache (www-data) y luego un alias para la lista de comandos relacionados con la configuración de la red.

Finalmente en la sección de “User privilege specification” se especifica que para el usuario con alias “APACHE” no se requerirá contraseña para ejecutar los comandos contenidos dentro de la lista de comandos con alias “NETWORK”.

4.15.2.2 GoogleMaps^{xxxvi}

Con el fin de poder visualizar claramente la posición del móvil en la interfaz Web se optó por utilizar una de las facilidades recientemente incorporadas por Google, que es la capacidad para utilizar todas las ventajas que ofrece el producto GoogleMaps mediante directivas realizadas en JavaScript.

El hecho de disponer de un mapa en la interfaz Web, permite al usuario final visualizar la ubicación del móvil y los puntos por los que ha pasado, en particular aquellos puntos en los que se han detectado supervivientes.



4.15.2.3 Diagrama de entidades

A continuación se muestra un diagrama de entidades en el que es posible visualizar la interacción entre todas las entidades involucradas en la interfaz Web.

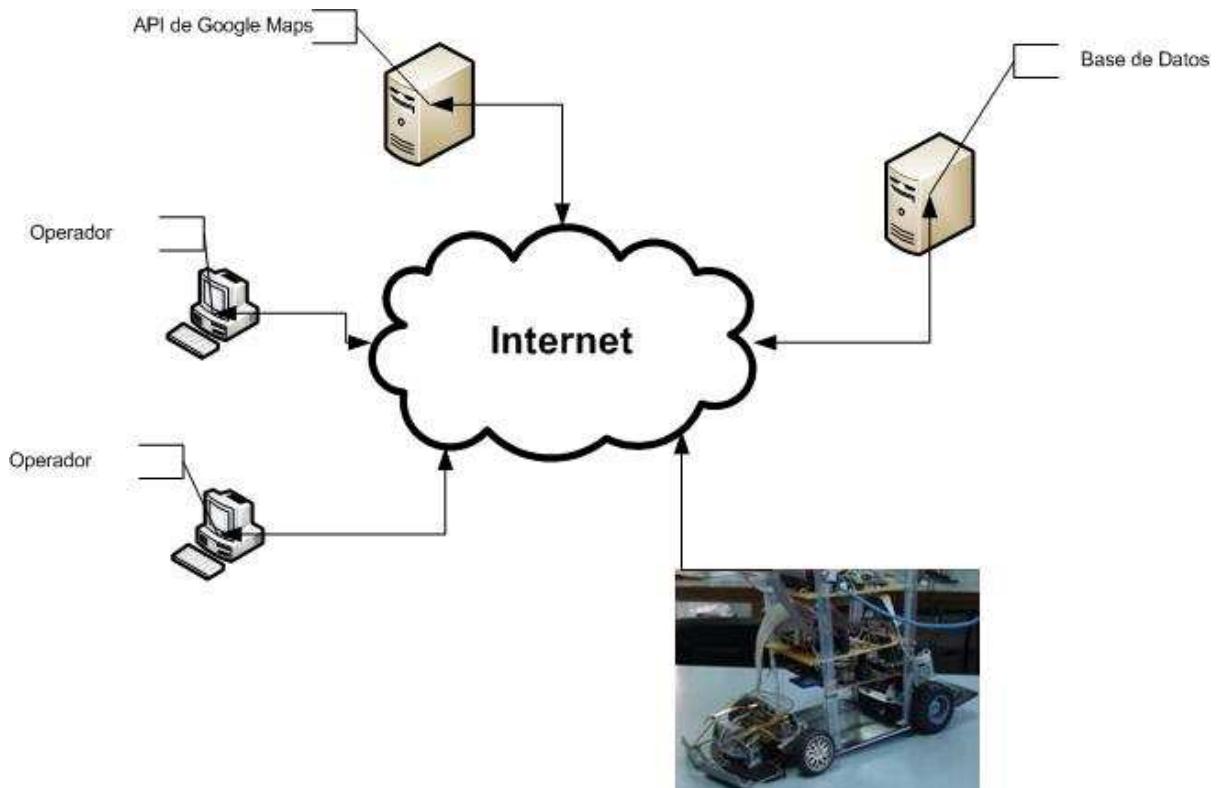


Figura 4.31 - Diagrama de entidades

En el móvil se ejecuta un servidor Web Apache, el cual permite que los operadores accedan a la interfaz Web del proyecto a través de Internet. La base de datos utilizada reside en otro servidor. El acceso a esta base de datos también se realiza a través de Internet. Por último, el servidor Web utiliza los servicios ofrecidos por la API de GoogleMaps, a la cual también se accede a través de Internet.



4.15.2.4 Diagrama de secuencia

A continuación se presenta un diagrama que ilustra la secuencia normal de operación de las funciones involucradas en la presentación del mapa con los marcadores correspondientes.

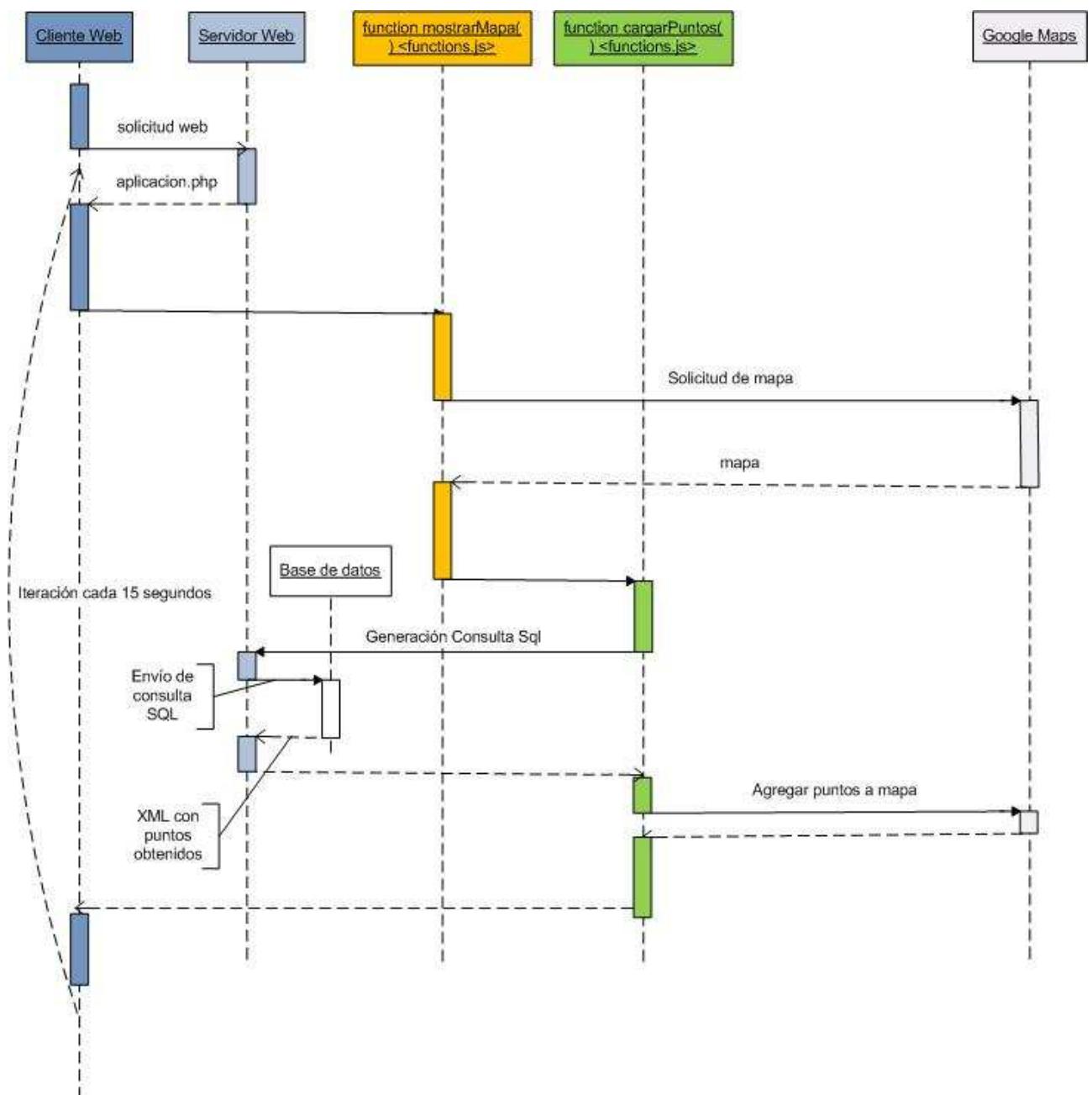


Figura 4.32 - Diagrama de secuencia



4.15.2.5 Movimiento

La funcionalidad de movimiento del sistema fue implementada mediante cuatro botones (Adelante, Atrás, Izquierda y Derecha), los cuales al ser presionados desencadenan una serie de eventos mediante los cuales el sistema es capaz de moverse y chequear el estado de los sensores frontales.

A continuación se presenta un diagrama que ilustra la secuencia de eventos que realiza el sistema cuando uno de los botones es presionado. A modo de ejemplo se muestra la secuencia de eventos que ocurren cuando se hace clic en el botón Adelante.

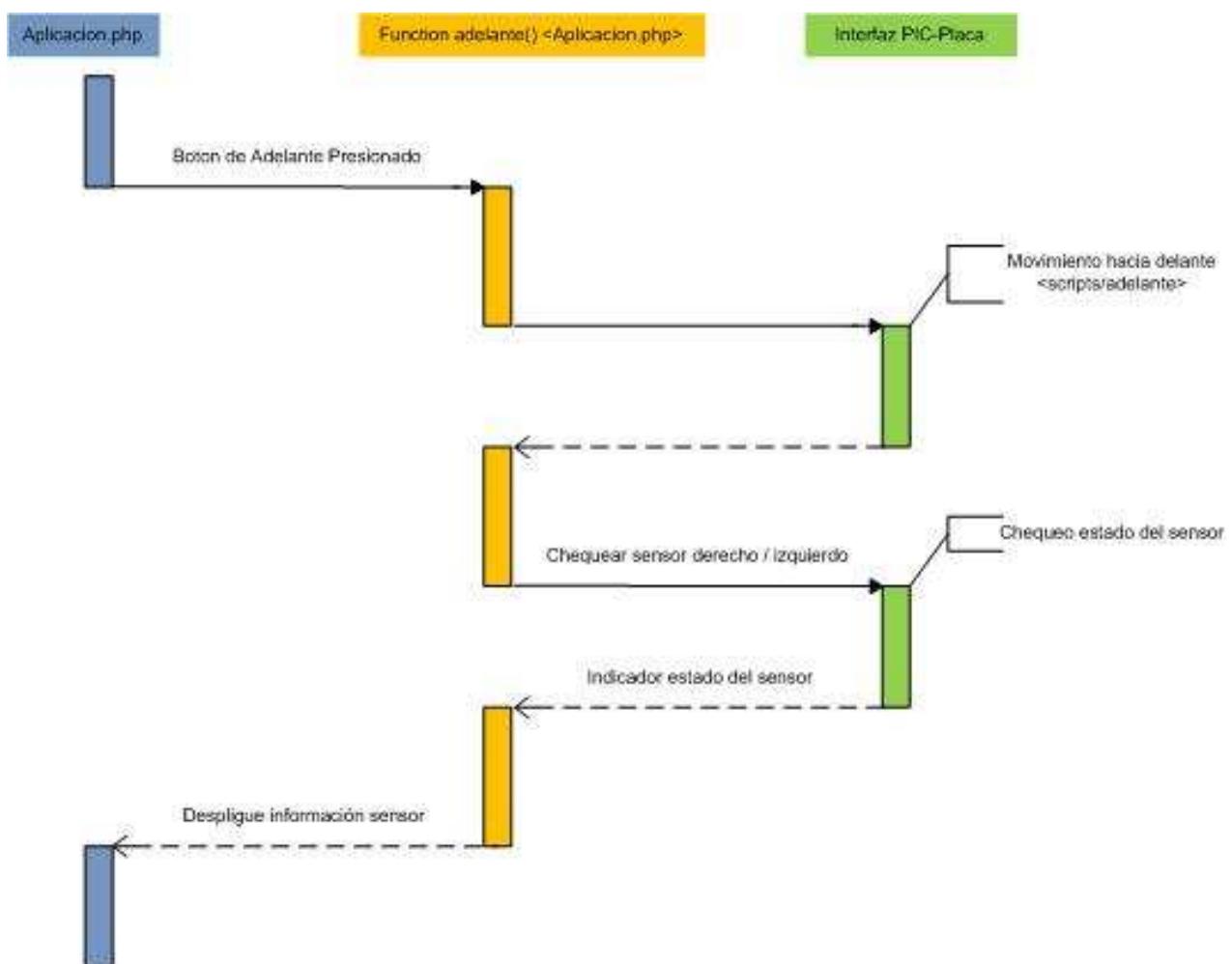


Figura 4.33 - Diagrama de secuencia movimiento

En el Anexo XI – Código interfaz Web se adjunta el código que contiene la lógica de la interfaz Web del proyecto.



4.16 Integración de los módulos

Inicialmente se decidió integrar la interfaz Web del proyecto con los otros módulos utilizando bibliotecas estáticas. Es decir, se pretendía que el código de la interfaz Web invocara las funciones definidas en las bibliotecas estáticas, pasándoles parámetros si era necesario y/u obteniendo los datos de salida en los casos que aplicara.

Sin embargo, luego de investigar sobre este tema se concluyó que la interacción entre el código PHP de la interfaz Web y las funciones de los programas desarrollados no era sencilla.

Por otro lado, teniendo en cuenta esta dificultad, se decidió que era mejor realizar ciertas interacciones con la base de datos directamente desde el código PHP de la interfaz Web y no a través del programa en lenguaje C que se había desarrollado. Se decidió que las funciones de consulta y borrado de la base de datos se realizaran desde el código de la interfaz Web directamente.

Para obtener la información del GPS, almacenarla en la base de datos y convertir las coordenadas se crearon los programas *GPSStore* y *GPSConvert*, los cuales invocan funciones de las bibliotecas *libGPS* y *libDB*.

GPSStore en primer lugar establece la comunicación con *GPSD*. Luego ingresa en un bucle infinito, lo que permite que, de forma periódica (cada cinco segundos), obtenga la fecha, la hora y la posición actual. Luego de obtener dicha información, obtiene el estado del sensor IR leyendo un archivo. Por último, almacena estos datos en la base de datos e invoca una instrucción *sleep*. El código de la interfaz Web luego obtiene la información consultando la base de datos.

Si *GPSD* aún no tiene información de posición o de fecha y hora, el punto no se almacena en la base de datos y se escribe un mensaje en un archivo informando de esta situación. Si se presionan las teclas Ctrl y C, se cierra el socket que se creó para comunicarse con *GPSD* y luego se interrumpe la ejecución de este programa.

En el Anexo IX – Código de *GPSStore* se adjunta el código de este programa.



GPSConvert es responsable de convertir las coordenadas obtenidas de GPSD en Datum WGS84 al Datum Yacaré, para esto recibe como parámetros la latitud y la longitud que se desean convertir. El primer paso es pasar las coordenadas recibidas a radianes para luego convertirlas al Datum Yacaré e imprimirlas en pantalla.

En el Anexo X – Código de *GPSConvert* se adjunta el código de este programa.



5 Conclusiones

En la entrega correspondiente al 60% de avance se logró el cumplimiento de todos los objetivos fijados para el primer prototipo. Sin embargo, para cumplir con estos objetivos, con la documentación y con los testeos necesarios para dicha entrega, fue necesario postergar ciertas tareas.

Uno de los grandes motivos de los atrasos fue el desarrollo de una aplicación C que consultara una base de datos SQL. El problema que se encontró fue que, en primer lugar, la cadena de herramientas (toolchain) utilizada no poseía las bibliotecas necesarias para cumplir esta función y además, el código generado causaba fallos de segmentación al ser ejecutado en la placa BB. La solución a esto fue instalar directamente el compilador dentro de la placa junto con las bibliotecas y compilarlo directamente sobre la placa.

Otro de los problemas encontrados se relacionó con la aplicación que establece la comunicación inalámbrica. En este caso en particular, algunas de las primitivas del kernel que utiliza dicha aplicación no estaban implementadas para la versión aplicable al procesador ARM. Es por esto que se optó por utilizar una aplicación menos amigable pero que resolvía este problema.

Durante las tareas de testeo de circuitos se experimentaron dificultades debido a que en la práctica los circuitos no se comportan exactamente igual a como lo hacen en la simulación. Si bien todos los circuitos habían sido diseñados y simulados en la aplicación Proteus, y se encontraban funcionando correctamente en la simulación, al momento de armar los circuitos en el laboratorio, estos no funcionaban como era esperado. Este problema se debió principalmente a que en la simulación no se consideran factores externos, tales como la presencia de fuentes espurias de ruido eléctrico o falsos contactos en los ProjectBoards.

También se experimentaron dificultades al intentar comunicar la placa BB con el PIC debido a que los mismos manejan lógicas con distintos niveles de voltaje. Si bien esta dificultad fue experimentada por otros equipos en otros proyectos, la solución encontrada utiliza circuitos integrados disponibles en el mercado extranjero pero no en el local.



Es importante tener en cuenta que al momento de la planificación del proyecto no se consideró la gran demanda de tiempo requerida para la elaboración de documentos. Este fue otro motivo por el cual fue necesario postergar tareas.

Faltando poco más de un mes para el plazo final de entrega de este proyecto, aún no se había podido comenzar con las tareas asociadas al movimiento automatizado del vehículo incluyendo la evasión de obstáculos. Frente al riesgo de no poder cumplir con los plazos establecidos, se solicitó al Consejo eliminar del alcance de este proyecto el desarrollo de un módulo de movimiento automatizado. Este planteo fue aprobado por el Consejo por lo que se postergó el desarrollo de dicho módulo para una posible futura ampliación del proyecto UMix.

Exceptuando el objetivo mencionado en el párrafo anterior, durante la segunda etapa del proyecto se logró cumplir de forma satisfactoria con todos los objetivos que habían quedado pendientes de la primera etapa.

En resumen, se cumplió con los siguientes puntos:

- Se compiló el kernel Linux OMAP para la placa BeagleBoard (BB).
- Se configuró en la placa BB una conexión a Internet a través de un modem inalámbrico 3G y un servidor DNS dinámico (DDNS), lo cual permite que la placa BB sea accedida desde cualquier PC conectado a Internet.
- Se logró establecer una comunicación entre la placa BB y el PIC, enviando directivas de movimiento desde la placa BB hacia el PIC y notificaciones de detección de vida y de obstáculos desde el PIC hacia la placa BB.
- Se instaló un servidor Apache con módulo para PHP en la placa BB.
- Se desarrolló una interfaz web que permite visualizar la posición del vehículo en un mapa, enviar directivas, recibir notificaciones y realizar configuraciones de red, entre otras funcionalidades.



