UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACION DE SISTEMAS DE CONTROL PARA ROBOTS BIPEDOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRICISTA

SEBASTIAN ISAO PARRA TSUNEKAWA

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTIN

MIEMBROS DE LA COMISIÓN:
LUIS VARGAS DIAZ
PAUL VALLEJOS SANCHEZ

SANTIAGO DE CHILE
Noviembre 2008

RESUMEN DEL INFORME FINAL PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: SEBASTIAN ISAO PARRA TSUNEKAWA
FECHA: 10 de noviembre de 2008
PROFESOR GUÍA: DR. JAVIER RUIZ DEL SOLAR

DISEÑO E IMPLEMENTACION DE SISTEMAS DE CONTROL PARA ROBOTS BIPEDOS

En el presente trabajo de memoria se realiza el diseño e implementación de un sistema de control, basado en la utilización de un *Digital Signal Controller*. El objetivo es construir completamente un sistema que permita el control de un robot bípedo para ser utilizado en las competencias RoboCup por el equipo de futbol robótico ROADRUNNERS.

El diseño del sistema comienza con la identificación de los requerimientos basado en la experiencia de otros sistemas de control. Ya fijados los requerimientos, se seleccionan los componentes que los cumplan, siendo el más relevante controlador.

La implementación de lo diseñado anteriormente se realiza etapa por etapa. Se diseñan los PCB (placas de circuitos impresos) desde la descripción de los componentes hasta la fabricación mediante la máquina de construcción de prototipos LPKF. Con lo anterior se genera un prototipo para realizar las pruebas necesarias para validar el funcionamiento de un *firmware* que fue implementado para controlar el robot UCH1 Tanker.

El resultado obtenido del desempeño del prototipo permite validar los objetivos planteados en un comienzo y permite el diseño final de la versión final donde toman en cuenta las restricciones de espacio para la instalación en el robot HR18.

En conclusión, la selección del *Digital Signal Controller* TMS320F28335 y la construcción del prototipo cumplen con los objetivos planteados, permitiendo controlar efectivamente un robot bípedo. Para trabajos futuros queda la optimización del *firmware* para aprovechar todas ventajas que ofrece el TMS320F28335.

Agradecimientos

En primer lugar quiero agradecer a mis padres por dejarme elegir la Universidad de Chile como lugar de estudio y haberme apoyado mucho durante todo este tiempo.

Agradezco a Suomi por todo su apoyo durante la realización de esta memoria, ya que sin su apoyo ésta no hubiese sido posible. Aprovecho de agradecer a mis amigos: Rafael Rodríguez, Mauricio Correa, Inés Otárola, Manuel Vargas, Felipe Concha, Cesar Torres, Rodrigo Asenjo y Santiago Moreno, por todos los buenos momentos pasados durante este tiempo como alumno.

Al Profesor Javier Ruiz del Solar, que desde mecatrónica ha permitido acercarme a esta maravillosa área. A Roberto Avilés por su confianza y apoyo. Ambos me motivaron a aprender cosas que no se encuentran en la malla de la carrera.

A Paul Vallejos por ser parte de la comisión de este trabajo de titulo y ayudarme a corregir la horrenda ortografía y gramática que tengo. Así como a los demás integrantes del Laboratorio de Robótica. Y nuevamente a Rafa y Mauricio por ayudarme a corregir.

Esta memoria no podría haber sido posible sin la ayuda de Tanker, aun cuando hubieron momentos en que lo quise dejar abandonado por ahí, igual lo estimo.

Por último debo agradecer ya que el presente trabajo fue financiado por el proyecto FONDECYT 1061158.

Índice general

1. Introducción **1**

1.1. Objetivos 2

1.2. Descripción de capítulos 3

2. Contexto **4**

2.1. Robots 4

2.2. RoboCup 4

2.3. Robot Bípedo 5

2.3.1. Robot Bípedo Pasivo 5

2.3.2. Robot Bípedo Activo 6

2.3.3. Equipo UCHILE-ROADRUNNERS 7

2.4. Microcontroladores y Procesadores 7

2.4.1. Buses y Módulos 7

2.5. Sensores y Actuadores 9

2.5.1. Giróscopo 10

2.5.2. Acelerómetro 10

2.5.3. Motores Dynamixel 11

2.5.4. Brújula o compás electrónico 12

3. Diseño **13**

3.1. Plataforma HC3 y Firmware 14

3.2. Requerimientos para nueva plataforma 15

3.3. Selección del Controlador 18

3.4. Conversores RS232 y RS485 . . . 20
3.5. Conversor USB . . . 21
3.6. Puerto de expansión . . . 21
3.7. Selector entre USB y RS232 . . . 22
3.8. Selector del modo de booteo . . . 22
3.9. Oscilador y PLL . . . 24
3.10. Alimentación de Energía . . . 24

4. Implementación **25**
4.1. Construcción del Prototipo . . . 25
4.1.1. Programas Utilizados . . . 25
4.1.2. Esquemáticos . . . 27
4.1.3. Ruteos . . . 28
4.1.4. LPKF y Construcción . . . 29
4.1.5. Revisión y soldado . . . 32
4.1.6. Pruebas del prototipo . . . 33
4.2. Implementación del *firmware* . . . 34
4.2.1. Estructura del *firmware* . . . 34
4.2.2. Módulo SD . . . 38
4.2.3. Comandos Seriales y RS232-USB . . . 38
4.2.4. Servidor de Parámetros . . . 39
4.2.5. Acelerómetro Digital I2C . . . 40
4.2.6. Giróscopo . . . 40
4.2.7. Brújula . . . 41
4.2.8. Motores Dynamixel . . . 41
4.2.9. Grabación del programa en Flash . . . 42

5. Discusión de resultados **43**
5.1. Pruebas e Instalación en UCH1 - Tanker . . . 43
5.1.1. FPU vs NO FPU . . . 43
5.1.2. Motores Dynamixel . . . 43

5.1.3. Módulo tarjeta SD 44
5.1.4. Acelerómetro I2C 45
5.1.5. Brújula 45
5.1.6. ADC 45
5.1.7. Instalación y pruebas en Tanker 46
5.2. Versión Final 51

6. Conclusiones **52**

Bibliografía **55**

Capítulo 1

Introducción

En la actualidad la robótica ha tenido avances considerables, esto es gracias a las mejoras que han tenido las tecnologías asociadas a esta ciencia procesadores más poderosos, pequeños y eficientes, sensores más precisos, nuevos actuadores y precios más accesibles son algunas de las razones que han hecho posible estos avances. Gracias a estos, pronto los robots y sistemas automatizados estarán en muchos ámbitos de nuestras vidas, siendo el hogar uno de de ellos.

Para fomentar el desarrollo de la robótica, se han generado iniciativas que buscan solucionar problemas complejos pero específicos. Actualmente la Universidad de Chile tiene varios equipos que participan en la competencia RoboCup. En la gran mayoría de las categorías de esta competencia se permite la fabricación de los robots. Tal es el caso del equipo de robots humanoides, los robots actuales usados por la Universidad fueron comprados a una empresa japonesa a un elevado valor. El laboratorio de robótica tiene los medios y la capacidad para la construcción de robots, por esto se busca la construcción de robots completamente desarrollados en Chile.

La construcción de un robot implica muchas partes de desarrollo y construcción. Por un lado está la parte mecánica, que incluye la estructura y los actuadores, y por otro lado está la parte de procesamiento, compuesta por la unidad de procesamiento y los sensores. Pero estas dos partes no se pueden comunicar directamente, ya que requieren tener una interfaz entre ambos. Algo que sea capaz por un lado de controlar los actuadores y recopilar datos de los sensores, y por el otro lado recibir órdenes desde la unidad de procesamiento con la que se efectuaran acciones en los actuadores, o sea un sistema de control.

1.1. Objetivos

Parte fundamental de un robot es su sistema de control, consistente en las componentes electrónicas y de *software* necesario para su funcionamiento. Ante necesidad de nuevos robots bípedos para la competencia de futbol robótico se abre la posibilidad de construirlos completamente en el laboratorio de robótica de la Universidad de Chile. Este trabajo de memoria se enmarca en la satisfacción de parte de esta necesidad, planteando los siguientes objetivos:

- Identificar las necesidades y restricciones que tiene un sistema de control para ser utilizado en un robot bípedo en base a las observaciones realizadas a los robots bípedos humanoides dejando un margen para futuras mejoras.
- Seleccionar un procesador que sea capaz de cumplir con las necesidades y restricciones del objetivo anterior.
- Diseñar y construir un prototipo del sistema de control.
- Implementar el código necesario para comprobar las funcionalidades del sistema de control y corroborar su correcto funcionamiento.
- Instalar el módulo en un robot bípedo para realizar pruebas de funcionamiento.
- Diseñar una versión final del sistema de control, compatible con la estructura del robot HR18

1.2. Descripción de capítulos

El capítulo 2 consiste en una breve explicación de los temas con que se trabajará durante la etapa de diseño e implementación.

En el capítulo 3 se explica las etapas de diseño que se siguieron al realizar el trabajo. Se analizan las diferentes funciones que debe cumplir el sistema de control para cubrir las necesidades determinadas del análisis de otros sistemas de control.

En el capítulo 4 se muestran los pasos que se siguieron para convertir los requerimientos planteados en el capítulo 3 en un prototipo funcional. Se parte de la etapa del diseño del esquemático hasta terminar la construcción. Además se explican las consideraciones que se tuvieron al momento de programar el *firmware*.

En el capítulo 5 se realiza una discusión de los resultados obtenidos de las etapas anteriores.

En el capítulo 6 se plantean las conclusiones finales del trabajo realizado.

En anexos del proyecto, donde se presentan los esquemáticos de la versión final del sistema de control, así como las diferentes capas de las placas.

Capítulo 2

Contexto

2.1. Robots

Los robots son entidades diseñadas en su totalidad por personas y construidas por personas u otras maquinas. Estos son diseñados para tareas específicas, y pueden tomar las decisiones a partir de sensores, conocimiento previo o instrucciones dadas por personas. En la actualidad los robots se encuentran en muchos ámbitos de la vida, y se espera que en un futuro cercano estos estén aún más en la vida cotidiana. Las tareas para las cuales son diseñados son peligrosas, repetitivas, o simplemente le son imposibles de realizar a las personas.

Existen diversos tipos de robots dependiendo de su estructura física, los que se tratarán en esta memoria consisten en los antropomorfos, que significa que son similares a un humano en sus proporciones de piernas, brazos y cuerpo.

2.2. RoboCup

La RoboCup es una iniciativa internacional de investigación y educación que tiene como metas promover la inteligencia artificial y la robótica. Propone un problema estándar a superar para que todos los esfuerzos estén enfocados en una misma dirección. Por este motivo, en 1993 se introduce la idea de fútbol robótico, y así en 1995 se realizan las primeras conferencias del tema, y en 1997 se llevan a cabo las primeras competencias en Nagoya, Japón.

El gran objetivo de RoboCup es: "En el año 2050, un equipo de robots completamente autónomos

pueda ganar contra el equipo humano campeón de fútbol". Para esto, la RoboCup se divide en variadas categorías, siendo principalmente dedicadas al fútbol. Las diferentes categorías tratan de afrontar diferentes áreas de desarrollo, diversificando los esfuerzos de los desarrolladores hacia diferentes ramas de estudio y aplicaciones, buscando alcanzar la meta impuesta para el año 2050. Para ello, cada año se van modificando las reglas de cada liga para conseguir nuevos desarrollos.

Actualmente el laboratorio de Robótica de la Universidad de Chile tiene 3 equipos en esta competencia: *Standard Platform League NAO* (SPL); *Humanoid league Kid size* (HLKS) y *RoboCup@Home* (@Home).SPL consiste en partidos de futbol con robots idénticos para todos los equipos a diferencia de lo que ocurre en HLKS, donde cada equipo puede tener robots diferentes permitiendo mejoras de *hardware*. En el caso de @Home lo que se busca es la interacción hombre-robot dentro de un ambiente hogareño.

2.3. Robot Bípedo

Un robot bípedo se define como un robot que su desplazamiento se basa en el movimiento de dos piernas. Puede tener o no tener el resto del cuerpo, en caso de tener forma humana (brazos, cabeza y cuerpo) se les agrega el adjetivo de antropomórfico.

Los robots bípedos se separan en dos categorías, pasivos o activos, dependiendo de la fuente de energía que utilizan para realizar sus movimientos. La caminata humana es una mezcla de pasividad y actividad, ya que en momentos se utiliza la energía de la caída del cuerpo y luego se utilizan los músculos para llegar al siguiente paso.

2.3.1. Robot Bípedo Pasivo

El robot bípedo pasivo se basa en la utilización del potencial gravitacional para realizar desplazamientos frontales. Es como si todo el tiempo fuese cayendo. Estos robots requieren de un plano inclinado para poder utilizar la pendiente como fuente de energía. Suelen tener ninguno o pocos actuadores, los cuales no son utilizados para el desplazamiento de las extremidades.

2.3.2. Robot Bípedo Activo

El robot bípedo activo es aquel que requiere de una fuente de energía externa para su funcionamiento y sin esta energía suelen no poder estar en pie. Utilizan actuadores para generar los movimientos de sus extremidades, los cuales están en cada articulación. A continuación se describen dos modelos de robots bípedos activos que posee el laboratorio de Robótica de la Universidad de Chile.

2.3.2.1. Robot Humanoide HR18 – Forrest y Chupete

HR18 (Hajime Robot 18) [13] es un robot humanoide bípedo activo construido por Hajime Research Institute Ltd. El desempeño de este modelo de robot en la RoboCup ha demostrado que es un robot muy apto para este tipo de competencias gracias a sus características [14].

Este es un robot activo el cual posee 21 grados de libertad (DOF) de rotación, 6 en cada pierna; 3 en cada brazo, 2 en el cuello y 1 en el tronco. Un DOF corresponde a la posibilidad de movimiento entorno a un eje, siendo posible que sea una rotación o una traslación. El robot está construido con motores Dynamixel de la empresa Robotis de dos tipos: DX-117 y RX64. Sus principales características son su alto torque y bus de comunicaciones RS485 *MultiDrop*, esto permite controlarlo mediante paquetes de instrucciones de manera individual o colectivamente. Otra característica de estos motores es que su carcasa permite ser usada como estructura, facilitando la construcción y robustez del modelo permitiendo frames más livianos. El peso de este robot al momento de terminar esta memoria era de 3420 [gr] incluyendo PDA y baterías.

Su placa de control es una Hajime Controller 3 (HC3) la cual está basada en un microcontrolador Renesas de 32bits SH2/7145 a 50MHz.

2.3.2.2. Robot Humanoide UCH1 - TANKER

Es un robot bípedo activo construido completamente en el laboratorio de Robótica de la Universidad de Chile. Esta construido en aluminio y tiene 22 DOF. Tiene un DOF más que el HR18 y corresponde a la rotación del tronco con respecto a la vertical.

Fue diseñado para ser el arquero del equipo de fútbol robótico. A diferencia del HR18 es 4 [cm] más alto midiendo 56 [cm]. El peso total del robot, incluyendo PDA y baterías, al momento de terminar esta memoria era de 4195 [gr], siendo 22,6 % más pesado que el robot HR18 y de este

hecho proviene su apodo de Tanker.

Originalmente fue construido solo con motores RX28 de Dynamixel, esto lo hacía inestable. Después de la prueba realizada en esta memoria fue remodelado y se cambiaron 3 motores por RX64, ambas rodillas y el tronco.

Dentro de los alcances de esta memoria era la utilización de este robot para las pruebas del sistema de control que se diseñó.

2.3.3. Equipo UCHILE-ROADRUNNERS

Equipo de la Universidad de Chile que participa en las competencias RoboCup en la Liga humanoide. Este equipo fue formado el año 2006 para participar en esta liga desde el año 2007.

El equipo está compuesto por 3 robots, 2 HR18 (Forrest y Chupete) y 1 UCH1 (Tanker). Se utiliza como sistema de control de alto nivel una PDA N560 de la empresa Fujitsu Siemens.

2.4. Microcontroladores y Procesadores

Son los encargados de realizar el procesamiento en los sistemas de control. Están especializados en cálculos y en manejar periféricos. La principal diferencia entre un procesador y un microcontrolador es que este último posee los periféricos y memoria incorporado en un mismo circuito integrado, en cambio el procesador no los posee y se le debe agregar memoria y periféricos externamente.