- Se creó una base de datos SQL y se desarrolló un programa en lenguaje C, el cual establece una comunicación con la base de datos e inserta los puntos por los que va pasando el vehículo.
- Se desarrolló un programa en lenguaje C que se encarga de convertir las coordenadas del móvil en Datum WGS84 al Datum Yacaré.
- Se diseñaron circuitos y se desarrollaron rutinas para poder controlar un motor de continua y un motor paso a paso a través de un PIC.
- Se diseñó un sensor infrarrojo (IR) para simular el sensor de vida. Cuando se detecta una condición positiva en este sensor, se envía una notificación a la placa BB, la cual se encarga de almacenar esta información y notificar al usuario final.
- Se diseñaron circuitos y se desarrollaron rutinas para poder detectar obstáculos y enviar la notificación correspondiente a la placa BB, la cual se encarga de notificar al usuario final.
- Se integró al sistema un receptor GPS y se desarrolló el código necesario para conocer la posición del móvil en todo momento.
- Se diseñó la alimentación del sistema utilizando reguladores de la familia 78xx para obtener los voltajes requeridos por los distintos componentes.



6 Recomendaciones

Por cuestiones de tiempo y para poder mantener bajos los costos del proyecto, se decidió simular el sensor de vida con otro sensor. Sin embargo, podría resultar interesante incorporar a futuros proyectos que utilicen al proyecto UMix como base, la implementación de un sensor que permita detectar supervivientes.

También resultaría interesante la implementación de una plataforma física de movimiento más flexible que la utilizada en este proyecto, la cual permitiera recorrer superficies más irregulares. Incluso se podrían integrar cámaras al móvil, a través de las cuales el usuario final pueda ver la zona que el móvil está recorriendo. Por último, también podría resultar interesante mejorar la autonomía del sistema utilizando paneles solares que carguen una batería.

Se recomienda ampliamente el uso del libro “Linux kernel in a nutshell” de Greg Kroah-Hartman^{xxxvii} y el libro “Linux Device Drivers” de Greg Kroah-Hartman, Jonathan Corbet y Alessandro Rubini^{xxxviii} de para comprender mejor el funcionamiento del kernel de Linux y así poder adaptar mejor sus componentes a la placa BB. Esto permitiría optimizar el rendimiento de la placa BB y el aprovechamiento de su memoria.

Respecto del módulo de posicionamiento, se podría estudiar la aplicación GEGpsd (GoogleEarth GPSD) y analizar la posibilidad de integrarla a este módulo. Respecto de la comunicación con GPSD, durante el desarrollo de este proyecto se liberó la primera versión estable de un nuevo protocolo para establecer una conexión con GPSD. Este nuevo protocolo se basa en JSON (JavaScript Object Notation). El uso de este metaprotocolo para pasar datos estructurados entre GPSD y la aplicación cliente evita los problemas de no-extensibilidad del antiguo protocolo, y permite la posibilidad de pasar a los clientes un conjunto más rico de tipos de registros.

Queda pendiente para futuras ampliaciones de este proyecto el desarrollo de un algoritmo que permita automatizar la tarea del vehículo de recorrer una zona determinada, esquivando los obstáculos que pudieran detectarse. Para facilitar esta tarea, podría agregarse a la base de datos la información de los obstáculos detectados.



7 Bibliografía

- ⁱbeagleboard.org. *BeagleBoard System Reference Manual Rev C3* [en línea] <http://beagleboard.org/static/BBSRM_latest.pdf> [citado en 7 de diciembre de 2009].
- ⁱⁱTexas Instruments. *OMAP3530/25 Applications Processor.* [en línea] <<http://focus.ti.com/lit/ds/symlink omap3530.pdf>> [citado en 7 de diciembre de 2009].
- ⁱⁱⁱHewlett-Packard Company et al. *Universal Serial Bus 3.0.* [en línea] <http://www.gaw.ru/pdf/interface/usb/USB%203%200_english.pdf> [citado en 7 de diciembre de 2009].
- ^{iv}Microchip Technology Inc. *PICmicro™ Mid-Range MCU Family Reference Manual* [en línea] <<http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf>> [citado en 4 de diciembre de 2009].
- ^vMicrochip Technology Inc. *MPLAB® IDE User's Guide* [en línea] <<http://ww1.microchip.com/downloads/en/DeviceDoc/51519B.pdf>> [citado en 4 de diciembre de 2009].
- ^{vi}Microchip Technology Inc. *MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide* [en línea] <<http://ww1.microchip.com/downloads/en/DeviceDoc/33014J.pdf>> [citado en 4 de diciembre de 2009].
- ^{vii}ATE-Universidad de Oviedo. *Adaptación de Salidas: Control de Motores* [en línea] <<http://www2.ate.uniovi.es/fernando/.../Control%20de%20motores.pdf>> [citado en 3 de diciembre de 2009].
- ^{viii}Martin, Daniel C., *Motorización* [en línea] <<http://www.x-robotics.com/motorizacion.htm>> [citado en 4 de diciembre de 2009].
- ^{ix}SGS-THOMSON Microelectronics. *Datasheet L293D.* [en línea] <<http://www.datasheetcatalog.org/datasheet/stmicroelectronics/1330.pdf>> [citado en 3 de diciembre de 2009].
- ^xRossing, Thomas D. *The Science of Sound.* 2nd Ed. Addison-Wesley. 1990. 686p
- ^{xi}Tomi Engdahl. *Powering microphones* [en línea] <http://www.epanorama.net/circuits/microphone_powering.html> [citado en 3 de diciembre de 2009].
- ^{xii}Mancini, Ron. *Opamps for everyone* [en línea] <<http://focus.ti.com.cn/cn/lit/an/slod006b/slod006b.pdf>> [citado en 4 de diciembre de 2009].
- ^{xiii}National Semiconductor Corporation. *LM567/LM567C Tone Decoder Datasheet* [en línea] <<http://www.national.com/ds/LM/LM567.pdf>> [citado en 4 de diciembre de 2009].
- ^{xiv}Fairchild Semiconductor. *LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator* [en línea] <<http://www.fairchildsemi.com/ds/LM/LM7805.pdf>> [citado en 26 de febrero de 2010].
- ^{xv}www.gps.gov. *Global Positioning System - Serving the world* [en línea] <<http://www.gps.gov>> [citado en 27 de enero de 2010].
- ^{xvi}gpsinformation.net. *NMEA Data* [en línea] <<http://www.gpsinformation.org/dale/nmea.htm>> [citado en 27 de enero de 2010].
- ^{xvii}Arq. Mercedes Frassia. *Entendiendo la proyección de los mapas* [en línea] <http://www.inia.org.uy/disciplinas/agroclima/agric_sat/gps/proyeccion_gauss-kruger.pdf> [citado en 30 de enero de 2010].



^{xviii} gpsd.berlios.de. *gpsd — a GPS service daemon* [en línea] <<http://gpsd.berlios.de/>> [citado en 27 de enero de 2010].

^{xix} Miguel Rueda Barranco. *Sockets: Comunicación Entre Procesos Distribuidos* [en línea] <<http://ftp.nluug.nl/pub/os/Linux/doc/LuCaS/Universitarios/seminario-2-sockets.html>> [citado en 30 de enero de 2010].

^{xx} Autor desconocido. *Programación en C* [en línea] <<http://cfp401.freeservers.com/cursos/c1/c.htm>> [citado en 24 de febrero de 2010].

^{xxi} www.carlospes.com. *Curso de lenguaje C* [en línea] <http://www.carlospes.com\curso_de_lenguaje_c\default.htm> [citado en 19 de febrero de 2010].

^{xxii} www.aquaphoenix.com. *The GNU C library* [en línea] <http://www.aquaphoenix.com\ref\gnu_c_library\default.htm> [citado en 19 de febrero de 2010].

^{xxiii} gcc.gnu.org. *GCC, The GNU Compiler Collection* [en línea] <<http://gcc.gnu.org/>> [citado en 23 de febrero de 2010].

^{xxiv} Grupo ARCO. *Librerías* [en línea] <<http://arco.esi.uclm.es/~david.villa/doc/repo/librerias/librerias.html>> [citado en 22 de marzo de 2010].

^{xxv} www.chuidiang.com. *Librerías* [en línea] <<http://www.chuidiang.com/clinix/herramientas/librerias.php>> [citado en 22 de marzo de 2010].

^{xxvi} profeblog.es. *Creación de librerías estáticas* [en línea] <<http://profeblog.es/blog/alfredo/2009/01/22/creacion-de-librerias-estaticas-en-c/>> [citado en 22 de marzo de 2010].

^{xxvii} dev.mysql.com. *MySQL 5.0 Reference Manual* [en línea] <<http://dev.mysql.com/doc/refman/5.0/en/c.html>> [citado en 23 de febrero de 2010].

^{xxviii} www.w3.org. *What is HTML?* [en línea] <<http://www.w3.org/TR/REC-html40/intro/intro.html#h-2.2>> [citado en 22 de marzo de 2010].

^{xxix} www.monografias.com. *Lenguaje de programación para páginas Web* disponible en <http://www.monografias.com/trabajos7/html/html.shtml>, 22 de marzo de 2010.

^{xxx} php.net. *Página oficial de PHP* [en línea] <<http://php.net/index.php>> [citado en 22 de marzo de 2010].

^{xxxi} www.webestilo.com. *JavaScript* [en línea] <<http://www.webestilo.com/javascript/>> [citado en 21 de marzo de 2010].

^{xxxii} Texas Instruments Incorporated. *TL070 JFET-INPUT OPERATIONAL AMPLIFIER Datasheet*. [en línea] <<http://pdf1.alldatasheet.com/datasheet-pdf/view/28771/TI/TL070.html>> [citado en 4 de diciembre de 2009].

^{xxxiii} www.ucl.ac.uk. *Calling MySQL from C* [en línea] <<http://www.ucl.ac.uk/is/mysql/c/>> [citado en 25 de febrero de 2010].

^{xxxiv} www.cyberciti.biz. *Howto: Connect MySQL server using C program API under Linux or UNIX* [en línea] <<http://www.cyberciti.biz/tips/linux-unix-connect-mysql-c-api-program.html>> [citado en 25 de febrero de 2010].

^{xxxv} zetcode.com. *MySQL C API programming tutorial* [en línea] <<http://zetcode.com/tutorials/mysqlcapitutorial/>> [citado en 25 de febrero de 2010].

^{xxxvi} GoogleMaps API. *Using PHP/MySQL with GoogleMaps* [en línea] <<http://code.google.com/intl/es-UY/apis/maps/articles/phpsqlajax.html>> [citado en 18 de abril de 2010].



^{xxxvii}Greg Kroah-Hartman. *Linux kernel in a nutshell*. 1st edition. Editorial O'Reilly. 2006. 182p

^{xxxviii}Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. *Linux Device Drivers*. 3rd edition. Editorial O'Reilly. 2005. 636p



8 Anexo I – Plan del Proyecto Final de Carrera

8.1 Acta del Proyecto

8.1.1 Nombre del Proyecto

Implementación de un sistema automatizado de búsqueda de supervivientes de catástrofes.

8.1.2 Fecha de comienzo

31 de Agosto de 2009

8.1.3 Áreas de conocimientos / procesos

Las áreas de conocimientos que están involucradas en este proyecto son:

Área General	Área específica	Descripción
Desarrollo de plataformas	Adaptación del sistema operativo Linux	El proyecto requiere la adaptación del sistema operativo Linux a determinada plataforma de desarrollo
Electrónica	Desarrollo de fuentes de alimentación de Corriente continua	Se utilizarán fuentes de alimentación de corriente continua que deben ser desarrolladas por el equipo de proyecto
	Drivers de motores	Los motores que se utilizan en el proyecto se manejarán mediante un hardware que será diseñado específicamente para el proyecto
	Interfaces de comunicación	Todas las interfaces de comunicación requieren una interfaz de hardware que será estudiada (y desarrollada en caso de ser necesario).
Programación	Assembler para el manejo de sensores y motores	Todo el manejo de sensores y motores será realizado mediante software desarrollado en assembler
	Interfaz Web	El proyecto requiere de una interfaz web que permita la administración del móvil
	Base de datos	Como requerimiento del proyecto se especificó la posibilidad de almacenar resultados en una base de datos. Es por esto que el proyecto requiere de la implementación de algún modelo que permita respaldar la información necesaria.
	Interfaces de comunicación	El proyecto depende en gran medida de la comunicación de los distintos módulos. Es por esto que es necesario desarrollar e



		implementar mediante software distintos protocolos de comunicación.
Protocolos de comunicación inalámbrica	Utilización de protocolos de comunicación inalámbrica	El proyecto es dependiente de la capacidad de comunicarse inalámbriamente con otros terminales. Es por esto que este proyecto involucra el estudio y uso de las redes inalámbricas existentes.

8.1.4 Áreas de aplicación (sector / actividad)

- Fuerzas armadas, ejércitos
- Entes Gubernamentales que participen en la búsqueda de supervivientes
- ONG que participen en la búsqueda de supervivientes
- Investigadores particulares de robótica
- Desarrolladores de Linux

8.1.5 Fecha de inicio del Proyecto

12 de Agosto de 2009

8.1.6 Fecha tentativa de finalización del Proyecto

15 de Marzo de 2010

8.2 Objetivos del proyecto

8.2.1 Objetivo General

Contribuir con el desarrollo de tecnologías que faciliten la tarea de búsqueda de supervivientes en áreas de catástrofes. Para ello se utilizarán componentes de fácil adquisición y estandarizados y se recurrirá a código open source, de forma de permitir futuros desarrollos sobre la base de este proyecto.

8.2.2 Objetivos Específicos

Implementar funcionalidades básicas de búsqueda de personas, es decir realizar un móvil que se desplace de forma autónoma en un área determinada e informe sobre los lugares específicos donde se encontraron posibles supervivientes. Para esto se utilizará un sistema operativo Linux sobre una placa BeagleBoard, y un micro controlador PIC 16F877A.

Desarrollar un módulo, dentro del móvil, que permita la comunicación inalámbrica con una computadora personal utilizando protocolos de red estandarizados. Esto permitirá la fácil administración y manipulación del móvil.

Incluir un módulo de posicionamiento que haga uso de un dispositivo GPS estándar para determinar la ubicación del móvil en todo momento y en particular cuando se detecta un superviviente.



Diseñar el sistema de movimiento de forma tal que la plataforma física de movimiento sea fácilmente intercambiable.

8.3 Descripción del producto

UMix pretende ser un sistema automatizado para la búsqueda de supervivientes de catástrofes.

Se utilizarán diversos dispositivos de ubicación, monitoreo y comunicación ya estandarizados para cumplir con su objetivo. Fundamentalmente hará uso de un sistema de GPS para contar con la ubicación del producto en todo momento y delimitar el área de búsqueda, en coordinación con sensores que le permitirán esquivar obstáculos y hallar los supervivientes y un sistema de comunicación inalámbrica estándar que permitirá notificar a un dispositivo cliente la ubicación de las víctimas.

Todo el sistema será implementado en base a software, interfaces y plataformas libres, lo cual permitirá la fácil adaptación de este proyecto a otros sistemas.

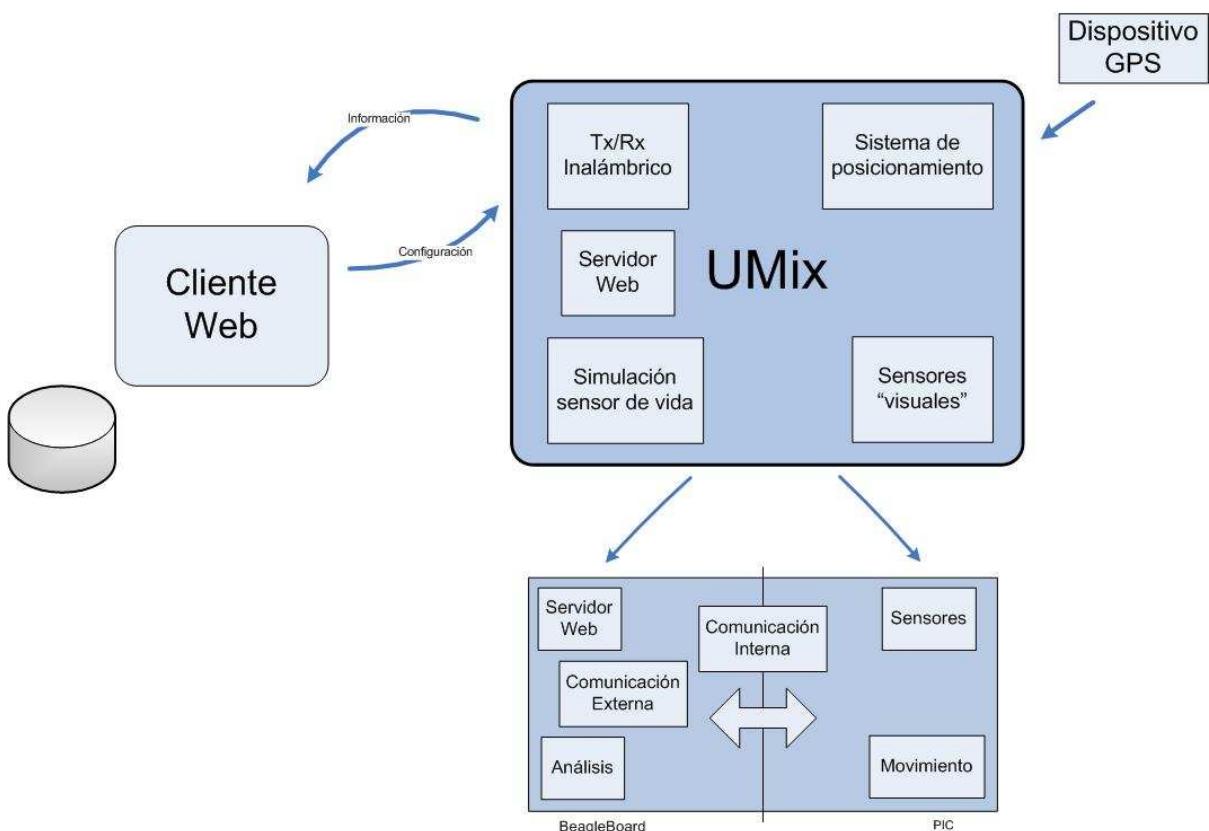


Figura 8.1 - Esquema modular

8.4 Necesidad del Proyecto

Hoy en día en el mundo suceden catástrofes muy frecuentemente. Esto genera enormes cantidades de heridos cuyo rescate suele ser una tarea ardua y compleja. Por este motivo es que cualquier proyecto que facilite dichas tareas es atractivo. Este proyecto en particular pretende automatizar la tarea de búsqueda de supervivientes de dichas catástrofes.



8.5 Alcance

8.5.1 Incluido

Las funcionalidades incluidas previstas para este proyecto abarcan:

- AP-1.** Sistema de posicionamiento: Este sistema hará uso de un dispositivo de GPS disponible comercialmente para establecer la posición del móvil en todo momento y permitir al usuario delimitar la zona de búsqueda.
- AP-2.** Sistema de sensores de objetos: Se utilizaran sensores cuya función será detectar posibles obstáculos para que el móvil pueda esquivarlos.
- AP-3.** Sensor de sonido: Para simplificar la implementación del prototipo se descartó utilizar sensores de signos vitales. En su lugar se utilizará un sensor de sonido.
- AP-4.** Sistema de comunicación: El sistema ofrecerá una interfaz web para su uso y administración, la cual podrá ser accedida por los usuarios mediante un receptor/transmisor 3G.
- AP-5.** Sistema de respaldo: Se ofrecerá la posibilidad de guardar diversa información en una base de datos remota.
- AP-6.** Circuitos anexos para integrar los sistemas anteriores

8.5.2 No incluido

El prototipo del sistema NO incluirá la implementación de los siguientes ítems:

- AP-7.** Placa multipropósito
- AP-8.** Plataforma física de movimiento
- AP-9.** Kernel de Linux
- AP-10.** Dispositivo GPS
- AP-11.** Adaptador 3G

8.6 Justificación del Impacto (aporte y resultados esperados)

UMix es un proyecto que no pretende competir con sistemas automatizados de búsqueda que utilicen alta tecnología, si no que su objetivo es brindar una solución económica, sencilla y modular que pueda adaptarse a distintas realidades. Además pretende la base para futuros proyectos de investigación y desarrollo que haya en la rama.