Existen diversas empresas y arquitecturas disponibles de este tipo de dispositivos, dependiendo del uso que se les dé. Los procesadores poseen mayor capacidad de cálculo pero consumen más energía. Para aplicaciones embebidas típicas se suele usar microcontroladores.

2.4.1. Buses y Módulos

Para la comunicación entre diversos dispositivos existen diferentes medios físicos y protocolos, los que están bien definidos y normados para que exista compatibilidad entre los dispositivos.

2.4.1.1. SPI

Serial Peripheral Interface Bus o SPI es un protocolo de comunicación síncrona inventado por Motorola. Está basado en *shift register*, ya que dispone de una línea de entrada y una de salida, y

los datos van siendo desplazados mediante la señal de *clock* que es manejada por el *Master*. Está ideado para múltiples dispositivos que se seleccionan mediante una línea de *chip select*. Por cada dispositivo se requiere una línea de *chip select* diferente.

En total se requiere de cuatro líneas para su funcionamiento, por esto también es conocido como 4-*wires*.

2.4.1.2. I2C

Inter-Integrated Circuit o I2C es un protocolo de comunicación síncrona diseñado por Phillips. Está basado en un sistema de *pull-down* donde para transmitir se tira la línea hacia abajo. Requiere solo de 2 líneas, ya que una es el *clock* y la otra es la de *data*. La línea de *clock* sólo es manejada por el *master*. Soporta varios *slaves* donde cada uno debe tener una dirección diferente, estas direcciones son de 7 *bits* y son comunicadas mediante un paquete que define el protocolo. El *master* cada vez que quiere comunicarse con un *slave* en específico debe enviar la dirección, además de un *bit* para indicar si quiere leer o escribir en el dispositivo.

Este protocolo es muy versátil ya que permite agregar dispositivos sin tener que agregar líneas de control. Ya que solo posee dos líneas de datos es también conocido como 2-*wires*.

2.4.1.3. UART

Universal Asynchronous Receiver-Transmiter o UART corresponde a una comunicación asíncrona que solo utiliza dos líneas, una para transmitir y otra para recibir. Es un medio de comunicación muy difundido por su sencilla manera de implementar. La velocidad de transmisión debe ser previamente definida por las dos partes.

En estas comunicaciones no existe jerarquía intrínseca para la comunicación, por definición nadie controla la comunicación.

2.4.1.4. RS232

Es una norma para la interfaz de comunicación asíncrona entre dos dispositivos, definiendo rango de voltajes y función de las líneas. En su versión más reducida, se implementa UART en este medio físico permitiendo comunicaciones a mayores distancias y mayor robustez.

Esta norma está muy difundida, ya que es la usada por los puertos seriales de los computadores, donde cada línea es trasmitida con voltajes positivos y negativos.

2.4.1.5. RS485

Es una evolución del RS232 pero que a diferencia de ésta, donde cada línea es transmitida con voltajes positivos y negativos, éstas son transmitidas de manera diferencial, por lo que se tiene una línea D+ y otra D- (siendo D- la negación de D+), y el resultado de la resta entre las dos líneas es el *bit* que se transmite. Con esto se pueden implementar comunicaciones *fullduplex* (ambos individuos hablan simultáneamente) o *halfduplex* (sólo un individuo habla a la vez) dependiendo de la cantidad de líneas que se quiere utilizar. En caso de implementarse como *halfduplex* hay que considerar que dos individuos hablando en la línea provoca la destrucción de ambos mensajes.

2.4.1.6. Análogo

Algunos dispositivos tienen como salida un voltaje continuo dentro de algún rango de operación. Para poder procesar la información de tipo de salida se debe digitalizar esta señal, para esto algunos microcontroladores tienen módulos incorporados para esta conversión.

Existen diversos mecanismos para esta conversión, pero las características que los resumen son: n° de *bits* que corresponde a la resolución con la que pueden hacer la conversión y el tiempo de conversión que es cuánto tiempo se requiere para terminar la conversión.

2.4.1.7. McBSP

Multichannel Buffered Serial Port o McBSP es una interfaz de comunicación que define una comunicación síncrona *fullduplex* de alta velocidad. Está presente en algunos periféricos como conversores de audio, conversores análogo digitales, etc.

Lo definen 6 líneas, que son similares al funcionamiento del SPI, donde cada periférico maneja 3 líneas: *clock*, *receive* y *transmit*.

2.5. Sensores y Actuadores

Los efectos físicos tales como movimiento, campos magnéticos o temperatura si se quieren procesar deben ser medidos de alguna forma. Transformando la manifestación física a una respuesta eléctrica que pueda ser procesada. Para esto se requiere de un sensor específico para cada característica física.

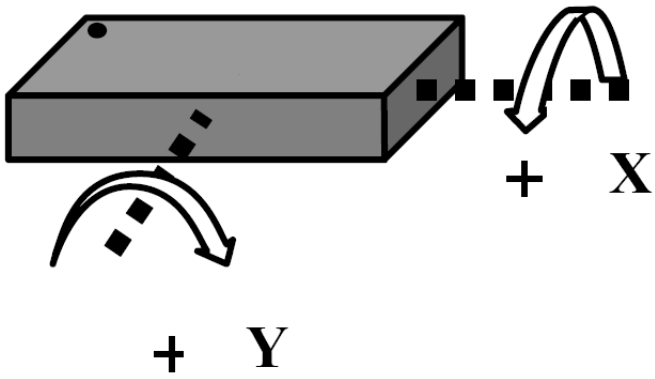


Figura 2.5.1: Dirección de los ejes de un giróscopo de 2 ejes

Los actuadores son elementos que realizan acciones sobre el mundo físico a partir de algún estímulo, los típicos actuadores son los motores eléctricos, los cuales transforman energía eléctrica en movimiento rotacional.

2.5.1. Giróscopo

El Giróscopo es un sensor cuya función es medir la velocidad angular. Existen de varios principios, pero el más común es que mediante un masa resonando se miden las fuerzas de Coriolis. La tecnología MEMS (*Micro Electro Mechanical Systems*) ha hecho posible miniaturizar estas estructuras y poder instalarlas dentro de un circuito integrado.

Un giróscopo mide la velocidad angular sobre un eje de rotación, en la figura 2.5.1 se muestra un esquema de los ejes de un giróscopo de dos ejes.

2.5.2. Acelerómetro

El acelerómetro es sensor cuya su función es medir la aceleración aplicada a él. Al igual que los giróscopos el principio se basa en la detección de las fuerzas aplicadas a una masa, y que la tecnología MEMS ha permitido miniaturizarlas e incorporarlas en circuitos integrados.

Gracias a que son capaces de medir las aceleraciones, con estos dispositivos es posible detectar la proyección de la aceleración gravedad sobre sus ejes. Esto permite saber la orientación que tiene el sensor sobre la superficie de la tierra.

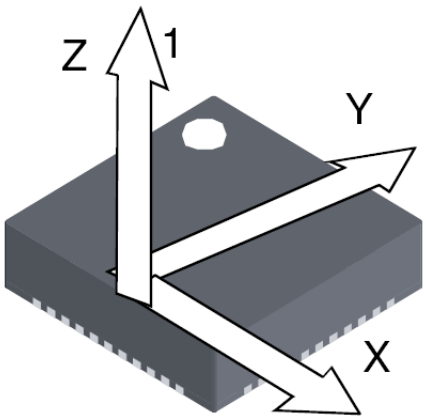


Figura 2.5.2: Dirección de los ejes de un acelerómetro de 3 ejes

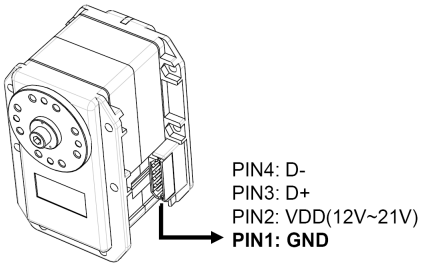


Figura 2.5.3: Motor Dynamixel RX28 con descripción de *pines*

En la figura 2.5.2 se muestra un esquema de un acelerómetro de 3 ejes. Estos ejes suelen ser ortogonales para que cada medición en un eje sea independiente de la otra.

2.5.3. Motores Dynamixel

Los motores Dynamixel son servo motores construidos por la empresa Robotis que incorpora un motor de la empresa MAXON, una caja de reducción y un controlador incorporado. La ventaja de que tenga un controlador incorporado es que todo el control de posición, velocidad y torque es manejado internamente, y sólo es necesario darle órdenes, las cuales son transmitidas mediante RS485 en los motores de las series RX y DX. [16–18]

Existen varios modelos de motores entre los cuales se diferencian principalmente por su torque máximo. La manera de controlarlos es igual para todos, esto hace que no sea importante para el controlador que modelo de motor sea.

Para la comunicación se deben formar paquetes de datos que incluyen información del identi-

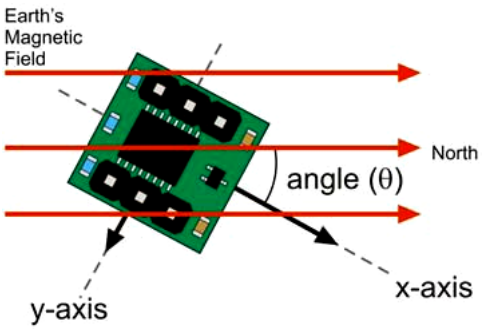


Figura 2.5.4: Compás de 2 ejes en un campo magnético

ficador del motor, comando, datos y *checksum* (para determinar errores). Además el motor puede responder con información de su estado actual, como puede ser la posición, velocidad, torque, temperatura, etc.

2.5.4. Brújula o compás electrónico

Son sensores especializados en la detección de campos magnéticos, en especial el campo magnético terrestres. Típicamente son de dos ejes mínimos y ven la proyección del vector del campo magnético sobre sus ejes. De esta manera pueden determinar la dirección del norte magnético terrestre.

Principalmente son utilizados como medios de orientación en sistemas de navegación robótica permitiendo tener una referencia de orientación global.

Capítulo 3

Diseño

La necesidad de diseñar una nueva plataforma de control de bajo nivel para robots bípedos surgió de la idea de mejorar el desempeño del equipo ROADRUNNERS, donde una necesidad era la de fabricar nuevos robots y la modificación de los robots ya existentes. Para ello no se disponía de una plataforma de control, ya que la plataforma de control HC3 que se encuentra en los robots HR18 tiene el inconveniente de ser propietaria de Hajime Research Institute Ltd. lo cual incomodaba, ya que el único proveedor esta en Japón. Por esto se buscó la construcción de una nueva plataforma de control, que permita probar nuevas ideas y avanzar en el desarrollo del equipo de fútbol robótico.

Una típica solución comercial para este tipo de problemas son las plataformas de desarrollo que permiten el uso de herramientas de *debug* sobre procesador, pero que por su generalidad, poseen más características de las necesarias en un tamaño que no permite su instalación en un robot de tamaño reducido.Además, incorpora *hardware* específico que no se es necesario y no se dispondría de algunos que sí lo son. Así es como se llegó a la solución de diseñar un sistema de control, que se ajusta a las necesidades de espacio, consumo energético, disponibilidad de periféricos y procesamiento. Además de que se posee un entendimiento de su funcionamiento para poder expandirla y usarla en otras aplicaciones, no sólo en estos robots.

Como base de comparación se utilizó el sistema de control HC3, ya que la placa que se diseñó debía ser a lo menos mejor que esta. A continuación se expone el análisis realizado a las características relevantes de HC3 y las razones de por qué de estas fueron o no fueron relevantes para el diseño.

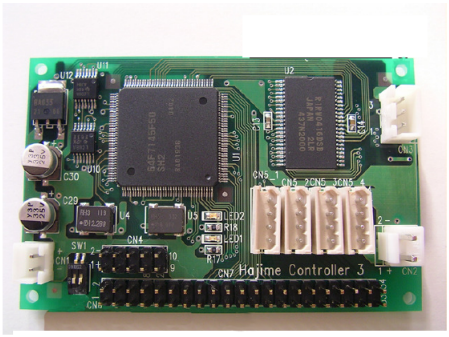


Figura 3.1.1: Hajime Controller 3

3.1. Plataforma HC3 y Firmware

Esta placa de control es la que originalmente viene con los robots HR18, basada un SH2/7145 de la compañía Renesas Technology como procesador principal, el cual es RISC de 32 *bits*, corre a 50 [MHz]. Posee 256 [kbyte] de memoria *Flash* y 8 [kbyte] de RAM interna. Posee dos conectores de energía, uno para la alimentación de los componentes electrónicos, y el otro para la alimentación de motores, que van a 5 conectores controlados por un único bus RS-485. Tiene un puerto de *pinheaders* que tiene entradas análogas y GPIOs varios. Además tiene RAM externa, 8 [Mbits] organizados en 32 *bits* (256k x 32 *bit*). [13,14]

De los periféricos del microcontrolador usa un puerto I2C para conectarse a un memoria EEPROM (AT24C1024) la cual tiene 128 [kbyte]. Ésta se utiliza para guardar los parámetros del *firmware*. El SH2/7145 tiene 4 SCI de los cuales sólo utiliza 2, uno es utilizado en conjunto con un conversor RS-232 (MAX3232E) para la comunicación del *firmware* el sistema de alto nivel de manera *fullduplex* asíncrona. El otro SCI es utilizado en conjunto con un conversor RS-485 (MAX3078EESA) para la comunicación con los motores mediante paquetes. Por la manera que esta utilizado el bus esta comunicación es *halfduplex*. Por esta razón, la CPU debe seleccionar entre leer o escribir con el conversor, selección que se realiza mediante un GPIO. Por último, se utiliza su conversor ADC de 10 *bits* para la lectura de sensores, en este caso, 3 giróscopos de 1 eje y 1 acelerómetro de 3 ejes. El ADC de este microcontrolador posee 8 canales y dos módulos de conversión independientes.

La grabación de la memoria *Flash* interna se hace mediante el puerto RS232 y para ello es necesario activar el *bootloader* del microcontrolador. Esto se hace mediante un par de *switch* su-

perficiales, que al cambiarlos de posición se modifica el *bootmode* permitiendo escribir la memoria *Flash*. La actualización del *firmware* puede ser realizada mediante la aplicación Flash Development Toolkit, utilidad que distribuye la empresa Renesas para sus dispositivos.

La placa por sí sola no puede ocupar los sensores, ya que estos tienen valores entre 0 [V] y 5 [V] y los conversores funcionan en un rango de 0 a 3.3 [V] por lo que se convierte al rango de funcionamiento del conversor. Esto se realiza en otra placa extra en la que se encuentran divisores de voltaje, y un regulador lineal (BA05FP) para generar los 5 [V] para la alimentación de los sensores.

La alimentación de todos los componentes electrónicos de la placa es de 3.3 [V] la cual está regulada por un regulador lineal (BA033FP). Este voltaje alimenta el controlador, RAM, conversores de los SCI y a la memoria EEPROM. El voltaje de entrada para esta placa está diseñado para utilizar un *pack* de baterías del tipo Ion-Litio Polímero (LiPo) con un voltaje de 8.4[V] cuando están cargadas al máximo, esto implica una pérdida grande de energía, porque el resto se disipa en el regulador. El voltaje mínimo para el correcto funcionamiento es de 6[V], en el regulador de BA05FP de los sensores se requiere eso como mínimo para tener una salida apropiada.

La forma de la placa es tal, que cabe en el interior del robot, y tiene fijaciones para poder afirmarse correctamente. La placa auxiliar para las entradas análogas también tiene orificios para fijarse al robot.

El *firmware* implementado originalmente para robots HR18 utiliza una excesiva cantidad de RAM en almacenar los movimientos especiales del robot, estos son movimientos que son cargados en la RAM y son ejecutados desde ella. Además, gran parte del cálculo de las caminatas está hecha en base a números flotantes, el problema es que el procesador no trae *hardware* para el cálculo de operaciones de este tipo de datos haciendo ineficientes las operaciones de cálculo.

3.2. Requerimientos para nueva plataforma

Como requerimiento base para una nueva plataforma era que debe tener el mismo desempeño o mejor que la anterior, por esta razón se definió una lista de requerimientos que se debían cumplir con el nuevo diseño. A continuación se listarán los distintos requerimientos y restricciones que se consideró para el diseño:

- Tamaño: La principal restricción fue el tamaño, ya que debía ser apta para ser instalada en un robot móvil, no podía ser de un tamaño excesivo, e idealmente debía instalarse en el

mismo espacio original del HR18 para ser su reemplazo. El tamaño de la placa original es aproximadamente 8 [cm] x 5 [cm] lo que fue un buen tamaño de reemplazo. Además debe disponer de orificios para su montaje, donde 4 orificios es lo mínimo necesario para una buena sujeción.