Si bien muchos de los componentes que se utilizarán en la implementación de UMix ya existen, no existe ningún proyecto hasta el momento que las vincule y las integre en una solución de estas características.

Se espera que el valor agregado por la integración de todas las tecnologías y las herramientas desarrolladas permitan que el proyecto UMix tenga un gran impacto y aceptación en el mercado.

8.7 Entregables

- EP-1.** **25 de agosto de 2009** Documento de definición del tema
- EP-2.** **9 de setiembre de 2009** Plan de Proyecto
- EP-3.** **16 de diciembre de 2009** Primer prototipo



-
- EP-4.** **16 de diciembre de 2009** Primer borrador del Informe (60%)
 - EP-5.** **15 de febrero de 2010** Segundo borrador del Informe (100%)
 - EP-6.** **25 de febrero de 2010** Informe Final
 - EP-7.** **30 de marzo de 2010** Afiche
 - EP-8.** **30 de abril de 2010** Paper para memoria de trabajos de difusión científica y técnica de la UM

8.8 Restricciones

Consideramos que los siguientes puntos son restricciones para este proyecto:

- RP-1.** No contar con el hardware adecuado para la implementación del sistema, es decir no disponer de una placa BeagleBoard y un PIC 16F877A en correcto funcionamiento.
- RP-2.** No disponer del software necesario para el proyecto, es decir no disponer de un kernel de Linux, drivers ó herramientas que funcionen adecuadamente.
- RP-3.** No disponer de documentación sobre el hardware, software y protocolos utilizados.

8.9 Identificación de grupos de interés (*directos e indirectos*)

Los siguientes grupos fueron identificados como interesados en el proyecto:

- GIP-1.** Grupos sociales que habiten en zonas de accidentes o catástrofes naturales frecuentes.
- GIP-2.** Gobiernos que prioricen la ayuda humanitaria
- GIP-3.** Organizaciones vinculadas al rescate de personas.
- GIP-4.** Empresas dedicadas a la venta de este tipo de tecnología
- GIP-5.** Desarrolladores de sistemas automatizados
- GIP-6.** Desarrolladores del kernel de Linux



9 Anexo II – Actas

9.1 Acta N°1

Montevideo, 2 de setiembre de 2009

En la clase del día Miércoles 26 de agosto de 2009 fue planteada por el equipo la opción de acotar el alcance del proyecto cambiando la tarea “Implementación del módulo de detección de obstáculos” por la tarea “Adaptación e integración del módulo de detección de obstáculos”.

Es decir que en lugar de desarrollar el módulo de detección de obstáculos desde cero se obtendrá código existente en Internet y se procederá a hacer las modificaciones necesarias para adaptarlo e integrarlo al sistema del proyecto.

Este cambio fue aprobado por los tutores del proyecto, Marcelo Abreu y Thomas Hobbins.

En el día de la fecha se deja constancia del cambio mencionado mediante la presente acta, la cual será firmada por los tutores mencionados y por los integrantes del equipo (Juan Brenes y Sofía Silva).



9.2 Acta N°2

Montevideo, 16 de Setiembre de 2009

Durante la semana comprendida entre la clase del día Miércoles 9 de setiembre de 2009 y la clase del día de la fecha el equipo decidió construir la plataforma de movimiento en lugar de utilizar un carro comprado.

La mencionada plataforma será construida reutilizando el chasis de un auto de juguete, el motor paso a paso (para la dirección) de una disquetera, un motor de continua (para la tracción) de algún otro juguete y algunos otros implementos.

Este cambio fue aprobado por los tutores del proyecto, Ing. Marcelo Abreu e Ing. Thomas Hobbins.

En el día de la fecha se deja constancia del cambio mencionado mediante la presente acta, la cual será firmada por los tutores mencionados y por los integrantes del equipo (Juan Brenes y Sofía Silva).



9.3 Acta N°3

Montevideo, 14 de Octubre de 2009

En la clase del día Miércoles 7 de octubre de 2009 el equipo planteó un cambio en el orden de las tareas que había sido establecido en el cronograma inicial y manifestó la necesidad de agregar una subtarea de la tarea “Adaptación e integración del módulo de detección de obstáculos” debido a que ésta no había sido tenida en cuenta.

Este cambio fue aprobado por los tutores del proyecto, Ing. Marcelo Abreu e Ing. Thomas Hobbins, por lo que se realizó el control de cambio con la justificación correspondiente.

En el día de la fecha se deja constancia del cambio mencionado mediante la presente acta, la cual será firmada por los tutores mencionados y por los integrantes del equipo (Juan Brenes y Sofía Silva).



9.4 Acta N°4

Montevideo, 21 de Octubre de 2009

Durante la semana comprendida entre la clase del día Miércoles 7 de octubre de 2009 y la clase del día Miércoles 14 de octubre de 2009 se tomaron decisiones respecto al sensor de sonido y respecto a los sensores de obstáculos. Además, se avanzó con la implementación a nivel del PIC de rutinas de atención de las interrupciones generadas por los sensores y por la placa BeagleBoard.

Se decidió implementar el sensor de sonido utilizando un micrófono, un amplificador operacional y un detector de tono. El micrófono captará los sonidos del ambiente y entregará una señal de voltaje. El amplificador operacional permitirá filtrar y amplificar la señal para luego entregársela al detector de tono. La salida del detector de tono activará un interruptor controlado por voltaje, lo cual hará que el PIC reciba un “1” lógico en la entrada correspondiente al sensor de sonido (RB6).

Se decidió utilizar flip-flops RS asíncronos para retener el pulso generado por los sensores al detectar un obstáculo. También se decidió implementar dichos flip-flops utilizando compuertas NOR.

Se implementaron las rutinas de atención de interrupciones a nivel del PIC, las cuales permiten distinguir entre los distintos tipos de interrupciones: generada por la placa BeagleBoard para enviar una directiva, generada por alguno de los sensores de obstáculos (derecho o izquierdo) al detectar un obstáculo y generada por el sensor de sonido al detectar un sonido en una frecuencia predeterminada.

Estos avances fueron verificados por los tutores de proyecto, Ing. Marcelo Abreu e Ing. Thomas Hobbins.

En el día de la fecha se deja constancia de los avances mencionado mediante la presente acta, la cual será firmada por el tutor mencionado y por los integrantes del equipo (Juan Brenes y Sofía Silva).



9.5 Acta N°5

Montevideo, 21 de Octubre de 2009

Durante la semana comprendida entre la clase del día Miércoles 14 de octubre de 2009 y la clase del día de la fecha se realizó un debug del programa principal del PIC en el modelo simulado con el programa Proteus. El resultado de este debug fue que todas las rutinas implementadas hasta el momento funcionaron como se esperaba.

En esta semana también se investigó sobre los circuitos integrados 308 y 567 a utilizarse para el sensor de sonido y se documentó lo investigado.

Estos avances fueron verificados por los tutores de proyecto, Ing. Marcelo Abreu e Ing. Thomas Hobbins.

En el día de la fecha se deja constancia de los avances mencionado mediante la presente acta, la cual será firmada por el tutor mencionado y por los integrantes del equipo (Juan Brenes y Sofía Silva).



9.6 Acta N°6

Montevideo, 11 de Noviembre de 2009

Durante la semana comprendida entre la clase del día Miércoles 4 de noviembre de 2009 y la clase del día de la fecha se avanzó con la implementación de circuitos.

Por un lado, se implementaron los flip-flops para los sensores de obstáculos utilizando la compuerta NOR CMOS 4001 y se verificó que funcionaran correctamente.

Por otro lado, se implementó el circuito para el sensor de sonido. Dicho circuito está compuesto por tres etapas: micrófono, amplificador operacional y detector de tono. Se logró el correcto funcionamiento de las tres etapas por separado. Luego, se integró la etapa de amplificación con la etapa de detección de tono y se logró el correcto funcionamiento de esta etapa. Sin embargo, no se logró el correcto funcionamiento de las tres etapas juntas, ya que al conectar la salida del micrófono al amplificador operacional se pudo observar que la señal se contamina con ruido, por lo que el harmónico de 1 kHz no es detectado por el detector de tono.

Estos avances fueron verificados por los tutores de proyecto, Ing. Marcelo Abreu e Ing. Thomas Hobbins.

En el día de la fecha se deja constancia de los avances mencionado mediante la presente acta, la cual será firmada por el tutor mencionado y por los integrantes del equipo (Juan Brenes y Sofía Silva).



10 Anexo III – Configuración de HyperTerminal

A continuación se detallan los pasos a seguir para establecer una comunicación serial con la placa BeagleBoard.

1. Iniciar una sesión en Windows y ejecutar el programa **HyperTerminal** para crear una nueva conexión. Debería aparecer una pantalla similar a la siguiente:



Figura 10.1 - Nueva conexión HyperTerminal

2. Escoger un nombre para la conexión. A continuación aparecerá una pantalla donde se podrá elegir el tipo de conexión a realizar. En el caso de este proyecto se seleccionó **COM3** aunque esto depende de cada computadora en particular.



Figura 10.2 - Establecimiento de la conexión serial



3. Para terminar de configurar la conexión, la aplicación HyperTerminal presenta una ventana en donde se pueden cambiar determinados parámetros de la conexión. Para conectarse con la placa BB se debe seleccionar la siguiente configuración.

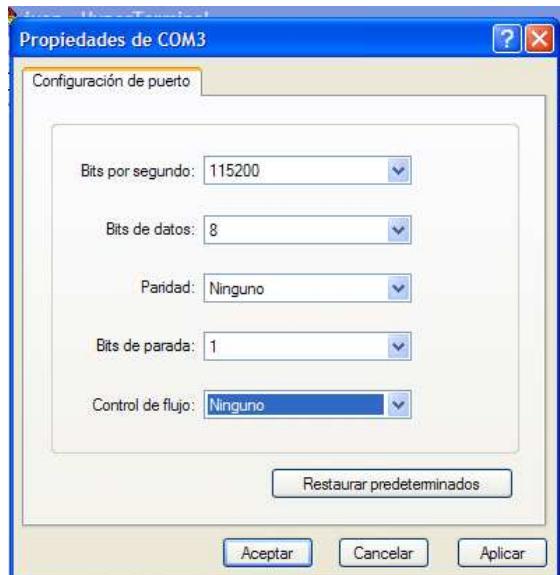


Figura 10.3 - Propiedades de la conexión

4. En caso de haber funcionado correctamente debería aparecer una serie de caracteres como los siguientes:

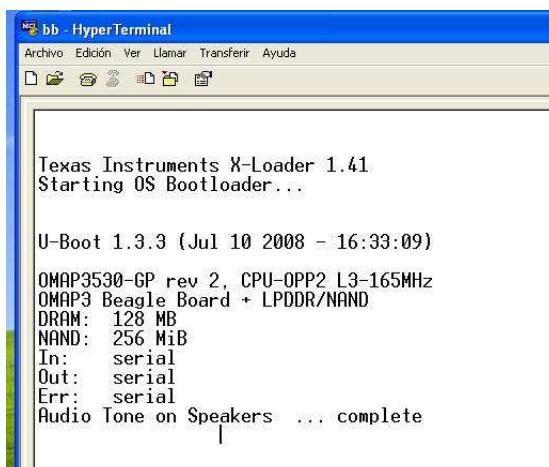


Figura 10.4 - Conexión establecida



11 Anexo IV – Código del PIC

```
;*****  
;* Microchip Technology Incorporated  
;* PIC 16F877A  
;*****  
  
; Include file, change directory if needed  
include "p16f877a.inc"  
list p=16f877a  
ERRORLEVEL 1,-302  
  
ORG 0x00  
goto Main  
  
ORG 0x04  
goto AtencionInterrupcion  
  
ORG 0x20  
CONT  
ORG 0x21  
PASOS  
ORG 0x22  
TEMP ;Variable temporal  
ORG 0x23  
Tipoint ;Variable auxiliar para grabar el tipo de interrupción  
ORG 0x24  
Aux  
  
ORG 0x25  
Main  
call Inicializacion  
bsf PORTD,7 ;DEBUG  
sleep  
nop  
Chequeo  
call banco0  
movlw b'00110010'  
movwf Aux ;Seteo Aux en 50 para que el tiempo de espera para notificar  
           ;a la BB sea de 2 seg aprox  
btfsc Tipoint,0 ;Me fijo si me enviaron Adelante  
goto SubrutinaAdelante ;Si, entonces ejecuto SubrutinaAdelante  
btfsc Tipoint,1 ;No, entonces me fijo si me enviaron Atrás  
goto SubrutinaAtras ;Si, entonces ejecuto SubrutinaAtras  
btfsc Tipoint,2 ;No, entonces me fijo si me enviaron Derecha  
goto SubrutinaDerecha ;Si, entonces ejecuto SubrutinaDerecha  
btfsc Tipoint,3 ;No, entonces me fijo si me enviaron Izquierda  
goto Subrutinalzquierda ;Si, entonces ejecuto Subrutinalzquierda  
btfsc Tipoint,4 ;No, entonces me fijo si se detectó un obstáculo en la  
               ;derecha
```



```
goto SubrutinaObstDer ;Si, entonces ejecuto SubrutinaObstDer
btfsc Tipoint,5 ;No, entonces me fijo si se detectó un obstáculo en la
;izquierda
goto SubrutinaObstIzq ;Si, entonces ejecuto SubrutinaObstIzq
btfsc Tipoint,6 ;No, entonces me fijo si se detectó luz IR
goto SubrutinalIR ;Si, entonces ejecuto SubrutinalIR
goto Main ;No, entonces vuelvo a Main

Inicializacion
call banco0
clrf PORTA
clrf PORTB
clrf PORTC
clrf PORTD
call banco1
movlw b'11111111'
movwf TRISA ;Configuramos el puerto A como entrada

movlw b'01110001' ;Configuramos los bits RB0 y RB4:RB6 como entradas y el
;resto como salidas,
movwf TRISB ;vamos a utilizar las entradas para la interrupción para
;recibir directivas (RB0) y para los sensores (RB4:RB6)
;y las salidas RB1 y RB2 para volver a poner en cero los flip-
;flops asociados a los sensores de obstáculos.

movlw b'00000000'
movwf TRISC ;Configuramos el puerto C como salida, lo utilizaremos para
;manejar los motores.

movlw b'00001111'
movwf TRISD ;Configuramos los bits RD0:RD3 como entrada (directivas de
;movimiento) y el resto como salidas (los bits RD4:RD6 se
;utilizan para reportarle a la placa condiciones positivas en
;los sensores).

call banco0 ;Nos posicionamos en el banco 0
clrf INTCON ;Deshabilitamos todas las interrupciones
clrf PORTD ;Borra el puerto D
clrf PORTC ;Borra el puerto C
bsf PORTB,1 ;Restea flip-flop asociado al sensor derecho
bsf PORTB,2 ;Restea flip-flop asociado al sensor izquierdo
clrf PORTB ;Borra el puerto B
clrf PORTA ;Borra el puerto A

clrf CONT ;Pongo en cero el contador
clrf TEMP ;Pongo en cero la variable temporal
clrf Tipoint ;Pongo en cero la variable para el tipo de interrupción
movlw b'00001010'
movwf PASOS ;Seteo en 7 la cantidad de pasos (Es necesario ajustarlo)
call banco1
movlw 0x06
movwf ADCON1 ;Seteo puerto analógico como digital
```



```
bsf    OPTION_REG,6      ;Configuramos flanco ascendente para la interrupción
                                ;en RB0.

call   banco0
clrf   PORTB
bcf    INTCON,RBIE
movf   PORTB,W
movlw  b'10011000'

movwf  INTCON
return
```

AtencionInterrupcion

```
call   banco0
bcf    PORTD,7          ;DEBUG
bcf    INTCON,GIE
bcf    INTCON,RBIE
bcf    INTCON,INTE
clrf   PORTC
bsf    PIR1,1
btfsr  INTCON,RBIF
goto  AtencionSensores ;La interrupción fue por un cambio en RB4:RB7?
                        ;Si, entonces voy a la rutina de atención
                        ;correspondiente
btfsr  INTCON,INTF
                        ;No, entonces me fijo si la interrupción fue porque la
                        ;placa BB va a enviar una directiva.

goto  RecibirDirectiva ;Si, entonces veo qué directiva me envió la placa
                        ;;No, entonces vuelvo a habilitar interrupciones
                        ;globales,
bsf    INTCON,RBIE
bsf    INTCON,INTE
retfie
```

;y regreso a donde estaba antes de la interrupción

AtencionSensores

```
call   banco0
movf   PORTB,W
movwf  TEMP
bcf    INTCON,RBIF
                        ;Leo el puerto B
                        ;Pongo lo leído en la variable temporal
                        ;Pongo en cero el flag correspondiente a la
                        ;interrupción RB port change
btfsr  TEMP,4
goto  Tipo4
btfsr  TEMP,5
                        ;Me fijo si fue el Sensor de obstáculos Derecho
                        ;Si, entonces voy a Tipo4
                        ;No, entonces me fijo si fue el Sensor de obstáculos
                        ;Izquierdo
goto  Tipo5
btfsr  TEMP,6
goto  Tipo6
                        ;Si, entonces voy a Tipo5
                        ;No, entonces me fijo si fue el Sensor IR
                        ;Si, entonces voy a Tipo6

bsf    INTCON,RBIE
                        ;No, entonces vuelvo a habilitar la interrupción RB
                        ;port change,
bsf    INTCON,INTE
bsf    INTCON,GIE
retfie
```

;y regreso a donde estaba antes de la interrupción



Tipo4

```
bsf    Tipoint,4      ;Seteo el flag 4 de la variable Tipoint
bsf    PORTB,1        ;Envío Reset al flip-flop
bcf    PORTB,1        ;Apago el reset del flip-flop
bsf    INTCON,RBIE    ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE     ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

Tipo5

```
bsf    Tipoint,5      ;Seteo el flag 5 de la variable Tipoint
bsf    PORTB,2        ;Envío Reset al flip-flop
bcf    PORTB,2        ;Apago el reset del flip-flop
bsf    INTCON,RBIE    ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE     ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

Tipo6

```
bsf    Tipoint,6      ;Seteo el flag 6 de la variable Tipoint
bsf    INTCON,RBIE    ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE     ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

RecibirDirectiva