- Procesador: El punto más importante que se decide, ya que es el procesador es el que fija el desempeño y alcances que puede lograr la plataforma. Debe ser capaz de realizar cálculos eficientes para números flotantes, esto lleva a pensar en la utilización de una FPU (*Floating Point Unit*) para mejorar el rendimiento del código que requiera cálculo. Además debe poseer memoria suficiente para la ejecución del *firmware*, es decir RAM y Flash. En caso de no disponerla internamente debe poder expandirla externamente siendo agregada al diseño. Otro punto importante debe ser la documentación existente para el modelo seleccionado, ya que dependiendo de la marca seleccionada, la documentación puede variar su calidad con lo cual hace más difícil el diseño de la placa y su posterior programación.

- RS485: Debe ser capaz de comunicarse con los motores Dynamixel, debe tener un conversor de SCI a RS485. Debido al tipo de paquetes con los que se comunica con los motores, es poco eficiente la manera de respuesta de estos. Por esta razón, una buena forma de mejorar esto es agregando más buses para el control de motores, es decir, formar grupos de motores y manejarlos independientes unos de otros. De esta manera el tiempo disponible para realizar lecturas de ellos aumenta.

- RS232: Debe mantenerse la compatibilidad con la placa HC3, la cual se comunica con el sistema de alto nivel mediante RS232. El RS232 es una manera de comunicarse con dispositivos muy difundida gracias a su facilidad de implementación, ya que es soportado por la mayoría de los computadores, pero a su vez trae implicancias, debido a su antiguedad, las velocidades de comunicaciones no pueden ser muy altas.

- USB: Como no se quiere dar sólo compatibilidad hacia atrás, sino que además se busca poder usarla en diseños futuros, es útil la incorporación de un puerto de comunicaciones más veloz que el RS232 y que sea muy difundido entre los computadores, así es como surge como idea el uso de USB, ya que cumple con las exigencias anteriores, y la mayoría de las plataformas actuales las traen.

- ADC: El conversor análogo digital es necesario en la medida que se quieran hacer mediciones de sensores análogos o voltajes de interés. Además los sensores originales del robot HR18 tienen todas salidas análogas, por lo que los ADC son necesarios. Al menos se debe cumplir con 6 canales para la digitalización, y mínimo una conversión de 10 *bits*. La actual tendencia de los sensores es tratar de no depender de estos módulos de conversión del microcontrolador, sino incorporar los conversores dentro del mismo circuito integrado y que se comuniquen mediante algún enlace digital al procesador, de esta manera disminuir el ruido producido por la transmisión de la señal análoga ya que entre más cerca se realice la conversión menor será el ruido.

- I2C: Se busca instalar sensores, memorias, u otros dispositivos que utilicen este medio de comunicación, por esto es necesario que el procesador lo tenga integrado. Lo favorable es que I2C permite múltiples esclavos, con esto sólo se necesita un módulo I2C.

- SPI: Al igual que el módulo I2C, el modulo SPI permite la interacción con otros dispositivo o periféricos, pero la gran diferencia es que por cada dispositivo extra se debe disponer de un GPIO que haga de línea CS (*chip select*).

- Debug: El procesador debe disponer de una interfaz de debuggeo, la cual facilite el desarrollo de software en la plataforma, ya que de lo contrario se vuelve una tarea tediosa e ineficiente. Además de ser una manera de cargar código directo a la FLASH o RAM para probarlo.

- Facilidad de construcción: Hoy en día, la gran mayoría de los circuitos integrados tienden a ser cada vez más pequeños para el ahorro de espacio, esto es bueno para la fabricación en masa, ya que vuelve más baratos los costos. Cuando se construyen placas en bajas cantidades y de manera manual, algunos de los encapsulados modernos no son factibles de ser soldados a mano. Por esta razón se deben seleccionar sólo integrados que puedan ser manipulados y soldados al PCB a mano. Esto se debe cumplir para los circuitos integrados, y también para las resistencias, condensadores e inductancias ya que, no pueden ser muy pequeñas pues la construcción del PCB, mediante LPKF(empresa Alemana de máquinas para la prototipos), tiene un límite de separación entre pistas dado por el diámetro de la herramienta.

- Alimentación: Debe ser compatible con la alimentación de la HC3, es decir, debe recibir una alimentación desde 6 [V] a 8,4 [V]. Además debe tener otra entrada de alimentación para los

motores, cuyas pistas deben soportar la corriente que circula debido al consumo de todos los motores, sin subir su temperatura a niveles peligrosos. El consumo general de todo el nuevo sistema no debe ser excesivo, ya que por ser instalada en un sistema móvil, debe alimentarse desde las baterías del robot, esto quiere decir ser menor a 1 [A].

Esos son los requisitos principales del sistema que se diseñó, siendo algunos de ellos muy restrictivos. Por ejemplo, los tipos de encapsulados que son factibles de soldar a mano son al mismo tiempo los más grandes impidiendo obtener un tamaño muy reducido o simplemente algunos modelos de controladores no tienen estos encapsulados (son considerados viejos) dejando como no factible algunos modelos con muy buenas características.

3.3. Selección del Controlador

La selección del controlador o de una familia de controladores es un punto muy importante en la fabricación de cualquier plataforma de procesamiento, esto principalmente porque entre diferentes modelos no hay una compatibilidad directa. Luego de haber revisado las distintas marcas de microcontroladores, diferentes arquitecturas y modelos se eligió el TMS320F28335 de Texas Instruments. [11]

El TMS320F28335 (28335), catalogado como DSC (*Digital Signal Controller*) une las características de un controlador, con una capacidad superior de procesamiento. Es parte de la familia de microcontroladores C28x pero se diferencia de ellos por ser el primero de rama con FPU, con lo cual obtuvo el título del microcontrolador líder de procesamiento de punto flotante del año 2008, con esto logra la unión de procesamiento de punto flotante con la diversidad de periféricos integrados de un microcontrolador, cosa que no se logra con un DSP, ya que no disponen de todos los periféricos, en especial de los ADC. Además la ventaja de ser controlador es que está especializado en aplicaciones de alta exigencia de latencia, por esta razón tiene un excelente sistema de interrupciones y DMA para reducir el uso del procesador en atender a los periféricos.

Las principales razones por las que se eligió el TMS320F28335 como controlador fueron las siguientes:

- FPU: Capacidad de procesar números de punto flotante y no sólo de punto fijo, ya que el *firmware* necesita de mucho cálculo de punto flotante para el cálculo de la caminata del robot.

Esta FPU es referida como C28x+FPU por que incluye todas las funcionalidades de la CPU de la familia de controladores C28x, más la FPU que da soporte para números de punto flotante IEEE754, obteniendo compatibilidad de código con los anteriores C28x. [9]

- SCI: Dispone de 3 módulos SCI independientes que pueden funcionar en *fullduplex*. Cada uno posee dos FIFO (*buffer* del tipo *First In First Out*) de 16 *bytes* para transmisión y para recepción. La principal ventaja de esto es el significativo ahorro de tiempo de procesador, ya que permite no atender tan seguido la transmisión y recepción por el SCI. [4]

- I2C y SPI: Estos módulos en el 28335, al igual que el caso de SCI disponen de dos FIFO para la recepción y la transmisión, aunque en este caso, por como es la comunicación I2C, no se puede transmitir simultáneamente, pero aun así es un ahorro significativo en el uso de procesador. Para el caso del SPI sí es un ahorro de procesador, ya que permite una continua transmisión de información mediante el SPI si es conocida la cantidad de datos a enviar. [6, 7]

- McBSP: Este puerto no estaba en las características que se buscaban originalmente, pero surge como una alternativa de alta velocidad para algunos tipos de periféricos al igual que como una buena interfaz de comunicación con algún otro procesador. Dispone el TMS320F28335 de 2 módulos de este tipo. Este puede ser dejado como posible conexión de expansión para el uso de otra plataforma similar en conjunto. [10]

- Multiplexión de Salidas: Un gran punto a favor es el hecho de disponer de un multiplexor para las salidas de los módulos y las distintas salidas GPIO, lo que permite configurar de distintas maneras por cual pin se saca los distintos modulos. Debido a la gran cantidad de módulos, no se disponen de pines de salidas suficientes para poder sacarlos todos, pero como existen módulos que no se utilizan simultáneamente, se pueden elegir cuales usar y por donde sacarlos según sea más cómodo para el diseño del PCB. [11]

- ADC: Dispone de 16 canales con una conversión de 12 *bits* con un tiempo de conversión de 80 [ns]. Una de las características interesantes de éste, es la posibilidad de secuenciar el orden de las entradas de la conversión para tener mediciones más continuas. En conjunto con el DMA permite la captura de datos de manera independiente del procesador. [11]

- JTAG: Compatible con JTAG permitiendo el *debuggeo* a tiempo real o análisis de la memoria en ejecución.