```
call   banco0
movf   PORTB,W
bcf    INTCON,INTF    ;Pongo en cero el flag correspondiente a la
                      ;interrupción externa de RB0
btfsr  PORTD,0        ;Testeo si me enviaron Adelante
goto   Tipo0           ;Si, entonces voy a Tipo0
btfsr  PORTD,1        ;No, entonces testeo si me enviaron Atrás
goto   Tipo1           ;Si, entonces voy a Tipo1
btfsr  PORTD,2        ;No, entonces testeo si me enviaron Derecha
goto   Tipo2           ;Si, entonces voy a Tipo2
btfsr  PORTD,3        ;No, entonces testeo si me enviaron Izquierda
goto   Tipo3           ;Si, entonces voy a Tipo3
bsf    INTCON,RBIE    ;No, entonces vuelvo a habilitar la interrupción RB
                      ;port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE     ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

Tipo0

```
bsf    Tipoint,0      ;Seteo el flag 0 de la variable Tipoint
bsf    INTCON,RBIE    ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE     ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```



Tipo1

```
bsf    Tipoint,1      ;Seteo el flag 1 de la variable Tipoint
bsf    INTCON,RBIE   ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE    ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

Tipo2

```
bsf    Tipoint,2      ;Seteo el flag 2 de la variable Tipoint
bsf    INTCON,RBIE   ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE    ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

Tipo3

```
bsf    Tipoint,3      ;Seteo el flag 3 de la variable Tipoint
bsf    INTCON,RBIE   ;Vuelvo a habilitar la interrupción RB port change,
bsf    INTCON,INTE    ;interrupcion externa
bsf    INTCON,GIE    ;e interrupciones globales
retfie                         ;y regreso a donde estaba antes de la interrupción
```

```
*****
;Control del motor de Continua (Tracción)
;
;Para controlas el motor de continua se utilizarán dos canales de un driver L293D.
;Se conectarán 3 pines del puerto B: RC0, RC1 y RC2 a los pines Enable1, Input1 e
;Input2 del L293D respectivamente.
;Cuando RC0 esté en 1 el puente 1 del L293D estará habilitado, en caso contrario dicho
;puente quedará deshabilitado por lo que el motor quedará libre.
;Estando el puente 1 del L293D habilitado: si RC1 = 1 y RC2 = 0 el motor girará en un
;sentido (hacia adelante), si RC1 = 0 y RC2 = 1 el motor girará en sentido contrario (hacia
;atrás) y si RC1 = RC2 el motor estará detenido.
*****
```

Adelante

```
call  banco0
bsf   PORTC,0    ;Habilitamos canales del L293D (Tomamos el control del
;motor)
bsf   PORTC,1    ;Seteamos el bit RC1
bcf   PORTC,2    ;Ponemos en cero el bit RC2
return
```

SubrutinaAdelante

```
call  banco0
bcf   Tipoint,0   ;Limpio el flag de Tipoint
call  Adelante
call  Espera
call  Detener
call  LiberarMotorTraccion
goto Chequeo
```



Atras

```
call banco0
bsf PORTC,0      ;Habilitamos canales (Tomamos el control del motor)
bcf PORTC,1      ;Ponemos en cero el bit RC1
bsf PORTC,2      ;Seteamos el bit RC2
return
```

SubrutinaAtras

```
call banco0
bcf Tipoint,1    ;Limpio el flag de Tipoint
call Atras
call Espera
call Detener
call LiberarMotorTraccion
goto Chequeo
```

Detener

```
call banco0
bcf PORTC,1      ;Ponemos en cero el bit RC1
bcf PORTC,2      ;Ponemos en cero el bit RC2
return
```

LiberarMotorTraccion

```
call banco0
bcf PORTC,0      ;Deshabilitamos canales del L293B (Liberamos el motor)
return
```

;*****
;

;Control del motor Paso a Paso (Dirección)

;

;Para controlar la dirección del carro se utilizará un motor PaP bipolar. Para controlar este tipo de motores es necesario invertir las polaridades de los terminales de las bobinas en una determinada secuencia para lograr un giro en un sentido y en secuencia opuesta para que gire en el otro sentido.

;Se utilizará un driver L293D.

;Se conectará el pin RC3 a los pines Enable1 y Enable2 del L293D. Con el bit RC3 se habilitarán/deshabilitarán los puentes del L293D.

;Se conectará el pin RC4 al pin Input1 del L293D, el pin RC5 al pin Input2 del L293D, el pin RC6 al pin Input3 del L293D y el pin RC7 al pin Input4 del L293D.

;Para controlar el motor se debe tener en cuenta la siguiente tabla:

;

Num de Pasos	RC4	RC5	RC6	RC7
Paso 1	+Vcc	GND	+Vcc	GND
Paso 2	+Vcc	GND	GND	+Vcc
Paso 3	GND	+Vcc	GND	+Vcc
Paso 4	GND	+Vcc	+Vcc	GND

Derecha

```
bsf PORTC,3      ;Tomamos el control del motor
call Paso1
call EsperaPaP
call Paso2
```



```
call EsperaPaP
call Paso3
call EsperaPaP
call Paso4
call EsperaPaP
incf CONT,1      ;Incrementamos el contador
movf PASOS,0     ;Escribo la cantidad de pasos en W para que la secuencia 1-
                  ;2-3-4 se ejecute PASOS veces
subwf CONT,0      ;Le resto W a CONT y lo almaceno en W
btfsr STATUS,2    ;Me fijo si la operacion dio 0
goto Derecha      ;Si no dio 0, CONT aún no llegó a PASOS entonces repite
bcf PORTC,3       ;Si dio 0, CONT llegó a PASOS entonces libero el motor,
clr CONT          ;pongo en 0 el contador,
call LiberarMotorPaP ;libero el motor
return
```

SubrutinaDerecha

```
call banco0
bcf Tipoint,2    ;Limpio el flag de Tipoint
call Adelante
call Derecha
call Detener
call LiberarMotorTraccion
goto Chequeo
```

Izquierda

```
bsf PORTC,3       ;Tomamos el control del motor
call Paso4
call EsperaPaP
call Paso3
call EsperaPaP
call Paso2
call EsperaPaP
call Paso1
call EsperaPaP
incf CONT,1      ;Incrementamos el contador
movf PASOS,0     ;Escribo la cantidad de pasos en W para que la secuencia 4-
                  ;3-2-1 se ejecute PASOS veces
subwf CONT,0      ;Le resto W a CONT y lo almaceno en W
btfsr STATUS,2    ;Me fijo si la operacion dio 0
goto Izquierda    ;Si no dio 0, CONT aún no llegó a PASOS entonces repite
bcf PORTC,3       ;Si dio 0, CONT llegó a PASOS entonces libero el motor,
clr CONT          ;pongo en 0 el contador,
call LiberarMotorPaP ;libero el motor
return
```

Subrutinalzquierda

```
call banco0
bcf Tipoint,3    ;Limpio el flag de Tipoint
call Adelante
call Izquierda
call Detener
call LiberarMotorTraccion
```



```
        goto    Chequeo

LiberarMotorPaP
    call    banco0
    bcf    PORTC,3      ;Deshabilitamos canales del L293B (Liberamos el motor)
    return

Paso1
    bsf    PORTC,4
    bcf    PORTC,5
    bsf    PORTC,6
    bcf    PORTC,7
    return

Paso2
    bsf    PORTC,4
    bcf    PORTC,5
    bcf    PORTC,6
    bsf    PORTC,7
    return

Paso3
    bcf    PORTC,4
    bsf    PORTC,5
    bcf    PORTC,6
    bsf    PORTC,7
    return

Paso4
    bcf    PORTC,4
    bsf    PORTC,5
    bsf    PORTC,6
    bcf    PORTC,7
    return

;Rutinas de espera

Espera
    call    banco1
    movlw 0xFF
    movwf PR2    ;Escribimos 255 en PR2 para que TMR2 incremente hasta 255
    call    banco0
    clrf    TMR2      ;Pongo en 0 el registro del timer 2
    movlw b'01111011' ;Configuramos prescaler y postscaler 1:16
    movwf T2CON
    bcf    PIR1,1      ;Ponemos en cero el flag del timer2
    bsf    T2CON,2      ;Encendemos el Timer 2
    call    Test         ;Testeo si terminó la espera
    decf    Aux,1        ;Decremento Aux
    btfss   STATUS,2      ;Me fijo si Aux llegó a 0
    goto    Espera       ;Si no llegó a 0 vuelvo a esperar
    return             ;Si llegó a 0 regreso
```



EsperaPaP

```
call    banco1
movlw b'11111111' ;El período típico de conmutación del L298 son 40us
movwf PR2 ;Escribimos 255 en PR2 para que TMR2 incremente hasta 255
call    banco0
clrf    TMR2      ;Pongo en 0 el registro del timer 2
movlw b'01111011' ;Configuramos prescaler y postscaler 1:16
movwf T2CON
bcf    PIR1,1      ;Ponemos en cero el flag del timer2
bsf    T2CON,2      ;Encendemos el Timer 2
call    Test        ;Testeo si terminó la espera
return
```

Test

```
btfss  PIR1,1      ;Terminó la espera?
goto   Test         ;Si no terminó sigo esperando
bcf    PIR1,1      ;Si terminó, limpio el flag
bcf    T2CON,2      ;Apago el timer 2
return
```

;y regreso

SubrutinaObstDer

```
call    banco0
bcf    Tipoint,4    ;Limpio el flag de Tipoint
bsf    PORTD,4      ;Notifico a la placa que se detectó un obstáculo del lado
;derecho
call    EsperaNotific ;Ejecuto rutina de espera para que la placa lea el dato
bcf    PORTD,4      ;Limpio el bit RD4
goto   Chequeo
```

SubrutinaObstIzq

```
call    banco0
bcf    Tipoint,5    ;Limpio el flag de Tipoint
bsf    PORTD,5      ;Notifico a la placa que se detectó un obstáculo del lado
;izquierdo
call    EsperaNotific ;Ejecuto rutina de espera para que la placa lea el dato
bcf    PORTD,5      ;Limpio el bit RD5
goto   Chequeo
```

SubrutinaIR

```
call    banco0
bcf    Tipoint,6    ;Limpio el flag de Tipoint
bsf    PORTD,6      ;Notifico a la placa que se detectó luz IR
call    EsperaNotific ;Ejecuto rutina de espera para que la placa lea el dato
bcf    PORTD,6      ;Limpio el bit RD6
goto   Chequeo
```

EsperaNotific

```
call    banco1
movlw 0xFF
```



```
movwf PR2 ;Escribimos 255 en PR2 para que TMR2 incremente hasta 255
call banco0
clrf TMR2 ;Pongo en 0 el registro del timer 2
movlw b'01111011' ;Configuramos prescaler y postscaler 1:16
movwf T2CON
bcf PIR1,1 ;Ponemos en cero el flag del timer2
bsf T2CON,2 ;Encendemos el Timer 2
call Test ;Testeo si terminó la espera
decf Aux,1 ;Decremento la variable Aux
btfs s STATUS,2 ;Me fijo si llegó a cero
goto EsperaNotific ;Si no llegó a 0 vuelvo a esperar
return ;Si llegó a 0 regreso

banco0
bcf STATUS,RP1
bcf STATUS,RP0
return

banco1
bcf STATUS,RP1
bsf STATUS,RP0
return

end
```



12 Anexo V – Código del módulo de posicionamiento

```
//ProgGPS.h
#ifndef _ProgGPS_H
#define _ProgGPS_H

extern int errno;

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <errno.h>

/*Estructura que contiene los datos de un punto */
typedef struct Point_Data_Structure
{
    double latitude;
    double longitude;
    char *datetime;
    int found;
} Point_Data;

/* Prototipos de funciones */
int stablish();

Point_Data getWGS84Pos(int gpsd);

/*La funcion de conversion recibe Latitud y Longitud en radianes. El receptor GPS
 *entrega las coordenadas en grados. Por lo tanto, es necesario hacer la conversion entre grados y
 *radianes.*/
double toRad(double deg);

/*Convierte las coordenadas en el Datum WGS84 al Datum Yacare*/
void convert(double lat, double lon, double *northing, double *easting);

/*Imprime en pantalla informacion de fecha y hora, latitud y longitud en Datum WGS84 y Easting y
 *Northing en Datum Yacare. Ademas, almacena la info en la BD.*/
void getInfo();

#endif

//ProgGPS.c

#define PI 3.14159265358979323e0
#define MOLODENSKY_MAX (89.75 * PI / 180.0) /* Polar limit */

#define WGS84_a 6378137.0 /* Semi-eje mayor de la elipsoide WGS84 en metros */
#define WGS84_f 1/298.257223563 /* Achatamiento de la elipsoide WGS84 */

/* Parametros de la elipsoide Internacional 1924 (Utilizada por el Datum yacare.) */
#define aIN24 6378388.000 /* Semi-eje mayor de la elipsoide Internacional 1924 en metros */
#define dIN24 WGS84_a - aIN24 /* Diferencia en los semi-ejes mayores */
#define fIN24 1/297.000000000 /* Achatamiento de la elipsoide Internacional 1924 */
#define dfIN24 WGS84_f - fIN24 /* Diferencia en los achatamientos */
#define dxIN24 -154
```



```
#define      dyIN24      162
#define      dzIN24      46

/* Parametros de la proyección Mercator Transversa (usada por el Datum Yacare) */
#define      originLat     0.0
#define      centralMeridian -0.973893722
#define      falseEasting   500000.0
#define      falseNorthing  4002288.0
#define      scaleFactor    1.0
#define      MAX_LAT        ((PI * 89.99)/180.0) /* 89.99 degrees in radians */
#define      MAX_DELTA_LONG ((PI * 90)/180.0) /* 90 degrees in radians */
#define      SPHTMD(Latitude) ((double) (TranMerc_ap * Latitude \
- TranMerc_bp * sin(2.e0 * Latitude) + TranMerc_cp * sin(4.e0 * Latitude) \
- TranMerc_dp * sin(6.e0 * Latitude) + TranMerc_ep * sin(8.e0 * Latitude) ))
#define      SPHSN(Latitude) ((double) (TranMerc_a / sqrt( 1.e0 - TranMerc_es \
* pow(sin(Latitude), 2))))
#define      SPHSR(Latitude) ((double) (TranMerc_a * (1.e0 - TranMerc_es) \
/pow(DENOM(Latitude), 3)))
#define      DENOM(Latitude) ((double) (sqrt(1.e0 - TranMerc_es * pow(sin(Latitude),2)))))

#include      <sys/types.h>
#include      <sys/socket.h>
#include      <netinet/in.h>
#include      <arpa/inet.h>
#include      <netdb.h>
#include      <unistd.h>
#include      <sys/select.h>
#include      <math.h>
#include      "ProgGPS.h"

/* Ellipsoid Parameters, default to WGS 84 */
static double TranMerc_a = 6378137.0;           /* Semi-major axis of ellipsoid in meters */
static double TranMerc_f = 1 / 298.257223563;    /* Flattening of ellipsoid */
static double TranMerc_es = 0.0066943799901413800; /* Eccentricity (0.08181919084262188000)
*squared */
static double TranMerc_ebs = 0.0067394967565869; /* Second Eccentricity squared */

/* Transverse_Mercator projection Parameters */
static double TranMerc-Origin_Lat = 0.0;         /* Latitude of origin in radians */
static double TranMerc-Origin_Long = 0.0;          /* Longitude of origin in radians */
static double TranMerc_False_Northing = 0.0;       /* False northing in meters */
static double TranMerc_False_Easting = 0.0;         /* False easting in meters */
static double TranMerc_Scale_Factor = 1.0;          /* Scale factor */

/* Isometric to geodetic latitude parameters, default to WGS 84 */
static double TranMerc_ap = 6367449.1458008;
static double TranMerc_bp = 16038.508696861;
static double TranMerc_cp = 16.832613334334;
static double TranMerc_dp = 0.021984404273757;
static double TranMerc_ep = 3.1148371319283e-005;

/* Maximum variance for easting and northing values for WGS 84. */
static double TranMerc_Delta_Easting = 40000000.0;
static double TranMerc_Delta_Northing = 40000000.0;

/* These state variables are for optimization purposes. The only function
* that should modify them is Set_Transverse_Mercator_Parameters. */
typedef struct Geodetic_Tuple_Structure
```



```
{  
    double longitude; /* radians */  
    double latitude; /* radians */  
} Geodetic_Tuple;  
  
typedef struct Transverse_Mercator_Tuple_Structure  
{  
    double easting; /* meters */  
    double northing; /* meters */  
} Transverse_Mercator_Tuple;  
  
int establish(){  
    const char *host = "127.0.0.1";  
    const char *service = "2947";  
    const char *protocol = "tcp";  
    struct hostent *phe;  
    struct servent *pse;  
    struct protoent *ppe;  
    struct sockaddr_in sin;  
    int gpsd, type, proto, one, n, w = 1;  
    char descarte[1000]; /* Buffer para descartar los datos que no nos interesan. */  
  
    memset((char *) &sin, 0, sizeof(sin));  
    sin.sin_family = AF_INET;  
  
    if ((pse = getservbyname(service, protocol)))  
        sin.sin_port = htons(ntohs((unsigned short) pse->s_port));  
    else if ((sin.sin_port = htons((unsigned short) atoi(service))) == 0){  
        fprintf(stderr, "Can't get service entry, %s(%d)\n", strerror(errno), errno);  
        exit(1);  
    }  
  
    if ((phe = gethostbyname(host)))  
        memcpy((char *) &sin.sin_addr, phe->h_addr, phe->h_length);  
#ifndef S_SPLINT_S  
    else if ((sin.sin_addr.s_addr = inet_addr(host)) == INADDR_NONE){  
        fprintf(stderr, "Can't get host entry, %s(%d)\n", strerror(errno), errno);  
        exit(1);  
    }  
#endif /* S_SPLINT_S */  
  
    ppe = getprotobyname(protocol);  
    type = SOCK_STREAM;  
    proto = (ppe) ? ppe->p_proto : IPPROTO_TCP;  
  
    if ((gpsd = socket(PF_INET, type, proto)) == -1){  
        fprintf(stderr, "Can't create socket, %s(%d)\n", strerror(errno), errno);  
        exit(1);  
    }  
  
    if (setsockopt(gpsd, SOL_SOCKET, SO_REUSEADDR, (char *)&one, sizeof(one))==1) {  
        (void)close(gpsd);  
        fprintf(stderr, "Error SETSOCKOPT SO_REUSEADDR, %s(%d)\n", strerror(errno),  
                errno);  
        exit(1);  
    }  
}
```



```
if (connect(gpsd, (struct sockaddr *) &sin, sizeof(sin)) == -1) {
    (void)close(gpsd);
    fprintf(stderr, "Fallo connect, %s(%d)\n", strerror(errno), errno);
    exit(1);
}

w = write(gpsd, "\r\n", strlen("\r\n"));
if ((ssize_t)strlen("\r\n") != w) {
    (void)fprintf(stderr, "Error al enviar \\"r\\n", %s(%d)\n", strerror(errno), errno);
    exit(1);
}

n = (int)read(gpsd, descarte, sizeof(descarte)-1);

printf("En descarte hay: %s\n", descarte);

n = (int)read(gpsd, descarte, sizeof(descarte)-1);

printf("En descarte hay: %s\n", descarte);

return gpsd;