- Encapsulado: Disponible en formato LQFP176 (*Low-profile Quad Flat Package*) , el cual tiene pines hacia los costados ubicados cada 0.5 [mm], de manera que es factible de ser soldado a mano sin problemas, a diferencia del popular BGA (*Ball Grid Array*), y utilizarse en un prototipo de sólo dos capas. El largo de cada lado es de aproximadamente 26 [mm] formando un cuadrado. Siendo factible para la construcción de una placa no muy grande, dentro de los requerimientos planteados. [11]

- Voltajes y consumo: Requiere de dos voltajes independientes para funcionar, los cuales son 1.9 [V] para el núcleo y 3.3 [V] para los periféricos en general. Hay que diferenciar que otros DSP funcionan con 1.8 [V] lo que lo hace distinto a los típicos DSP con respecto a los requerimientos de energía. Con respecto al consumo, en el peor caso especificado en la documentación no supera 500 [mA], lo cual dependiendo del rendimiento es factible. [11]

- XINTF: Es la interfaz externa para memorias, permite la conexión de memorias RAM y FLASH. Tiene un ancho de 32 *bits* de datos y de 20 *bits* de direcciones. [11]

3.4. Conversores RS232 y RS485

Fue necesario convertir de SCI a RS485 para la comunicación con los motores, y a RS232 para la comunicación con el sistema de alto nivel. Para esto se debió seleccionar circuitos integrados que realizaran esta conversión, con encapsulados apropiados y que no tuvieran funciones que nos se fuesen a ocupar y de esta manera se reduce el espacio físico.

El *transceiver* seleccionado para RS485 fue el MAX3078EASA de MAXIM, el cual provee velocidades de transmisión hasta 16 [Mbps] gracias a que el *slew rate* no viene limitado como ocurre en otros dispositivos de su misma familia. El encapsulado de este integrado es un SO-8. Por como es el protocolo de comunicación de los motores, sólo se necesita que sea *halfduplex* por esta razón se necesita un selector entre transmisión y recepción. No requiere componentes externos salvo un condensador de *decoupiling* de 0.1 [μF]. Se utilizaron 3 *transceiver*, uno para cada SCI. [15]

Para el RS232 se seleccionó el MAX3227ECDBR de MAXIM que trae un conversor, transmisión y recepción, en un encapsulado SSOP-16. Este sí requiere de componentes externos los cuales son 5 condensadores de 0.1 [μF], uno corresponde al de *decoupiling* y los otros 4 son del *charge pump*. Las características especiales de este *transceiver* le permiten velocidades de transmisión de hasta

1 [Mbps] y posee un modo de *autoshut down* que le permite ahorrar energía mientras no esté en uso. [8]

3.5. Conversor USB

Para incorporar de USB a esta plataforma, que no lo incluye como módulo interno el 28335, se eligió usar un circuito integrado que emula un puerto serial sobre el USB. Esto hizo que la implementación fuese transparente para los programas y para el *firmware*, lo único que cambiaba es el medio físico por el que se transporta. [2]

El circuito integrado elegido fue el FT232RL de FTDI Chip en un encapsulado SSOP-28 y que soporta USB 2.0 *fullspeed*. Como el USB provee a los periféricos de fuente de alimentación este integrado posee la capacidad de alimentarse directamente desde USB, esto permite que sólo este activo el conversor cuando se está conectado al BUS y en caso que no lo esté simplemente no se active, ya que no tiene razón de hacerlo. Otra ventaja de hacerlo de esta manera es el poder resetear la placa si está conectado al sistema de alto nivel sin que este tenga que volver a reconocer al dispositivo USB. Además provee de todas las líneas necesarias para un puerto serial completo pero sólo se utilizaron las líneas de RX y TX, junto con otras salidas especiales que sirven para monitorear actividad, por eso se instalaron leds para saber cuando está funcionando.

Para su correcto funcionamiento se requirieron componentes externos. Una inductancia, para el filtrado de la energía proveniente del USB, un condensador para lo mismo. Además un condensador, para el regulador interno de 3.3 [V]. Los condensadores de *decoupiling* recomendados son dos, 4.7 y 0.1 [μF].

Como conector se eligió el mini-USB tipo B, el cual es de mucho menor tamaño que el de cliente USB normal y es de montaje superficial. Este es el típico conector que tienen los reproductores de audio portátiles.

3.6. Puerto de expansión

Puede ser que en algún futuro se hubiese deseado conectar esta plataforma a otra que fuese capaz de realizar mayor cantidad de cálculos, o simplemente para tener una manera más eficiente de comunicación con otras placas similares se dejó un módulo McBSP para su comunicación (6

GND	GND	GND	MCLKRA(GPIO7)	MDRA(GPIO21)	MFSXA(GPIO23)	SCI-TXD
VIN	VIN	VIN	MFSRA(GPIO5)	MDXA(GPIO20)	MCLKXA(GPIO22)	SCI-RXD

Figura 3.6.1: Descripción del puerto de expansión.

EXT_TXD	EXT_RXD
USB_RXD	USB_TXD
28_TXD	28_RXD
28_TXD	28_RXD
232_RXD	232_TXD
EXT_TXD	EXT_RXD

Figura 3.7.1: Descripción de selección de SCI

líneas), así como se recibe un SCI de manera de poder conectarlo a los módulos de RS232 o USB así como al SCI-A del 28335. Además se agregaron líneas con tierra y alimentación no regulada para la otra placa de manera que fuese un único punto de conexión, esto llevó a un conector de 7x2 *pines*. En la figura 3.6.1 se muestra la distribución de líneas del puerto de expansión.

3.7. Selector entre USB y RS232

Internamente para el 28335 es lo mismo el MAX3227 o el FT232, por esta razón ellos van conectados mismos *pines* del SCI-A. Debido a la multiplexión, sólo una de la varias salidas que tiene el SCI-A funciona mientras se está en modo *bootloader*-SCI, a estos pines van conectados los conversores RS-232 y USB. Otro de los pares de *pines* que se obtienen de la multiplexión del SCI-A se conectó uno de los MAX3078. Por esta razón el SCI-A pudo tener dos funciones, comunicación serial a un sistema de alto nivel o a un bus RS485, pero se debe elegir entre el USB o el RS232 mediante un selector.

Como selector se diseñó un arreglo de *jumpers* y *pinheaders* para seleccionar cuál se usa, de manera que cambiando *jumpers* se pueden seleccionar entre RS232 y USB del SCI-A de esta placa o del SCI proveniente del puerto de expansión, permitiendo el uso simultáneo de ambos conversores. En figura 3.7.1 se muestra como es el arreglo las entradas y salidas.