}

Point_Data getWGS84Pos(int gpsd) {
    float trys; /* Variable para intentar comunicarse con GPSD durante 10 segundos. */
    trys = 10.0;
    int n, w; /*Cantidad de caracteres leídos */
    char bufP[50]; /*Buffer para los datos de posicion */
    char bufDT[50]; /* Buffer para la fecha y la hora */
    char descarte[1500]; /* Buffer para descartar los datos que no nos interesan. */
    Point_Data punto;
    punto.found = 0;

    while(trys >= 0.0){

/* p Returns the current position in the form "GPSD, P=%f %f"; numbers are in degrees, latitude first. */
        w = write(gpsd, "p\r\n", strlen("p\r\n"));
        if ((ssize_t)strlen("p\r\n") != w) {
            (void)fprintf(stderr, "Error al enviar comando, %s(%d)\n", strerror(errno),
            errno);
            exit(1);
        }

        /* Recibe la salida del GPS Daemon. */
        n = (int)read(gpsd, bufP, sizeof(bufP)-1);

        printf("En bufP hay: %s\n", bufP);

        if (n < 0){
            fprintf(stderr, "Fallo al recibir respuesta de gpsd, %s(%d)\n",
            strerror(errno), errno);
            exit(1);
        }else{
            if(n == 0 || (strncmp(bufP,"GPSD,P=?",8)==0){
```



```
(void)sleep(0.4); /* Esperar 0.4 segundos y volver a intentar */
trys = trys - 0.4;
printf("No hay fix, volviendo a intentar\n");
}else break;

}

}

if((strncmp(bufP,"GPSD,P=?",8))==0){
    printf("El receptor no tiene una lectura de posicion\n");
    punto.found = -1;
}else{
    const char delimiter[] = "GPSD ,=";

    punto.latitude = strtod(strtok(bufP, delimiter),NULL);
    punto.longitude = strtod(strtok(NULL, delimiter),NULL);
}

/* d Returns the UTC time in the ISO 8601 format, "D=yyyy-mm-ddThh:nmm:ss.ssZ". Digits of
precision in the fractional-seconds part will vary and may be absent. Por ejemplo: GPSD,D=2010-
02-25T13:05:14.0Z */
w = write(gpsd, "\n", strlen("\n"));
if ((ssize_t)strlen("\n") != w) {
    (void)fprintf(stderr, "Error al enviar \"%s\", %s(%d)\n", strerror(errno), errno);
    exit(1);
}

n = (int)read(gpsd, descarte, sizeof(descarte)-1);

printf("En descarte hay: %s\n", descarte);

w = write(gpsd, "d\n", strlen("d\n"));
if ((ssize_t)strlen("d\n") != w) {
    (void)fprintf(stderr, "Error al enviar comando, %s(%d)\n", strerror(errno), errno);
    exit(1);
}

/* Recibe la salida del GPS Daemon.*/
n = (int)read(gpsd, bufDT, sizeof(bufDT)-1);
printf("En bufDT hay: %s\n", bufDT);

if (n < 0) {
    fprintf(stderr, "Fallo al recibir respuesta de gpsd, %s(%d)\n", strerror(errno), errno);
    exit(1);
}else if (n == 0){
    fprintf(stderr, "No se leyó nada\n");
    exit(1);
}

punto.datetime = "0000-00-00 00:00:00";
if((strncmp(bufDT,"GPSD,D=?",8))==0){
    printf("El receptor no tiene timestamp aun\n");
    punto.found = -2;
}else{
    const char delimiter[] = "GPSD ,T::";
    char *fecha;
    int hora;
    int horaUru;
    char *minutos;
```



```
char *segundos;

fecha = (char *)strtok(bufDT, delimiters);
hora = strtod((char *)strtok(NULL, delimiters), NULL);
horaUru = hora -3;
minutos = (char *)strtok(NULL, delimiters);
segundos = (char *)strtok(NULL, delimiters);
char fechaHora[20];
sprintf(fechaHora,"%s %d:%s:%s", fecha, horaUru, minutos, segundos);
punto.datetime = fechaHora;
printf("%s\n", punto.datetime);

}

return punto;
}

double toRad(double deg){
    double rad;
    rad = deg*2*PI/360;
    while(rad >= PI){
        rad -= 2*PI;
    }
    while(rad < -PI){
        rad += 2*PI;
    }
    return rad;
}

/* A continuacion se encuentra el codigo necesario para convertir las coordenadas entre el Datum
 * Yacare y el Datum utilizado por GoogleEarth (WGS84).
 * Se adapto codigo obtenido de Geotrans, un producto de la agencia nacional de inteligencia
 * geoespacial (NGA) y del centro de investigacion y desarrollo del ejercito de los Estados Unidos,
 * cuya licencia permite su reutilizacion. */

void Convert_Geodetic_To_Transverse_Mercator (double Latitude,
                                                double Longitude,
                                                double *Easting,
                                                double *Northing)

{   /* BEGIN Convert_Geodetic_To_Transverse_Mercator */

/*
 * The function Convert_Geodetic_To_Transverse_Mercator converts geodetic
 * (latitude and longitude) coordinates to Transverse Mercator projection
 * (easting and northing) coordinates, according to the current ellipsoid
 * and Transverse Mercator projection coordinates.
 *
 * Latitude      : Latitude in radians          (input)
 * Longitude    : Longitude in radians         (input)
 * Easting      : Easting/X in meters          (output)
 * Northing     : Northing/Y in meters         (output)
 */
    double c;      /* Cosine of latitude           */
    double c2;
    double c3;
    double c5;
```



```
double c7;
double dlam; /* Delta longitude - Difference in Longitude */
double eta; /* constant - TranMerc_ebs *c *c */
double eta2;
double eta3;
double eta4;
double s; /* Sine of latitude */
double sn; /* Radius of curvature in the prime vertical */
double t; /* Tangent of latitude */
double tan2;
double tan3;
double tan4;
double tan5;
double tan6;
double t1; /* Term in coordinate conversion formula - GP to Y */
double t2; /* Term in coordinate conversion formula - GP to Y */
double t3; /* Term in coordinate conversion formula - GP to Y */
double t4; /* Term in coordinate conversion formula - GP to Y */
double t5; /* Term in coordinate conversion formula - GP to Y */
double t6; /* Term in coordinate conversion formula - GP to Y */
double t7; /* Term in coordinate conversion formula - GP to Y */
double t8; /* Term in coordinate conversion formula - GP to Y */
double t9; /* Term in coordinate conversion formula - GP to Y */
double tmd; /* True Meridional distance */
double tmndo; /* True Meridional distance for latitude of origin */
double temp_Origin;
double temp_Long;

/*
 * Delta Longitude
 */
dlam = Longitude - TranMerc-Origin_Long;

if (dlam > PI)
    dlam -= (2 * PI);
if (dlam < -PI)
    dlam += (2 * PI);
if (fabs(dlam) < 2.e-10)
    dlam = 0.0;

s = sin(Latitude);
c = cos(Latitude);
c2 = c * c;
c3 = c2 * c;
c5 = c3 * c2;
c7 = c5 * c2;
t = tan (Latitude);
tan2 = t * t;
tan3 = tan2 * t;
tan4 = tan3 * t;
tan5 = tan4 * t;
tan6 = tan5 * t;
eta = TranMerc_ebs * c2;
eta2 = eta * eta;
eta3 = eta2 * eta;
eta4 = eta3 * eta;

/* radius of curvature in prime vertical */
sn = SPHSN(Latitude);
```



```
/* True Meridional Distances */
tmd = SPHTMD(Latitude);

/* Origin */
tmdo = SPHTMD(TranMerc_Origin_Lat);

/* northing */
t1 = (tmd - tmdo) * TranMerc_Scale_Factor;
t2 = sn * s * c * TranMerc_Scale_Factor / 2.e0;
t3 = sn * s * c3 * TranMerc_Scale_Factor * (5.e0 - tan2 + 9.e0 * eta \
+ 4.e0 * eta2) / 24.e0;

t4 = sn * s * c5 * TranMerc_Scale_Factor * (61.e0 - 58.e0 * tan2 \
+ tan4 + 270.e0 * eta - 330.e0 * tan2 * eta + 445.e0 * eta2 \
+ 324.e0 * eta3 - 680.e0 * tan2 * eta2 + 88.e0 * eta4 \
- 600.e0 * tan2 * eta3 - 192.e0 * tan2 * eta4) / 720.e0;

t5 = sn * s * c7 * TranMerc_Scale_Factor * (1385.e0 - 3111.e0 * \
tan2 + 543.e0 * tan4 - tan6) / 40320.e0;

*Northing = TranMerc_False_Northing + t1 + pow(dlam,2.e0) * t2 \
+ pow(dlam,4.e0) * t3 + pow(dlam,6.e0) * t4 \
+ pow(dlam,8.e0) * t5;

/* Easting */
t6 = sn * c * TranMerc_Scale_Factor;
t7 = sn * c3 * TranMerc_Scale_Factor * (1.e0 - tan2 + eta) / 6.e0;
t8 = sn * c5 * TranMerc_Scale_Factor * (5.e0 - 18.e0 * tan2 + tan4 \
+ 14.e0 * eta - 58.e0 * tan2 * eta + 13.e0 * eta2 + 4.e0 * eta3 \
- 64.e0 * tan2 * eta2 - 24.e0 * tan2 * eta3) / 120.e0;
t9 = sn * c7 * TranMerc_Scale_Factor * (61.e0 - 479.e0 * tan2 \
+ 179.e0 * tan4 - tan6) / 5040.e0;

*Easting = TranMerc_False_Easting + dlam * t6 + pow(dlam,3.e0) * t7 \
+ pow(dlam,5.e0) * t8 + pow(dlam,7.e0) * t9;

} /* END OF Convert_Geodetic_To_Transverse_Mercator */

void Set_Transverse_Mercator_Parameters(double a,
                                         double f,
                                         double Origin_Latitude,
                                         double Central_Meridian,
                                         double False_Easting,
                                         double False_Northing,
                                         double Scale_Factor)

{ /* BEGIN Set_Transverse_Mercator_Parameters */
/*
 * The function Set_Transverse_Mercator_Parameters receives the ellipsoid
 * parameters and Transverse Mercator projection parameters as inputs, and
 * sets the corresponding state variables.
 *
 * a : Semi-major axis of ellipsoid, in meters (input)
 * f : Flattening of ellipsoid (input)
 * Origin_Latitude : Latitude in radians at the origin of the projection (input)
 * Central_Meridian : Longitude in radians at the center of the projection (input)
 * False_Easting : Easting/X at the center of the projection (input)
 */
```



```
* False_Northing : Northing/Y at the center of the projection (input)
* Scale_Factor : Projection scale factor (input)
*/
double tn; /* True Meridional distance constant */
double tn2;
double tn3;
double tn4;
double tn5;
double dummy_northing;
double TranMerc_b; /* Semi-minor axis of ellipsoid, in meters */
double inv_f = 1 / f;

TranMerc_a = a;
TranMerc_f = f;
TranMerc-Origin_Lat = Origin_Latitude;
TranMerc-Origin_Long = Central_Meridian;
TranMerc_False_Northing = False_Northing;
TranMerc_False_Easting = False_Easting;
TranMerc_Scale_Factor = Scale_Factor;

/* Eccentricity Squared */
TranMerc_es = 2 * TranMerc_f - TranMerc_f * TranMerc_f;
/* Second Eccentricity Squared */
TranMerc_ebs = (1 / (1 - TranMerc_es)) - 1;

TranMerc_b = TranMerc_a * (1 - TranMerc_f);
/*True meridional constants */
tn = (TranMerc_a - TranMerc_b) / (TranMerc_a + TranMerc_b);
tn2 = tn * tn;
tn3 = tn2 * tn;
tn4 = tn3 * tn;
tn5 = tn4 * tn;

TranMerc_ap = TranMerc_a * (1.e0 - tn + 5.e0 * (tn2 - tn3)/4.e0 \
+ 81.e0 * (tn4 - tn5)/64.e0 );
TranMerc_bp = 3.e0 * TranMerc_a * (tn - tn2 + 7.e0 * (tn3 - tn4) \
/8.e0 + 55.e0 * tn5/64.e0 )/2.e0;
TranMerc_cp = 15.e0 * TranMerc_a * (tn2 - tn3 + 3.e0 * (tn4 - tn5 )/4.e0 )/16.0;
TranMerc_dp = 35.e0 * TranMerc_a * (tn3 - tn4 + 11.e0 * tn5 / 16.e0) / 48.e0;
TranMerc_ep = 315.e0 * TranMerc_a * (tn4 - tn5) / 512.e0;

Convert_Geodetic_To_Transverse_Mercator(MAX_LAT,
MAX_DELTA_LONG + Central_Meridian,
&TranMerc_Delta_Easting,
&TranMerc_Delta_Northing);

Convert_Geodetic_To_Transverse_Mercator(0,
MAX_DELTA_LONG + Central_Meridian,
&TranMerc_Delta_Easting,
&dummy_northing);
TranMerc_Delta_Northing++;
TranMerc_Delta_Easting++;

} /* END of Set_Transverse_Mercator_Parameters */
```

/* La función convert pasa las coordenadas entregadas por el receptor GPS (en el Datum WGS84)



* a coordenadas en el Datum Yacare. */

```
void convert(double lat, double lon, double *northing, double *easting)
{
    Set_Transverse_Mercator_Parameters(alN24, fIN24, originLat, centralMeridian,
                                       falseEasting, falseNorthing, scaleFactor);

    Transverse_Mercator_Tuple coord;

    Convert_Geodetic_To_Transverse_Mercator(lat, lon, &(coord.easting), &(coord.northing));

    *northing = coord.northing;
    *easting = coord.easting;
}

//pruebaGPS
#include      "ProgGPS.h"

void main(){
    double yacEasting;
    double yacNorthing;

    Point_Data posActual;
    posActual = getWGS84Pos();
    printf("Fecha y hora: %s\n", posActual.datetime);
    printf("En WGS84: La latitud en grados es %f y la longitud en grados es %f\n",
           posActual.latitude, posActual.longitude);
    double WGS84latRad = toRad(posActual.latitude);
    double WGS84longRad = toRad(posActual.longitude);
    printf("En WGS84: La latitud en radianes es %f y la longitud en radianes es %f\n",
           WGS84latRad, WGS84longRad);
    convert(WGS84latRad, WGS84longRad, &yacNorthing, &yacEasting);
    printf("En Yacare: El easting (Coord en X) es %f y el northing (Coord en Y) es %f\n",
           yacEasting, yacNorthing);
}
```



13 Anexo VI – Procedimientos en LAMPP

13.1 *Instalación de LAMPP*

La instalación de LAMPP consiste simplemente en descargar un archivo .tar.gz de la página www.apachefriends.org y descomprimirlo en el directorio /opt. Puede ser necesario modificar algún archivo de configuración si se desea personalizar el comportamiento de LAMPP, pero en el caso de este proyecto no fue necesario. A continuación se explican en mayor detalle los pasos a seguir para instalar y ejecutar LAMPP:

1. **Descarga del archivo .tar.gz.** Al momento de realizar este proyecto, la versión más reciente de LAMPP es la versión 1.7.3. El archivo .tar.gz correspondiente a esta versión se puede obtener en la siguiente URL: <http://www.apachefriends.org/download.php?xampp-linux-1.7.3a.tar.gz> (disponible 23 de febrero de 2010).
2. **Instalación.** Para poder descomprimir el archivo en el directorio /opt es necesario estar loggeado como usuario root, por lo que puede ser necesario invocar el comando *su* y luego ingresar la contraseña de root. Luego, debe descomprimirse y desempaquetarse el archivo .tar.gz en al directorio /opt. Esto puede lograrse ejecutando el siguiente comando:

```
tar xvfz xampp-linux-1.7.3a.tar.gz -C /opt
```

3. **Ejecución de LAMPP.** Para iniciar LAMPP basta con invocar el siguiente comando:

```
/opt/lampp/lampp start
```

Luego de invocar el comando anterior, si no hubo ningún error, debería verse en pantalla lo siguiente:

```
Starting XAMPP 1.7.3a...
LAMPP: Starting Apache...
LAMPP: Starting MySQL...
LAMPP started.
```

4. **Verificación.** Para verificar que todo funciona correctamente basta con ingresar la URL <http://localhost> en algún navegador web. Debería verse una pantalla similar a la que se muestra a continuación.

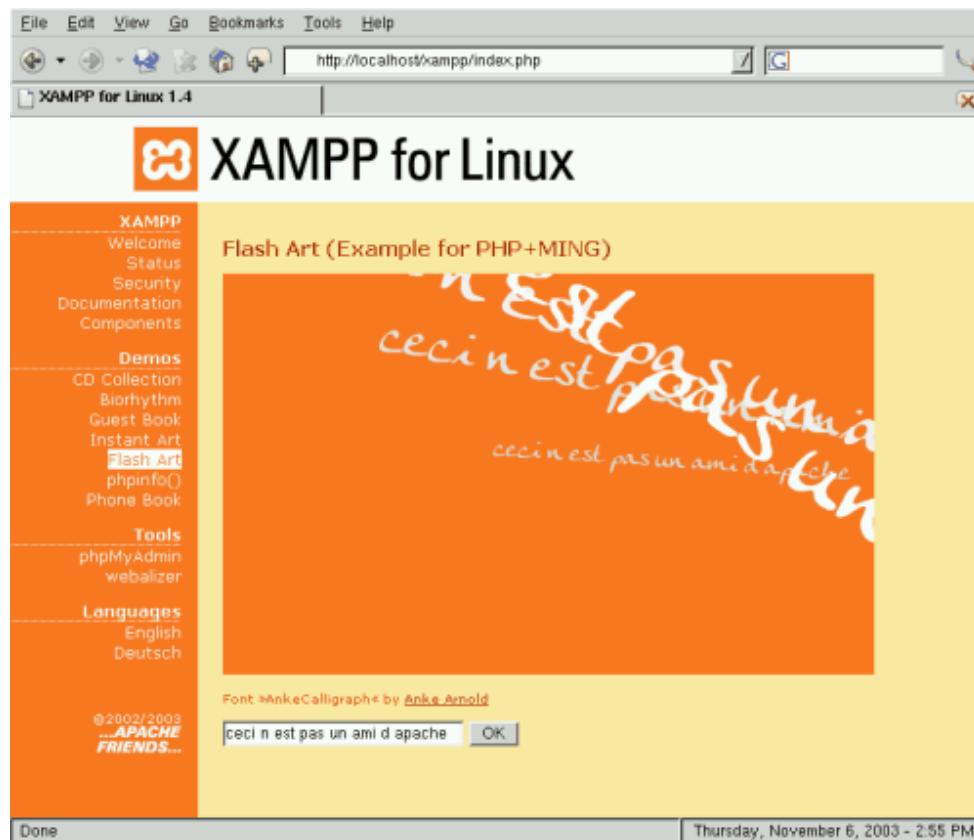


Figura 13.1 - LAMPP

13.2 Creación de la base de datos

Para acceder a phpMyAdmin es necesario ingresar la URL: <http://localhost/phpmyadmin/> en la barra de dirección de un navegador Web. Se podrá ver una pantalla similar a la que se muestra a continuación.

Figura 13.2 - Pantalla principal phpMyAdmin



Para crear una nueva base de datos, basta con ingresar el nombre que se desea darle en el campo “Crear nueva base de datos” y luego hacer clic en el botón **Crear**.

The screenshot shows the phpMyAdmin interface for MySQL localhost. In the central panel, there's a form titled "Crear nueva base de datos" (Create new database) with the input field containing "UMix". Below it, a dropdown menu shows "Cotejamiento de las conexiones MySQL: utf8_general_ci". At the bottom of the form, there's a "Crear" (Create) button. To the right of the form, there's a "MySQL" sidebar with server information and a "Servidor web" sidebar with Apache and PHP details. The bottom left corner of the interface says "Listo".

Figura 13.3 - Nueva base de datos

Luego de hacer clic en el botón **Crear**, aparecerá una pantalla similar a la que se muestra a continuación, confirmando que la base de datos se creó con éxito y mostrando la sentencia SQL que se ejecutó:

The screenshot shows the phpMyAdmin interface after creating the database "UMix". The main message is "La base de datos UMix se creó." (The database UMix was created.) with a green checkmark. Below it, the SQL command "CREATE DATABASE UMix;" is displayed. At the bottom, there's a form titled "Crear nueva tabla en la base de datos UMix" (Create new table in database UMix) with fields for "Nombre:" and "Número de campos:". The bottom left corner says "Listo".

Figura 13.4 - Base de datos creada

En esta misma pantalla, phpMyAdmin notifica que no hay tablas en la base de datos y da la opción de crear una tabla, para lo cual se debe ingresar el nombre de la misma y la



cantidad de campos que tendrá. Se procede a crear la tabla *Puntos*, ingresando su nombre en el campo “Nombre” y el número 5 en el campo “Número de campos”.

The screenshot shows the phpMyAdmin interface on a local host. The main menu includes Archivo, Editar, Ver, Historial, Marcadores, Herramientas, and Ayuda. The address bar shows http://localhost/phpmyadmin/. The left sidebar lists 'Más visitados' and 'Getting Started'. The main area shows 'Servidor: localhost > Base de datos: UMix'. Under 'Operaciones', there is a green message: '✓ La base de datos UMix se creó.' Below it, a SQL command is shown: 'CREATE DATABASE `UMix` ;'. A sub-menu for 'Crear nueva tabla en la base de datos UMix' is open, with 'Nombre: Puntos' and 'Número de campos: 5' selected. A 'Continuar' button is visible at the bottom right of the sub-menu.

Figura 13.5 - Nueva tabla

Luego, se debe hacer clic en el botón “Continuar” para proceder con la creación de la tabla. Se mostrará una pantalla en la que debe ingresarse la información de los campos de la tabla.

The screenshot shows the phpMyAdmin interface on a local host. The main menu and address bar are the same as in Figure 13.5. The main area shows 'Servidor: localhost > Base de datos: UMix > Tabla: Puntos'. The table structure is displayed with five rows. The columns are 'Campo', 'Tipo', 'Longitud/Valores*', and 'Predeterminado'. All fields are set to 'INT' type and 'None' as the default value. Below the table, there are sections for 'Comentarios de la tabla:', 'Motor de almacenamiento:', and 'Cotejamiento:'. The 'Motor de almacenamiento:' dropdown is set to 'MyISAM'. A note at the bottom says 'definición de la PARTICIÓN:'. The status bar at the bottom left says 'Listo'.

Figura 13.6 - Campos de la tabla

Como se mencionó en la sección Diseño de la base de datos, la tabla Puntos tendrá 5 campos: Identificador, Latitud, Longitud, Timestamp y Encuentro.



Para el campo “Identificador” se provee la siguiente información:

- Nombre: Identificador
- Tipo: INT
- Índice: Primario
- A_I (Auto Increment)
- Comentario: Identificador de los puntos almacenados.

Para los demás campos solo se completa la información de nombre, tipo y comentario según se detalla en la siguiente tabla:

Nombre	Tipo	Comentario
Latitud	DOUBLE	Coordenada de latitud de la posición del móvil.
Longitud	DOUBLE	Coordenada de longitud de la posición del móvil.
Timestamp	DATETIME	Fecha y hora de la posición del móvil.
Encontro	TINYINT	Flag que indica si se detectó un superviviente en este punto.

Tabla 13-1 - Campos de la tabla Puntos

La información que no es provista se completa con los valores por defecto.

A continuación se muestra la pantalla con la información de los campos de la tabla:

The screenshot shows the phpMyAdmin interface for the 'UMix' database. On the left, there's a sidebar with 'Base de datos' set to 'UMix'. The main area displays the 'Servidor: localhost > Base de datos: UMix > Tabla: Puntos' screen. It shows a table with four columns: 'Campo', 'Tipo', 'Longitud/Valores', and 'Predeterminado'. The rows correspond to the fields defined in the 'Campos de la tabla' table above. Below the table, there are sections for 'Comentarios de la tabla:', 'Motor de almacenamiento:' (set to MyISAM), and 'Cotejamiento:'. At the bottom, there's a 'definición de la PARTICIÓN:' section and a 'Listo' button.

Figura 13.7 - Información de los campos

Luego, es necesario hacer clic en el botón **Grabar** que se encuentra más abajo en la misma pantalla. Se mostrará otra pantalla confirmando que la operación fue exitosa y mostrando la sentencia SQL ejecutada.



The screenshot shows the phpMyAdmin interface with the following details:

- Servidor:** localhost
- Base de datos:** UMix
- Tabla:** Puntos
- SQL Query:**

```
CREATE TABLE `UMix`.`Puntos` (
  `Identificador` INT NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT 'Identificador de los puntos almacenados',
  `Latitud` DOUBLE NOT NULL COMMENT 'Coordenada de latitud de la posición del móvil',
  `Longitud` DOUBLE NOT NULL COMMENT 'Coordenada de longitud de la posición del móvil',
  `Timestamp` DATETIME NOT NULL COMMENT 'Fecha y hora de la posición del móvil',
  `Encuentro` TINYINT NOT NULL COMMENT 'Flag que indica si se detectó un superviviente en este punto.'
) ENGINE = MYISAM ;
```
- Table Structure:**

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
Identificador	int(11)			No	None	auto_increment
Latitud	double			No	None	
Longitud	double			No	None	
Timestamp	datetime			No	None	
Encuentro	tinyint(4)			No	None	
- Buttons:** Vaciar, Eliminar, Editar, Crear código PHP.
- Bottom Buttons:** Continuar, Índices.

Figura 13.8 - Campos agregados

13.3 Obtención del script de creación de la base de datos

Para poder importar la base de datos creada al servidor definitivo es necesario obtener el script de creación de la base de datos. Para esto se debe acceder al tab “Exportar” desde la página principal.

The screenshot shows the phpMyAdmin interface with the following details:

- Servidor:** localhost
- Bases de datos:** UMix (1)
- Estado actual:** MySQL local host
- Acciones:**
 - Crear nueva base de datos
 - Cotejamiento de las conexiones MySQL: utf8_general_ci
- MySQL:**
 - Servidor: MySQL host info: Localhost via UNIX socket
 - Versión del servidor: 5.1.41
 - Versión del protocolo: 10
 - Usuario: root@localhost
 - Juegos de caracteres de MySQL: UTF-8 Unicode (utf8)
- Servidor web:**
 - Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l PHP/5.3.1 mod_apreq2-20090110/2.7.11 mod_perl/2.0.4 Perl/v5.10.1
 - Versión del cliente: mysqld 5.0.5-dev - 081106 - SRevision: 289630 \$ extensión PHP: mysqli
- phpMyAdmin:**
 - Acerca de esta versión: 3.2.4
 - Documentación
 - Wiki
 - Página oficial de phpMyAdmin
 - [ChangeLog] [Subversion] [Lists]

Figura 13.9 - Tab Exportar



Luego de hacer clic en el tab “Exportar” se podrá ver una pantalla similar a la siguiente:

Figura 13.10 - Exportar la base de datos

En esta pantalla se deberá seleccionar la base de datos que se desea exportar y configurar algunas opciones. En este caso, se seleccionó la base de datos UMix y se deshabilitó la sección “Datos” debido a que solo se busca exportar la estructura de la base de datos.

Luego se debe hacer clic en el botón **Continuar** y aparecerá una ventana de descarga similar a la que se muestra a continuación:



Figura 13.11 - Descargar archivo

Se deberá seleccionar la opción “Guardar archivo” y luego hacer clic en el botón **Aceptar**.

Se podrá ver otra pantalla en la que se debe seleccionar el nombre con el que se desea guardar el archivo y el destino en el que se desea almacenar el mismo.



Figura 13.12 - Nombre y destino del archivo

Una vez seleccionados el nombre y el destino del archivo, basta con hacer clic en el botón **Guardar**.

En el “Anexo VII – Script de creación de la base de datos” se adjunta el script que se obtuvo.

13.4 Creación de usuario

Por cuestiones de seguridad, para no tener que utilizar el usuario root, fue necesario crear un usuario para trabajar sobre esta tabla. Esto es posible accediendo al tab “Privilegios” en la pantalla principal.

The screenshot shows the phpMyAdmin interface with the following details:

- Header:** Archivo, Editar, Ver, Historial, Marcadores, Herramientas, Ayuda.
- Address Bar:** http://localhost/phpmyadmin/
- Sidebar:** Shows databases: UMix (1), cdcol (1), information_schema (28), mysql (23), phpmyadmin (8), test. A link "Seleccionar una base de datos" is present.
- Main Navigation:** Bases de datos, SQL, Estado actual, Variables, Juegos de caracteres, Motores, **Privilegios** (highlighted), Procesos, Exportar, Importar.
- MySQL localhost:** Includes fields for "Crear nueva base de datos" (with dropdown for collation: Cotejamiento), "Cotejamiento de las conexiones MySQL" (set to utf8_general_ci), and "Interfaz" settings (Language: Español - Spanish, Theme: Original, Color Change button, Font Size: 82%).
- MySQL Information:** Shows server info: Servidor: MySQL host info: Localhost via UNIX socket, Versión del servidor: 5.1.41, Versión del protocolo: 10, Usuario: root@localhost, Juegos de caracteres de MySQL: UTF-8 Unicode (utf8).
- Server web:** Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8I PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1, Versión del cliente: mysqlnd 5.0.5-dev - 081106 - \$Revision: 289630 \$ extensión PHP: mysqli.
- phpMyAdmin:** Acerca de esta versión: 3.2.4, Documentación, Wiki.

Figura 13.13 - Tab Privilegios



Al hacer clic en el tab “Privilegios” se mostrará la pantalla “Vista global de usuarios”, similar a la pantalla que se muestra a continuación:

The screenshot shows the phpMyAdmin interface for managing MySQL users. The top navigation bar includes links for Archivo, Editar, Ver, Historial, Marcadores, Herramientas, and Ayuda. The address bar shows the URL http://localhost/phpmyadmin/. The main menu has tabs for Bases de datos, SQL, Estado actual, Variables, Juegos de caracteres, Motores, Privilegios, Procesos, Exportar, and Importar. The 'Privilegios' tab is selected. Below the tabs, the title 'Vista global de usuarios' is displayed. A grid table lists users with columns for Usuario, Servidor, Contraseña, Privilegios globales, and Conceder. The table contains entries for 'cualquiera' (localhost), 'pma' (localhost), and 'root' (localhost). A note at the bottom of the table states: 'Nota: phpMyAdmin obtiene los privilegios de los usuarios 'directamente de las tablas de privilegios MySQL'. El contenido de estas tablas puede diferir de los privilegios que usa el servidor si se han realizado cambios.' At the bottom of the page, there is a section for 'Agregar un nuevo usuario' (Add new user) and a warning about deleting selected users.

Figura 13.14 - Vista global de usuarios

Luego, se debe hacer clic en “Agregar nuevo usuario” y completar los campos correspondientes con la información del usuario. A continuación se muestra una pantalla de ejemplo:

The screenshot shows the 'Agregar un nuevo usuario' (Add new user) form in the phpMyAdmin interface. The form fields include: Nombre de usuario: 'UMixUser', Servidor: 'Local' (localhost), Contraseña: 'umixuser', and Debe volver a escribir: 'umixuser'. Below the form, there are sections for 'Base de datos para el usuario' (Database for the user) with options for 'Ninguna', 'Crear base de datos con el mismo nombre y otorgue todos los privilegios', and 'Otorgue todos los privilegios al nombre que contiene comodín (username\%)'. There is also a section for 'Privilegios globales' (Global privileges) with checkboxes for various permissions under 'Datos', 'Estructura', and 'Administración' categories. A note at the bottom of this section says: 'Nota: Los nombres de los privilegios de MySQL están expresados en inglés'.

Figura 13.15 - Agregar nuevo usuario



En este caso se creó el usuario UMixUser, que va a trabajar desde el servidor local y cuya contraseña es “umixpass”. No se le asignan privilegios globales sino que se le asignan únicamente privilegios específicos para trabajar sobre la base de datos del proyecto.

Para asignar dichos privilegios específicos, se debe hacer clic en el botón **Continuar** que se encuentra más abajo en la misma pantalla. Se podrá ver una pantalla similar a la que se muestra a continuación, en la cual se confirma la creación del usuario, se muestra la sentencia SQL ejecutada y se permite editar los privilegios del usuario.

Ha agregado un nuevo usuario.
CREATE USER 'UMixUser'@'localhost' IDENTIFIED BY '*****';
GRANT USAGE ON *.* TO 'UMixUser'@'localhost' IDENTIFIED BY '*****' WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;

Editar los privilegios: Usuario 'UMixUser'@'localhost'

Privilegios globales (Marcar todos/as / Desmarcar todos)

Datos Estructura Administración

SELECT CREATE
INSERT ALTER
UPDATE INDEX
DELETE DROP
FILE CREATE TEMPORARY TABLES
SHOW VIEW
CREATE ROUTINE
ALTER ROUTINE
EXECUTE
GRANT
SUPER
PROCESS
RELOAD
SHUTDOWN
SHOW DATABASES
LOCK TABLES
REFERENCES
REPLICATION CLIENT

Figura 13.16 - Usuario creado

Se debe buscar más abajo en la pantalla, el sector “Privilegios específicos para la base de datos” e ingresar el nombre de la base de datos sobre la cual se desea que el usuario tenga privilegios. En este caso, UMix.



The screenshot shows the phpMyAdmin interface for managing privileges. In the top right, there are configuration options for queries per hour, updates per hour, connections per hour, and user connections, all set to 0. Below this, under 'Privilegios específicos para la base de datos', it shows a table with one row for the 'UMix' database, labeled 'Ninguna'. A note says 'Añadir privilegios a esta base de datos: Use el campo de texto: UMix'. Under 'Cambio de contraseña', there are fields for a new password and its confirmation, hashing method (MySQL 4.1+ selected), and a 'Generar' button. At the bottom right of each panel is a 'Continuar' button.

Figura 13.17 - Privilegios específicos

Luego de hacer clic en el botón **Continuar**, se podrá observar una pantalla similar a la siguiente, en la cual es posible asignar los privilegios:

This screenshot shows the 'Edit privileges' screen for the 'UMixUser' user. The top navigation bar includes links for Bases de datos, SQL, Estado actual, Variables, Juegos de caracteres, Motores, Privilegios (selected), Procesos, Exportar, and Importar. The main area is titled 'Editar los privilegios: Usuario 'UMixUser'@'localhost' - Base de datos UMix'. It displays three sections: 'Datos' (with checkboxes for SELECT, INSERT, UPDATE, and DELETE), 'Estructura' (with checkboxes for CREATE, ALTER, INDEX, DROP, etc.), and 'Administración' (with checkboxes for GRANT, LOCK TABLES, and REFERENCES). A note at the top states 'Nota: Los nombres de los privilegios de MySQL están expresados en inglés'. At the bottom right is a 'Continuar' button.

Figura 13.18 - Privilegios otorgados

Teniendo en cuenta que este usuario es creado para acceder a la base de datos desde la aplicación desarrollada para este proyecto, el usuario solo necesitará privilegios para



hacer consultas y modificaciones de datos. No necesitará privilegios para hacer cambios en la estructura ni necesitará privilegios para administrar la base de datos. Por lo tanto, solo se le otorgan privilegios para SELECT, INSERT, UPDATE y DELETE.

Al hacer clic en el botón **Continuar**, se puede ver una pantalla confirmando que los privilegios se actualizaron con éxito.

The screenshot shows the phpMyAdmin interface for MySQL. The top navigation bar includes Archivo, Editar, Ver, Historial, Marcadores, Herramientas, and Ayuda. The address bar shows the URL http://localhost/phpmyadmin/. Below the address bar, there are links for 'Más visitados', 'Getting Started', and 'Latest Headlines'. The main menu has tabs for Bases de datos, SQL, Estado actual, Variables, Juegos de caracteres, Motores, Privilegios, Procesos, Exportar, and Importar. The 'Privilegios' tab is selected, indicated by a green highlight. A message box at the top right says 'Ha actualizado los privilegios para 'UMixUser'@'localhost''. Below this, a SQL command is displayed: GRANT SELECT, INSERT, UPDATE, DELETE ON 'UMix' . * TO 'UMixUser'@'localhost';. At the bottom right of the message box are '[Editar]' and '[Crear código PHP]' buttons. On the left sidebar, there's a list of databases: UMix (1), cdcol (1), information_schema (28), mysql (23), phpmyadmin (8), and test. A note says 'Seleccionar una base de datos'. In the center, there's a section titled 'Editar los privilegios: Usuario 'UMixUser'@'localhost' - Base de datos UMix'. It contains three boxes: 'Datos' (with checkboxes for SELECT, INSERT, UPDATE, DELETE checked), 'Estructura' (with checkboxes for CREATE, ALTER, INDEX, DROP, etc.), and 'Administración' (with checkboxes for GRANT, LOCK TABLES, REFERENCES). A note at the top of this section says 'Nota: Los nombres de los privilegios de MySQL están expresados en inglés'. At the bottom left of the main area, there's a 'Listo' button.

Figura 13.19 - Privilegios actualizados



14 Anexo VII - Script de creación de la base de datos

A continuación se adjunta el script de creación de la base de datos que se exportó de phpMyAdmin.

```
-- phpMyAdmin SQL Dump
-- version 3.2.4
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generación: 25-02-2010 a las 11:34:18
-- Versión del servidor: 5.1.41
-- Versión de PHP: 5.3.1

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--

-- Base de datos: `UMix`
--

CREATE DATABASE `UMix` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `UMix`;

-----
-- Estructura de tabla para la tabla `Puntos`
--

CREATE TABLE IF NOT EXISTS `Puntos` (
`Identificador` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Identificador de los puntos almacenados',
`Latitud` double NOT NULL COMMENT 'Coordenada de latitud de la posicion del movil.',
`Longitud` double NOT NULL COMMENT 'Coordenada de longitud de la posicion del movil.',
`Timestamp` datetime NOT NULL COMMENT 'Fecha y hora de la posicion del movil.',
`Encontro` tinyint(4) NOT NULL COMMENT 'Flag que indica si se detecto un superviviente en este punto.',
PRIMARY KEY (`Identificador`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=73 ;

--

-- Volcar la base de datos para la tabla `Puntos`
--
```



15 Anexo VIII - Código del módulo de persistencia

```
//DBAccess.h
#ifndef _DBAccess_H_
#define _DBAccess_H_

#include <mysql.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include "ProgGPS.h"

/*Estructura que contiene un array de puntos y la cantidad de puntos en el array.*/
typedef struct Point_Data_Structure_Array
{
    Point_Data *pointsArray;
    size_t length;
} Point_Data_Array;

/*Prototipos de funciones*/
Point_Data_Array getSurvivorPoints(); /*Devuelve un array de puntos en los que se detectaron
sobrevivientes. */

void storePoint(Point_Data point);      /* Almacena un punto en la base de datos. */

Point_Data loadPointStruct(double latitude, double longitude, char *datetime, int found); /* Carga la
estructura Point_Data con los datos que se pasan. */

int deleteAllPoints(); /* Borra todos los puntos de la base de datos. */

void recorrerArrayPuntos(Point_Data_Array points); /*Recorre el array de puntos, imprimiendo el
contenido. Este metodo es solo para debug. */

#endif

//DBAccess.c
#include "DBAccess.h"
#include "ProgGPS.h"

char server[40] = "localhost";
char *user = "UMixUser";
char *password = "umixpass";
char *database = "UMix";
char *DBfile = "/home/database";

void setServer(){
    FILE *fp;
    fp = fopen(DBfile, "r");
    if(fp == NULL){
        fprintf(stderr, "Error al abrir el archivo database: %s(%d)\n", strerror(errno), errno);
        exit(1);
    }else{
        fscanf(fp, "%s", server);
        printf("%s\n", server);
        fclose(fp);
    }
}
```



```
}

Point_Data_Array getSurvivorPoints(){
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    setServer();

    conn = mysql_init(NULL);

    /* Se conecta a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
        fprintf(stderr, "Error en connect: %s\n", mysql_error(conn));
        exit(1);
    }

    /* Envia la sentencia SQL */
    char *query = "SELECT * FROM Puntos WHERE Encontro = 1";

    if (mysql_real_query(conn, query, strlen(query))) {
        fprintf(stderr, "Error al enviar consulta: %s\n", mysql_error(conn));
        exit(1);
    }

    /*Almacena el resultado.*/
    res = mysql_store_result(conn);

    /* Ve cuantos resultados devolvio la consulta */
    int count;
    count = mysql_num_rows(res);

    Point_Data_Array pointsArrayStruct;
    pointsArrayStruct.length = count;

    Point_Data *points;
    points = malloc(count*sizeof(Point_Data)); /* Reserva memoria para el array de puntos. */

    int i =0;

    /* Almacena los resultados en el array. */

    printf("En los siguientes puntos se detectaron sobrevivientes:\n");
    while ((row = mysql_fetch_row(res)) != NULL){
        points[i].latitude = strtod(row[1], NULL);
        points[i].longitude = strtod(row[2], NULL);
        points[i].datetime = row[3];
        points[i].found = (int)strtol(row[4], NULL, 0);

        printf("Latitud: %s Longitud: %s Fecha y hora: %s Encontro: %s\n", row[1], row[2],
               row[3], row[4]);

        i++;
    }

    pointsArrayStruct.pointsArray = points;

    printf("Existen %d puntos en los que se detectaron sobrevivientes\n", i);
    /* Libera la memoria utilizada para el resultado y cierra la conexion */
    mysql_free_result(res);
```



```
mysql_close(conn);

return pointsArrayStruct;
}

void storePoint(Point_Data point){
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    setServer();

    conn = mysql_init(NULL);

    /* Se conecta a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
        fprintf(stderr, "Error en connect: %s\n", mysql_error(conn));
        exit(1);
    }

    /* Envía la sentencia SQL */
    char insert[200];
    sprintf(insert, "INSERT INTO `UMix`.`Puntos` (`Identificador`, `Latitud`, `Longitud`,
        `Timestamp`, `Encontro`) VALUES (NULL, '%f', '%f', '%s', '%d')", point.latitude,
        point.longitude, point.datetime, point.found);
    printf("Fecha y hora en la consulta: %s\n", point.datetime);
    printf("%s\n", insert);
    if (mysql_real_query(conn, insert, strlen(insert))) {
        fprintf(stderr, "Error al insertar: %s\n", mysql_error(conn));
        exit(1);
    }else{
        printf("Inserto\n");
    }

    /* Chequea si se inserto correctamente en la base de datos. */
    if(!mysql_affected_rows(conn)){
        printf("No se inserto correctamente\n");
        exit(1);
    }

    /* Cierra la conexion.*/
    mysql_close(conn);
}

Point_Data loadPointStruct(double latitude, double longitude, char *datetime, int found){
    Point_Data punto;
    punto.latitude = latitude;
    punto.longitude = longitude;
    punto.datetime = datetime;
    punto.found = found;
    return punto;
}

int deleteAllPoints(){
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    setServer();

    conn = mysql_init(NULL);
```



```
/* Se conecta a la base de datos */
if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
    fprintf(stderr, "Error en connect: %s\n", mysql_error(conn));
    exit(1);
}

/* Envia la sentencia SQL */
char *delete = "DELETE FROM `UMix`.`Puntos`";

if (mysql_real_query(conn, delete, strlen(delete))) {
    fprintf(stderr, "Error al borrar: %s\n", mysql_error(conn));
    exit(1);
}

/* Obtiene la cantidad de puntos que se eliminaron. */
int quant;
quant = mysql_affected_rows(conn);

/* Cierra la conexion.*/
mysql_close(conn);

return quant;
}

void recorrerArrayPuntos(Point_Data_Array points){
    printf("La estructura contiene %d puntos\n", points.length);
    int i = 0;

    while(i < points.length){
        printf("Latitud: %f\n", points.pointsArray[i].latitude);
        printf("Longitud: %f\n", points.pointsArray[i].longitude);
        printf("Fecha y hora: %s\n", points.pointsArray[i].datetime);
        printf("Encontro: %d\n", points.pointsArray[i].found);

        i++;
    }
    printf("Se intero %d veces\n", i);
}

//pruebaDB
#include      <stdio.h>
#include      "DBAccess.h"

void main(){
    double lat = -33;
    double lon = -57.4;
    char *fechahora = "2010-02-20 21:00:00";
    int flag = 1;

    Point_Data testPoint;
    testPoint = loadPointStruct(lat, lon, fechahora, flag);
    storePoint(testPoint);
    Point_Data_Array points;
    points = getSurvivorPoints();
    int deleted = deleteAllPoints();
    printf("Se borraron %d puntos\n", deleted);
    recorrerArrayPuntos(points);
}
```



16 Anexo IX – Código de GPSStore

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include "ProgGPS.h"
#include "DBAccess.h"

char *flagFile = "/home/flagIR";
char *errorFile = "/home/error";
int gpsd;

void sigfun(int sig);

int main(){
    gpsd = stablish();
    (void)signal(SIGINT, sigfun);
    while(1){
        printf("Entrando en while\n");
        double yacEasting;
        double yacNorthing;
        Point_Data posActual;
        posActual = getWGS84Pos(gpsd);

        char *msj;
        if(posActual.found == -1){
            msj = "El GPS no tiene lectura de posicion.\n";
        }else if(posActual.found == -2){
            msj = "El GPS no tiene datetime.\n";
        }else{
            msj = "OK.\n";
            int flagIR;
            flagIR = getFlagIR();
            posActual.found = flagIR;
            storePoint(posActual);
        }

        printf("%s\n", msj);
        FILE *er;
        er = fopen(errorFile, "w");
        if(er == NULL){
            fprintf(stderr, "Error al abrir el archivo %s: %s(%d)\n", errorFile,
                   strerror(errno), errno);
            exit(1);
        }else{
            fputs(msj, er);
            fclose(er);
        }

        printf("Durmiento\n");
        sleep(5);
    }
}

void sigfun(int sig){
    close(gpsd);
    printf("Socket cerrado\n");
```



```
        (void)signal(SIGINT, SIG_DFL);
}

int getFlagIR(){
    FILE *fp;
    fp = fopen(flagFile, "r+");
    int read;
    int cero = '0';
    if(fp == NULL){
        fprintf(stderr, "Error al abrir el archivo %s: %s(%d)\n", flagFile, strerror(errno),
                errno);
        exit(1);
    }else{
        read = fgetc(fp);
        printf("%c\n", read);
        ungetc(read, fp);
        fputc(cero, fp);
        fputs("\n", fp);
        fclose(fp);
        if(read == '1'){
            return 1;
        }else{
            return 0;
        }
    }
}
```



17 Anexo X – Código de GPSConvert

```
#include "ProgGPS.h"
#include "DBAccess.h"

int main(int argc, char *argv[]){
    if(argc == 1){
        printf("Debe ingresar la latitud y la longitud como parametros.\n");
        exit(1);
    }else{
        double yacEasting;
        double yacNorthing;

        double WGS84latRad = toRad(strtod(argv[1], NULL));
        double WGS84longRad = toRad(strtod(argv[2], NULL));
        convert(WGS84latRad, WGS84longRad, &yacNorthing, &yacEasting);
        printf("%f %f\n", yacEasting, yacNorthing);
    }
}
```



18 Anexo XI – Código interfaz Web

//aplicacion.php

```
<?php
    require ('xajax_core/xajax.inc.php');
    $xajax = new xajax();
    $xajax->registerFunction("adelante");
    $xajax->registerFunction("atras");
    $xajax->registerFunction("derecha");
    $xajax->registerFunction("izquierda");

    $xajax->processRequest();
    function adelante(){
        #\$salida = "Adelante";
        $obstaculoDer = "sudo cat /sys/class/gpio/gpio133/value";
        $obstaculoIzq = "sudo cat /sys/class/gpio/gpio183/value";
        $comando = "sudo /var/www/scripts/adelante";
        shell_exec($comando);
        $resObsDer = shell_exec($obstaculoDer);
        $resObsIzq = shell_exec($obstaculoIzq);
        $salidaDer="";
        if($resObsDer == "1"){
            $salidaDer='';
        }else{
            $salidaDer='';
        }
        $salidalzq="";
        if($resObsIzq == "1"){
            $salidalzq='';
        }else{
            $salidalzq='';
        }

        $salida = $salidaDer.$salidalzq;
        $respuesta = new xajaxResponse();
        $respuesta->assign("respuesta","innerHTML",$salida);
        return $respuesta;
    }

    function atras(){
        $comando = "sudo /var/www/scripts/atras";
        $obstaculoDer = "sudo cat /sys/class/gpio/gpio133/value";
        $obstaculoIzq = "sudo cat /sys/class/gpio/gpio183/value";
        #\$salida = "Atras";

        shell_exec($comando);
        $resObsDer = shell_exec($obstaculoDer);
        $resObsIzq = shell_exec($obstaculoIzq);
        $salidaDer="";
        if($resObsDer == "1"){
            $salidaDer='';
```



```
        }else{
            $salidaDer='';
        }
        $salidalzq="";
        if($resObsIzq == "1"){
            $salidalzq='';
        }else{
            $salidalzq='';
        }

        $salida = $salidaDer.$salidalzq;
        $respuesta = new xajaxResponse();
        $respuesta->assign("respuesta", "innerHTML", $salida);
        return $respuesta;
    }

    function derecha(){
        $$salida = "Derecha";
        $comando = "sudo /var/www/scripts/derecha";
        $obstaculoDer = "sudo cat /sys/class/gpio/gpio133/value";
        $obstaculoIzq = "sudo cat /sys/class/gpio/gpio183/value";
        $$salida = "Atras";
        shell_exec($comando);
        $resObsDer = shell_exec($obstaculoDer);
        $resObsIzq = shell_exec($obstaculoIzq);
        $salidaDer="";
        if($resObsDer == "1"){
            $salidaDer='';
        }else{
            $salidaDer='';
        }
        $salidalzq="";
        if($resObsIzq == "1"){
            $salidalzq='';
        }else{
            $salidalzq='';
        }

        $salida = $salidaDer.$salidalzq;
        $respuesta = new xajaxResponse();
        $respuesta->assign("respuesta", "innerHTML", $salida);
        return $respuesta;
    }

    function izquierda(){
        $$salida = "Izquierda";
        $comando = "sudo /var/www/scripts/izquierda";
        $obstaculoDer = "sudo cat /sys/class/gpio/gpio133/value";
        $obstaculoIzq = "sudo cat /sys/class/gpio/gpio183/value";
        $$salida = "Atras";
        shell_exec($comando);
        $resObsDer = shell_exec($obstaculoDer);
        $resObsIzq = shell_exec($obstaculoIzq);
        $salidaDer="";
        if($resObsDer == "1"){

    }
```



```
        $salidaDer='';
    }else{
        $salidaDer='';
    }
$salidalzq="";
if($resObslzq == "1"){
    $salidalzq='';
}else{
    $salidalzq='';
}

$salida = $salidaDer.$salidalzq;
$respuesta = new xajaxResponse();
$respuesta->assign("respuesta","innerHTML",$salida);
return $respuesta;
}

$xajax->processRequest();
$xajax_js = $xajax->getJavascript();

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<?php

$xajax->printJavascript();
?>
<meta content="text/html; charset=windows-1250" http-equiv="Content-Type">
<meta content="Microsoft FrontPage 4.0" name="GENERATOR">
<meta content="FrontPage.Editor.Document" name="ProgId">
<title>UMIX</title>
<script type='text/javascript' src="functions.js"></script>
<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAYrdeV_Nswc
K9b4JhAiG8-
hQmGsXG_ntmPmO65ulpWnbiaP5ibxSJkGkNYs5I8TQhcKG7aEQgTumBQQ"
type="text/javascript">
</script>

<style type="text/css" media="screen">
@import url(css/bloques.css);
</style>
<link rel="stylesheet" type="text/css" href="style.css">
<style>
BODY {
    scrollbar-arrow-color:094588;
    scrollbar-shadow-color:f0f0f0;
    scrollbar-face-color:f0f0f0;
    scrollbar-highlight-color:094588;
    scrollbar-darkshadow-color:094588;
}
</style>
```



```
</head>
<body topmargin="0" leftmargin="0" bottommargin="0" rightmargin="0"
onLoad="mostrarMapa()">
<table style="width: 100%;" border="0" cellpadding="0" cellspacing="0">
<tbody>
<tr>
<td style="width: 100%; text-align: left; vertical-align: middle;"> </td>
</tr>
</tbody>
</table>
<table background="pozadinatop.gif" bgcolor="#172886" border="0" cellpadding="0"
cellspacing="0" width="100%">
<tbody>
<tr>
<td width="100%">&nbsp;</td>
</tr>
</tbody>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="118%">
<tbody>
<tr>
<td width="100%"></td>
</tr>
</tbody>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="118%">
<tbody>
<tr>
<td valign="top" width="20%"><table style="background-color: rgb(16, 27, 89); width:
198px;" border="0" cellpadding="0" cellspacing="0">
<tbody>
<tr>
<td colspan="2" width="100%"></td>
</tr>
<tr>
<td width="5%">&nbsp;</td>
<td width="95%">&nbsp;</td>
</tr>
<tr>
<td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
<td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor=
"#101b59" width="95%"><font color="#ffffff" size="2"
face="Verdana"><a href="acercade.htm"
class="lijevi">Inicio</a></font></td>
</tr>
<tr>
<td align="left" width="5%">&nbsp;</td>
<td width="95%">&nbsp;</td>
```



```
</tr>
<tr>
  <td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
  <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana"
size="2"><a href="acercade.htm" class="lijevi">Acerca de UMix</a></font></td>
</tr>
<tr>
  <td align="left" width="5%">&ampnbsp</td>
  <td width="95%">&ampnbsp</td>
</tr>
<tr>
  <td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
  <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana"
size="2"><a href="servicios.htm" class="lijevi">Servicios</a></font></td>
</tr>
<tr>
  <td align="left" width="5%">&ampnbsp</td>
  <td width="95%">&ampnbsp</td>
</tr>
<tr>
  <td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
  <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana"
size="2"><a href="soporte.htm" class="lijevi">Soporte</a></font></td>
</tr>
<tr>
  <td align="left" width="5%">&ampnbsp</td>
  <td width="95%">&ampnbsp</td>
</tr>
<tr>
  <td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
  <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana"
size="2"><a href="faq.htm" class="lijevi">Faq</a></font></td>
</tr>
<tr>
  <td align="left" width="5%">&ampnbsp</td>
  <td width="95%">&ampnbsp</td>
</tr>
<tr>
  <td align="left" width="5%"><p style="margin-left: 25px;"></p></td>
  <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor =
'#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana"
size="2"><a href="contacto.htm" class="lijevi">Contacto</a></font></td>
</tr>
<tr>
```



```
<td width="5%">&nbsp;</td>
<td width="95%">&nbsp;</td>
</tr>
<tr>
    <td style="height: 18px;" align="left" width="5%"><p style="margin-left: 25px;"></p></td>
    <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor = '#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana" size="2"><a href="aplicacion.php" class="lijevi">Aplicaci&#242;n</a></font></td>
</tr>
<tr>
    <td style="height: 15px;" width="5%">&nbsp;</td>
    <td style="height: 15px;" width="95%">&nbsp;</td>
</tr>
<tr>
    <td style="height: 18px;" align="left" width="5%"><p style="margin-left: 25px;"></p></td>
    <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor = '#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" face="Verdana" size="2"><a href="debug.php" class="lijevi">Debug</a></font></td>
</tr>
<tr>
    <td style="height: 15px;" width="5%">&nbsp;</td>
    <td style="height: 15px;" width="95%">&nbsp;</td>
</tr>
<tr>
    <td style="height: 18px;" align="left" width="5%"><p style="margin-left: 25px;"></p></td>
    <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor = '#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" size="2" face="Verdana">Links</font></td>
</tr>
<tr>
    <td style="height: 15px;" width="5%">&nbsp;</td>
    <td style="height: 15px;" width="95%">&nbsp;</td>
</tr>
<tr>
    <td style="height: 18px;" align="left" width="5%"><p style="margin-left: 25px;"></p></td>
    <td onMouseOver="this.bgColor = '#9FA3BC';" onMouseOut="this.bgColor = '#101B59';" bgcolor="#101b59" width="95%"><font color="#ffffff" size="2" face="Verdana"><a href="configuracion.php" class="lijevi">Configuración</a></font></td>
</tr>
<tr>
    <td style="height: 78px;" width="5%">&nbsp;</td>
    <td style="height: 78px;" width="95%">&nbsp;</td>
</tr>
<tr>
    <td width="30%">&nbsp;
        <p>&nbsp;</p>
        <p>&nbsp;</p>
    <td width="80%">&nbsp;
        <p>&nbsp;</p>
```



```
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p></td>
</tr>
<tr>
    <td colspan="2" width="100%"></td>
    </tr>
</tbody>
</table></td>
<td valign="top" width="80%"><table width="682" height="338" border="0"
cellpadding="0" cellspacing="0">
    <tbody>
        <tr>
            <td width="100%" height="338"><p style="margin-left: 30px;">
                <div id="map2"> </div>

                <div id="botonesMovimiento">
                    <table width="140" >
                        <tr>
                            <td colspan="2" align="center"> Controles del móvil </td>
                        </tr>
                        <tr>
                            <td align="center" colspan="2"><form method="post" action="">
                                
                                </form></td>
                            </tr>
                            <tr>
                                <td width="64" align="right"><form method="post" action="">
                                    
                                    </form></td>
                                <td width="64" align="left"><form method="post" action="">
                                    
                                    </form></td>
                            </tr>
                            <tr>
                                <td align="center" colspan="2"><form method="post" action="">
                                    
                                    </form></td>
                            </tr>
                        </table>
                    </div>
                <div id="botonesBD">
                    <table width="140" >
                        <tr>
                            <td colspan="2" align="center"><p>Controles de la Base de Datos</p></td>
                        </tr>
                        <tr>
                            <td align="center" ><form method="post" action="scripts/borrarBD.php">
                                <input type="image" src="images/trash.jpg" >
                                </form></td>
                        </tr>
                    </table>
                </div>
            </td>
        </tr>
    </tbody>
</table>
```



```
</div>
<p style="margin-left: 30px; margin-right: 30px; margin-top: 0pt;"><br>
</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>
<div id="respuesta"> </div>

</p></td>
</tr>
</tbody>
</table>
<p>&nbsp;</p>
<p style="margin-left: 10px;" align="right"><a href="http://www.rickyswebtemplates.com"></a></p>
<p style="margin-left: 10px;" align="right">Design by<a href="http://www.rickyswebtemplates.com">www.rickyswebtemplates.com</a></p></td>
</tr>
</tbody>
</table>
<center>
<span style="width: 100%; font-family: helvetica; font-size: 6px;">Design downloaded from Zeroweb.org<br>
<a href="http://www.zeroweb.org" style="font-family: helvetica; font-size: 6px;">Web templates, layouts, and website tools for FREE!</a><br>
<a href="http://urlsnip.com" style="font-family: helvetica; font-size: 6px;">Shorten URL services</a><br>
</span>
</center>
<br>
<br>
</body>
</html>
```