3.8. Selector del modo de booteo

El 28335 dispone de diversos modos de *booteo*, es decir, lo que hace al momento de partir el dispositivo. Internamente dispone de un *bootloader* que es un programa que le permite comunicarse

Modo	SW0	SW1	SW2	SW3	Descripción
F	0	0	0	0	Salto a *Flash*
E	0	0	0	1	SCI-A *bootloader*
D	0	0	1	0	SPI-A *bootloader*
C	0	0	1	1	I2C-A *bootloader*
B	0	1	0	0	eCAN-A *bootloader*
A	0	1	0	1	McBSP-A *bootloader*
9	0	1	1	0	Salto a XINTF x16
8	0	1	1	1	Salto a XINTF x32
7	1	0	0	0	Salto a OTP
6	1	0	0	1	GPIO Paralelo *bootloader*
5	1	0	1	0	XINTF Paralelo *bootloader*
4	1	0	1	1	Salto a SARAM
3	1	1	0	0	Checkeo de *bootmode*
2	1	1	0	1	Salto a *Flash* sin calibración ADC
1	1	1	1	0	Salto a SARAM sin calibración ADC
0	1	1	1	1	SCI-A *bootloader* sin calibración ADC

Tabla 3.8.1: Configuraciones de los *switches* de *booteo*

mediante algún periférico y programar en la *Flash* el programa enviado. Los diferentes modos le permiten partir desde la *Flash* la ejecución del programa o desde algún periférico, y son controlados por 4 *bits*, siendo 16 los diferentes modos disponibles. Para poder seleccionarlos se utilizó un *dip switch* superficial de 4 interruptores, permitiendo así la selección de los 16 modos. Además debe ir con una resistencia de *pull up*, y como estas mismas líneas son usadas por el XINTF, deben tener deben tener una resistencia a tierra en caso de querer dar un cero lógico. Por esta razón la lógica queda invertida en el *switch*. En el cuadro 3.8.1 se muestran los diferentes modos y las configuraciones del *dip switch*.

Los principales modos que se utilizaron fueron el Modo F, E y 4. El modo F es el más utilizado, ya que corresponde a la operación normal de carga desde la *Flash* para la ejecución del *firmware*. El modo E se utiliza si se quiere cargar un nuevo *firmware* mediante el SCI. El modo 4 corresponde al *booteo* desde la RAM, lo que permite que en conjunto con el JTAG se corra el código desde la RAM sin necesitar de grabarlo a la *Flash* para realizar pruebas y *debuggeo*, siendo mucho más eficiente. [11]

3.9. Oscilador y PLL

Como opción de oscilador se utilizó un cristal, porque de esta manera se utiliza menos espacio que utilizando un oscilador externo, ya que los osciladores externos pequeños que había disponibles no eran posibles soldarlos a mano. Gracias al PLL interno que posee el 28335, se pueden lograr múltiples configuraciones de frecuencias que son válidas hasta un máximo de 150 [MHz], la cual se logra con un cristal de 30 [MHz], ya que se obtiene un multiplicador de 5x. Debido a que la documentación del 28335 al momento de realizar la compra del DSC presentaba errores, no se compro el cristal de 30 [MHz] sino uno de 20 [MHz] con lo que la frecuencia máxima alcanzable es de 140[MHz]. Con esta frecuencia se realizaron todas las pruebas en el prototipo. Para operar con cualquier cristal y el oscilador interno, sólo se requirieron unos condensadores externos descritos en el *datasheet* del 28335.

El módulo de PLL está compuesto por un multiplicador programable y por un divisor programable. Las opciones del multiplicador van de 1x a 10x y el divisor tiene opciones de /1, /2 o /4. Éste es el *clock* que alimenta a los módulos de los periféricos, por esta razón cualquier cambio en el PLL afecta también a las configuraciones de los periféricos. [11]

3.10. Alimentación de Energía

Cuando ya se seleccionaron los circuitos integrados que son parte de esta placa, se pudo elegir el regulador de voltaje para la placa completa. El consumo total no supera los 500 [mA]. Sobre los voltajes que se necesitaron fueron de 1.9 [V] para el *core* del 28335 y de 3.3 [V] para el resto del 28335, los conversores RS-232 y RS485. Ya que se necesitaron de estos voltajes, se eligió un regulador superficial dual, TPS767D301PWP, el que tiene un regulador fijo de 3.3 [V] de hasta 1 [A] y un regulador variable con el que se puede obtener los 1.9 [V] y también de 1 [A]. El encapsulado de este integrado es TSSOP28 PowerPAD, el que está diseñado para disipar calor mediante el un PAD de tierra que tiene por la superficie interior, lo que permite un mejor intercambio de calor. Los componentes externos necesarios para este regulador son básicamente condensadores como filtros, tanto a la entrada como a la salida, y resistencias para el regulador variable. [5]

Capítulo 4

Implementación

4.1. Construcción del Prototipo

Habiendo elegido el 28335 como núcleo del sistema de control, los conversores necesarios y otros componentes, se comenzó la construcción del sistema. Se definió que el prototipo debía ser diseñado en 2 capas (*Layers*) pues fue construido en la LPKF, por esta razón no se podía cumplir los requerimientos necesarios de espacio ya que la LPKF pone limitaciones al tamaño de las vías así como al de las pistas.

La construcción de esta placa conllevó varias etapas: dibujo de esquemáticos, ruteo, preparación para construcción, construcción del PCB y verificación del PCB. Una vez concluidas estas etapas que correspondieron a la fabricación de un PCB, se debió soldar los componentes para luego realizar pruebas de cada módulo que corroboraron su correcto funcionamiento. En la figura 4.1.1 se muestra un esquema de la placa que se implementó.

4.1.1. Programas Utilizados

Como se compone de varias etapas la construcción de un PCB, varios programas computacionales fueron utilizados por la función que cumple cada uno. En cada una de las etapas se busca obtener un tipo de archivo diferente al anterior. A continuación se explica que programas fueron usados y qué función cumplen:

- OrCAD Capture 15.7: Capture fue utilizado para crear el *netlist*, el cual es la descripción

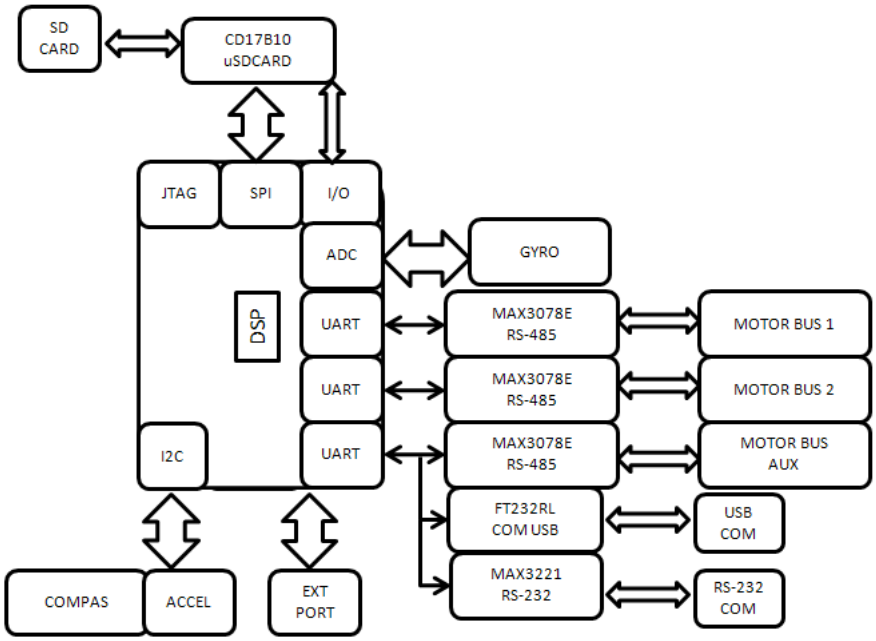


Figura 4.1.1: Esquema de la placa de control

de la conectividad entre diferentes componentes electrónicos. Esto lo hace a partir de un esquemático en el que se van agregando los diferentes componentes y realizando conexiones de manera grafica.

- OrCAD Layout 15.7: Una vez ya creado el *netlist*, se debió asociarle la forma física a cada componente y la superficie sobre la cual es soldada, y con la información del *netlist* se realizó la unión física del los componentes. Layout permite realizar diseños en múltiples capas, además de revisar restricciones de diseño. Entre las herramientas que caracterizan a este programa está la de autorouter, que realiza en forma automática la generación de pistas de acuerdo al *netlist*. En esta ocasión no se utilizó el *autorouter*, sino que se realizó de forma manual el routeo. El objetivo de esta etapa es la generación de archivos Gerber para la construcción de la placa.

- CircuitCAM 5.0: Por como es el principio de acción de la LPKF, se debe remover el material alrededor de una pista. Con la información del Gerber se trazan los recorridos para las fresas que deben remover el cobre para lograr los anchos de las pistas, así como seleccionar las herramientas para realizar los orificios de la placa. También se definen los bordes físicos de

la placa y la herramienta de corte. El objetivo de esta etapa es tener un archivo que contiene las instrucciones para la LPKF para la fabricación del PCB separadas por capas de proceso.

- BoardMaster 5.0: Con el archivo generado con el CircuitCAM, el BoardMaster da las instrucciones a la máquina para la fabricación de la placa y monitorea mientras se realizan los cortes y perforaciones.