//functions.js

```
var map;
function mostrarMapa(){
    if (GBrowserIsCompatible()) {
        //alert("1");
        map = new GMap2(document.getElementById("map2"));
        //map.setCenter(new GLatLng(-34.9, -56.2), 20);
        /*GEvent.addListener(map, "click", function(overlay, latlng) {
            if (latlng) {
                var myHtml = "The GPoint value is: " +
                map.fromLatLngToDivPixel(latlng) + " at zoom level " + map.getZoom();
                map.openInfoWindow(latlng, myHtml);
            }
        });
    */
}
```



```
map.addControl(new GSmallMapControl());
map.addControl(new GMapTypeControl());
//alert("2");
cargarPuntos();
}

function cargarPuntos(){
if(map!=null){
    //alert("3");
    map.clearOverlays();
    GDownloadUrl("phpsqlajax_genxml.php", function(data) {
        var xml = GXml.parse(data);
        var markers =
xml.documentElement.getElementsByTagName("marker");
        var point = new GLatLng(-34.9, -56.2);
        for (var i = 0; i < markers.length; i++) {
            point = new
GLatLng(parseFloat(markers[i].getAttribute("lat")),
parseFloat(markers[i].getAttribute("lng")));
            var marker;
            var encontro = markers[i].getAttribute("encontro");
            var icon = new GIcon();
            icon.image =
"http://labs.google.com/ridefinder/images/mm_20_red.png";
            icon.shadow =
"http://labs.google.com/ridefinder/images/mm_20_shadow.png";
            icon.iconAnchor = new GPoint(6, 20);
            icon.infoWindowAnchor = new GPoint(5, 1);
            if(encontro == "0"){
                marker = new GMarker(point,{icon:icon});
            }else{
                var icongreen = new GIcon(icon,
"http://labs.google.com/ridefinder/images/mm_20_green.png");
                marker = new GMarker(point, {icon:icongreen});
            }
            map.addOverlay(marker);
            marker.bindInfoWindowHtml(markers[i].getAttribute("conv"));
        }
        map.setCenter(point);
    });
    //alert("6");
    //alert("7");
    setTimeout("cargarPuntos()", 15000);
}
}

//phpsqlajax_genxml.php
<?php
$convertir="/home/ubuntu/UMix/GPSConvert ";

```



```
$username="UMixUser";
$password="umixpass";
$database="UMix";
$myFile = "conf/database";
$fh = fopen($myFile, 'r');
$url = fread($fh, filesize($myFile));
fclose($fh);

function parseToXML($htmlStr) {
    $xmlStr=str_replace('<','&lt;',$htmlStr);
    $xmlStr=str_replace('>','&gt;',$xmlStr);
    $xmlStr=str_replace('"','"',$xmlStr);
    $xmlStr=str_replace("'",'&#39;',$xmlStr);
    $xmlStr=str_replace("&",'&',$xmlStr);
    return $xmlStr;
}

// Opens a connection to a mySQL server
$connection=mysql_connect ($url, $username, $password);
if (!$connection) {
    die('Not connected : ' . mysql_error());
}

// Set the active mySQL database
$db_selected = mysql_select_db($database, $connection);
if (!$db_selected) {
    die ('Can\'t use db : ' . mysql_error());
}

// Select all the rows in the markers table
$query = "SELECT * FROM Puntos WHERE 1";
$result = mysql_query($query);
if (!$result) {
    die('Invalid query: ' . mysql_error());
}

header("Content-type: text/xml");

// Start XML file, echo parent node
echo '<markers>';

// Iterate through the rows, printing XML nodes for each
while ($row = @mysql_fetch_assoc($result)){
    // ADD TO XML DOCUMENT NODE
    echo '<marker >';
    // echo 'name="'.parseToXML($row['name']).'"';
    // echo 'address="'.parseToXML($row['address']).'"';
    echo 'lat="'. $row['Latitud'].'"';
    echo 'lng="'. $row['Longitud'].'"';
    $latlongcon = shell_exec($convertir.$row['Latitud']." ".$row['Longitud']);
    echo 'conv="'. $latlongcon.'"';
    echo 'encontro="'. $row['Encontro'].'"';
    echo '</marker>';
}
```



```
}

// End XML file
echo '</markers>';

?>

//Scripts de movimiento

//Adelante

#!/bin/bash

INT=142
MOV=137

echo ${MOV} > /sys/class/gpio/export
echo ${INT} > /sys/class/gpio/export

echo high > /sys/class/gpio/gpio${MOV}/direction
echo high > /sys/class/gpio/gpio${INT}/direction

sleep 3

echo low > /sys/class/gpio/gpio${INT}/direction
echo low > /sys/class/gpio/gpio${MOV}/direction

//Atras

#!/bin/bash

INT=142
MOV=143

echo ${MOV} > /sys/class/gpio/export
echo ${INT} > /sys/class/gpio/export

echo high > /sys/class/gpio/gpio${MOV}/direction
echo high > /sys/class/gpio/gpio${INT}/direction

sleep 3

echo low > /sys/class/gpio/gpio${INT}/direction
echo low > /sys/class/gpio/gpio${MOV}/direction

//Derecha

#!/bin/bash

INT=142
```



MOV=136

```
echo ${MOV} > /sys/class/gpio/export
echo ${INT} > /sys/class/gpio/export

echo high > /sys/class/gpio/gpio${MOV}/direction
echo high > /sys/class/gpio/gpio${INT}/direction

sleep 3

echo low > /sys/class/gpio/gpio${INT}/direction
echo low > /sys/class/gpio/gpio${MOV}/direction
```

//Izquierda

```
#!/bin/bash

INT=142
MOV=141

echo ${MOV} > /sys/class/gpio/export
echo ${INT} > /sys/class/gpio/export

echo high > /sys/class/gpio/gpio${MOV}/direction
echo high > /sys/class/gpio/gpio${INT}/direction

sleep 3
echo low > /sys/class/gpio/gpio${INT}/direction
echo low > /sys/class/gpio/gpio${MOV}/direction
```



19 Anexo XII – Desarrollo de drivers en Linux

19.1 Introducción

Linux ofrece varias bibliotecas que permiten elaborar drivers fácilmente, y disponer de esta forma de una gran adaptabilidad a cualquier sistema.

En el caso del proyecto UMIX se pensó en desarrollar un módulo específicamente para manejar la interfaz de comunicación entre el PIC y la placa BB. De esta forma sería posible capturar las interrupciones generadas al cambiar un pin a un nivel alto (evitando estar chequeando en todo momento el estado del pin) y acceder directamente al espacio de memoria asignado a cada pin sin necesidad de utilizar ninguna facilidad del resto del sistema operativo.

19.2 Marco teórico

19.2.1 Kernel

En todo sistema Unix (y en particular en Linux), diversos procesos concurrentes atienden a diferentes tareas. Cada proceso requiere recursos del sistema, como son: capacidad de procesamiento, memoria, conexión de red y otros recursos. El kernel en sí es una cantidad de código ejecutable encargado de manejar dichos requerimientos. Aunque la distinción entre las distintas tareas del kernel no siempre está totalmente clara, el rol del kernel puede ser separado en las siguientes categorías:

- Manejo de procesos
- Manejo de memoria
- Sistemas de archivos
- Control de dispositivos
- Conexiones de red

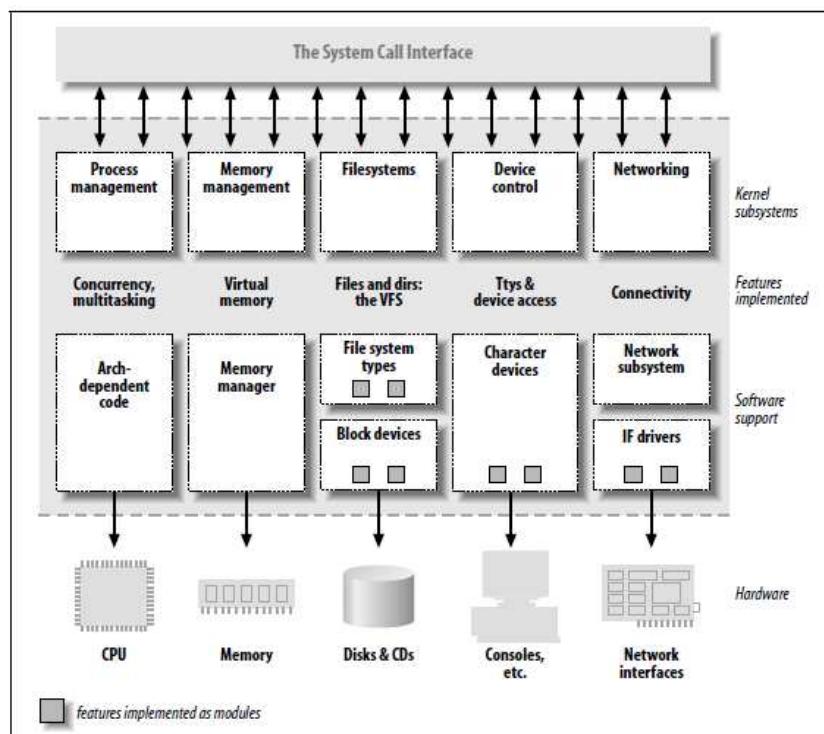


Figura 19.1 – Roles del kernel



Una de las cualidades de Linux es su capacidad para extender el conjunto de herramientas ofrecidas por el kernel durante el tiempo de ejecución. Esto significa que Linux permite agregar funcionalidades al kernel (y removérlas) mientras el sistema está levantado y ejecutando.

Cada porción de código que puede ser agregado al kernel en tiempo de ejecución se denomina “módulo” (en este documento se utilizará “driver” como sinónimo). Los módulos están formados por código de objetos (no enlazado en un ejecutable) que puede ser dinámicamente enlazado al kernel que se está ejecutando utilizando el comando **insmod** y removido mediante el comando **rmmmod**.

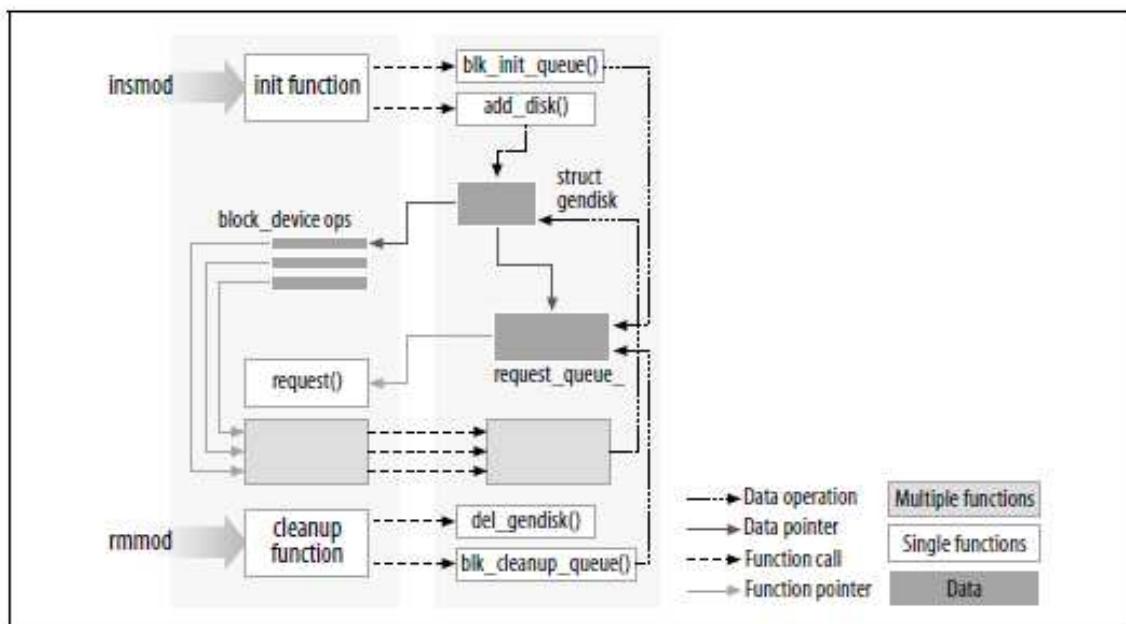


Figura 19.2 - Insmod y rmmmod

19.2.2 Módulos

19.2.2.1 Tipos de módulos

Básicamente Linux maneja 3 tipos de dispositivos diferentes que son:

- **Character devices**

Estos dispositivos son aquellos que son accedidos como una cadena de bytes (como un archivo). Un driver para estos dispositivos debe contemplar este comportamiento, típicamente implementando las funciones “abrir”, “cerrar”, “leer” y “escribir”.

- **Block devices**

Como los dispositivos carácter, los dispositivos de bloque son accedidos normalmente a través del sistema de archivos. En estos dispositivos, sin embargo, todas las operaciones se realizan mediante bloques de determinada longitud de bytes. Típicamente estos drivers se utilizan para almacenar sistemas de archivos en sí mismos (por ejemplo, representan un disco duro).

- **Network interfaces**

Todas las transacciones de red se realizan mediante una interfaz, esto es, un dispositivo que es capaz de intercambiar datos con otros hosts de la red. Usualmente una interfaz



representa un dispositivo físico, pero también puede ser puramente de software como la interfaz de loopback. Una interfaz de red está encargada de mandar y recibir paquetes de datos, generados por el subsistema de la red del kernel, sin realmente conocer cómo las transacciones individuales se mapean con los paquetes que son realmente transmitidos.

19.2.2.2 Módulos vs Aplicaciones

La utilización de módulos para resolver determinadas situaciones en vez de utilizar una aplicación puede ser muy beneficiosa, sin embargo, existen determinadas características que hay que tener en cuenta para optar por este tipo de solución.

1. Espacio de usuario vs espacio de usuario

La CPU tiene distintos modos de ejecución, con distintos privilegios. Por ejemplo x86 tiene 4 niveles de privilegios. Cuando un programa se ejecuta en User Mode tiene ciertas restricciones de acceso a memoria o hardware, en cambio, cuando una rutina se ejecuta en Kernel Mode accede a todos los recursos sin restricciones. Las diferentes CPUs proveen instrucciones especiales para pasar de User Mode a Kernel Mode y viceversa.

2. Concurrencia

La mayoría de las aplicaciones, con la excepción de las aplicaciones multihilo, se ejecutan secuencialmente sin necesidad de preocuparse sobre qué otro factor pueda estar alterando su ambiente de ejecución. En el kernel no es posible realizar dicha suposición, y hasta los más simples módulos deben tener en cuenta que muchas cosas pueden estar sucediendo al mismo tiempo y pueden estar alterando cualquier variable del ambiente.

3. El proceso actual

Aunque los módulos del kernel no se ejecutan secuencialmente como las aplicaciones, la mayoría de las acciones realizadas por el kernel son hechas en el nombre de un proceso específico. El código del kernel puede referirse a dicho proceso utilizando la variable global definida en `<asm/current.h>`, que mantiene un puntero a **struct task_struct**, definida por `<linux/Sched.h>`

19.3 Implementación

19.3.1 Código

A continuación se presenta el código utilizado para probar un módulo capaz de interceptar una interrupción generada por un cambio de estado hacia un nivel alto de una señal GPIO.

hello.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/interrupt.h>

MODULE_LICENSE("Dual BSD/GPL");

static int hello_init(void)
{
    printk(KERN_ALERT "Hello, world\n");
#define GPIO17 17
    u32 irq;
    omap_request_gpio( GPIO17 );
    irq = gpio_to_irq( GPIO17 );
```



```
gpio_direction_input( GPIO17 ); request_irq(lp->irq, hello_interrupt
,IRQF_TRIGGER_LOW,"ks8851isr", dev);
return 0;
}

irqreturn_t hello_interrupt(void){
    printk(KERN_ALERT "INTERRUPCION\n");
}
static void hello_exit(void)
{
printk(KERN_ALERT "Goodbye, cruel world\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

Makefile

```
obj-m := hello.o
```

19.3.2 Compilación y ejecución

La compilación del módulo se realiza con el siguiente comando:

```
$make ARCH=arm CROSS_COMPILE=<toolchain utilizada>
```

Se obtendrá un archivo **hello.ko**, que se puede testear en la plataforma objetivo ejecutando el siguiente comando:

```
#insmod hello.ko
```

El módulo hello.ko puede removverse con el comando:

```
#rmmod hello.ko
```

19.4 Problemas encontrados

Por alguna razón que no se pudo determinar, el código anterior reflejaba una interrupción continua, y por lo tanto se descartó su utilización.



20 Anexo XIII - Acrónimos y abreviaturas

A/D = Conversor Analógico-Digital
ALU = Arithmetic Logic Unit
API = Application Programming Interface
APT = Advanced Packaging Tool
BB = BeagleBoard
CPU = Center Processing Unit
CUS = Conductor Universal en Serie
DOM = Document Object Model
FET = Field Effect Transistor
GIE = Global Interruption Enable
GPR = General Purpose Register
GPS = Global Positioning System
GPS = Global Positioning System
GPSD = GPS Daemon
IR = Infrarrojo
IRP = Indirect Register Pointer
ISP = Internet Service Provider
LED = Light Emitting Diode
MDDR = Mobile Double Data Rate
NA = Normalmente Abierto
NC = Normalmente Cerrado
NMEA = National Marine Electronics Association
ONG = Organización No Gubernamental
PaP = Paso a Paso
PC = Program Counter
PCON = Power Control Register
POP = Package On Package
PSP = Parallel Slave Port
PWM = Pulse Width Modulation
RISC = Reduced Instruction Set Computer
SCP = Secure Copy
SFR = Special Function Register
SFTP = Secure File Transfer Protocol
SSH = Secure Shell
SSP = Synchronous Serial Port
TFTP = Trivial File Transfer Protocol
TMR2IF = Timer2 Interrupt Flag
USART = Universal Synchronous Asynchronous Receiver Transmitter
USB = Universal Serial Bus
USB-OTG = USB On-The-Go
WDT = Watch Dog Timer