4.1.2. Esquemáticos

La primera etapa de la generación de los esquemáticos fue la generación de los diferentes componentes que se utilizaron. La librería disponible en Capture es muy amplia, pero algunos componentes son más nuevos y/o muy específicos por lo que no están incluidos. Un gran punto a favor es que el fabricante del 28335, Texas Instruments, suele poner a disposición pública modelos de sus componentes en diferentes formatos, para el diseño de esquemáticos. Para los otros componentes que no estaban en la librería hubo que hacer el componente en Capture, pero la herramienta que ofrece para realizarlo es simple y fácil de utilizar.

Para el dibujo de los esquemáticos para los diferentes bloques del sistema, la mayoría de los componentes ofrecen en sus *datasheet* sus configuraciones de uso más comunes y especifican formulas para el cálculo de los componentes a utilizar. Basadas en ellas se construyó el sistema como se deseaba, habiendo verificado el cumplir con los requerimientos para cada componente.

Uno de los atributos que posee cada componente en Capture es el PCB *footprint* donde se debía poner el nombre del encapsulado, que más tarde utilizó Layout para elegir de su librería el *footprint* adecuado. Para los componentes como resistencias y condensadores se eligió un tamaño pequeño pero montable manualmente: 0805 que tiene un tamaño de 80 [mil] x 50 [mil] que típicamente tiene una separación aproximada de 1 [mm] entre sus PADS, el cual es apropiado para la LPKF.

Una vez ya dibujados todos los esquemáticos para el sistema completo y agregado a cada componente su atributo de PCB *footprint*, se generó el *netlist* (se debe especificar con que programa se continua el proceso). En este caso se exportó el netlist para ser usado con Layout con la opción ECO, opción que permite que se reflejan los cambios realizados en Capture en Layout inmediatamente.

4.1.3. Ruteos

Cuando se comenzó a trabajar con Layout a partir del *netlist* creado con Capture, se apreció que Layout no disponía de todos los *footprints* necesarios, ya que algunos de estos eran muy específicos (LPQF176, HTSSOP28 y conectores). Por este motivo lo primero que se realizó fue crear estos footprint, basados en la descripción expuesta en el *datasheet* correspondiente, esta descripción en algunos casos es la recomendación para el *footprint* en sí y en otros casos es la descripción del encapsulado que hay que adaptarla para creer un *footprint* adecuado.

Lo siguiente realizado fue el *place*, que corresponde a posicionar los componentes en la placa. Se debió definir un tamaño de placa inicial para ir colocando los componentes dentro de los límites.

Típicamente la capa superior es conocida como capa de componentes y la inferior como capa de soldado, pero esto no implica que en la capa inferior no se puedan situar componentes. Se decidió en primera instancia que componentes iban por arriba y que componentes por abajo. La posición de los componentes se fue modificando a medida que se trazaban las pistas.

Para el trazado de las pistas se debió tomar en cuenta las separaciones entre pistas, ancho y largo de estas. En este caso, la separación debía ser tal que las herramientas de la LPKF fuesen capaces de cortar, así como al soldar quedasen bien aisladas unas de otras. Dado que el componente que tenía menor separación entre sus pines era el 28335, aproximadamente de 10 [mil] y la menor herramienta de la LPKF es de 8 [mil] era factible esa separación mínima. Con respecto al ancho mínimo se decidió usar de 12 [mil] y el ancho de cada pista se iba decidiendo dependiendo de la función que cumplían, en caso de las alimentaciones. Como éstas tienen que estar distribuidas a lo largo de toda la placa, se van ramificando desde pistas anchas hasta pistas delgadas para llegar a cada componente, de esta manera no se produjeron pistas ni muy largas ni muy angostas. Ya que se realizó en dos capas, se debía pasar una pista de una capa a otra con orificios, los cuales son llamados VIAS. El número de estas no debía ser muy elevado en lo posible. Los pasos anteriores fueron iterados, modificando la posición de los componentes, corriendo y moviendo pistas para reducir el tamaño final de la placa. Así mismo se debió definir el borde, lo que va a fijar el tamaño de la placa.

En las figuras 4.1.2 y 4.1.3 se muestran el resultado del ruteo de la placa. De esta etapa se obtuvieron los archivos *GerberX* (descripción de las pistas) y el *Aperture List* (descripción de los hoyos) con la que se pasó a la siguiente etapa de fabricación.

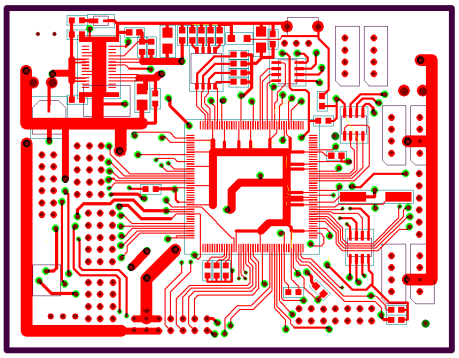


Figura 4.1.2: TOP *Layer* del Prototipo

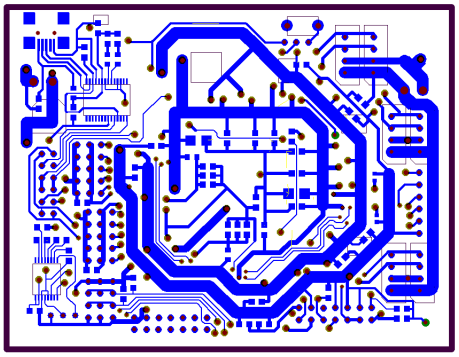


Figura 4.1.3: BOTTOM *Layer* del Prototipo

4.1.4. LPKF y Construcción

Para la confección de la placa en la LPKF, el archivo Gerber no se podía utilizar directamente, por esta razón se debió importar cada capa en un proyecto de CircuitCAM. Aquí se importaron en las capas superior, inferior, borde y aperturas.

En las capas superior e inferior se debió procesar para realizar la aislación de las pistas, con esto se definió la trayectoria de las fresas de manera de remover el cobre que se encuentra alrededor de las pistas, y con esto formar la pista de cobre. Se seleccionaron para este trabajo dos herramientas, una fina de 8 [mil] y una más gruesa de 1 [mm], la aislación seleccionada es de 1 [mm] de esta manera la aislación se realizó con ambas herramientas, donde el desgaste principal se realizó con la herramienta de 1 [mm] y las partes donde no pudo entrar esa herramienta se realizaron con la de 8 [mil], incluyendo el borde de cada pista. Los resultados de esta etapa se muestra en las figuras 4.1.4 y 4.1.5.

El siguiente paso fue desarrollar la capa de corte exterior, para ello se seleccionó de los *Gerbers* el

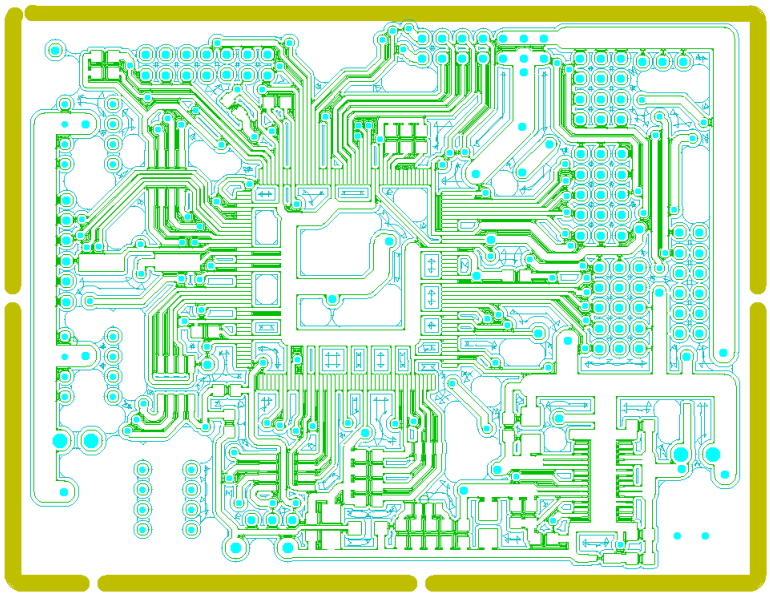


Figura 4.1.4: Instrucciones de corte para TOP layer

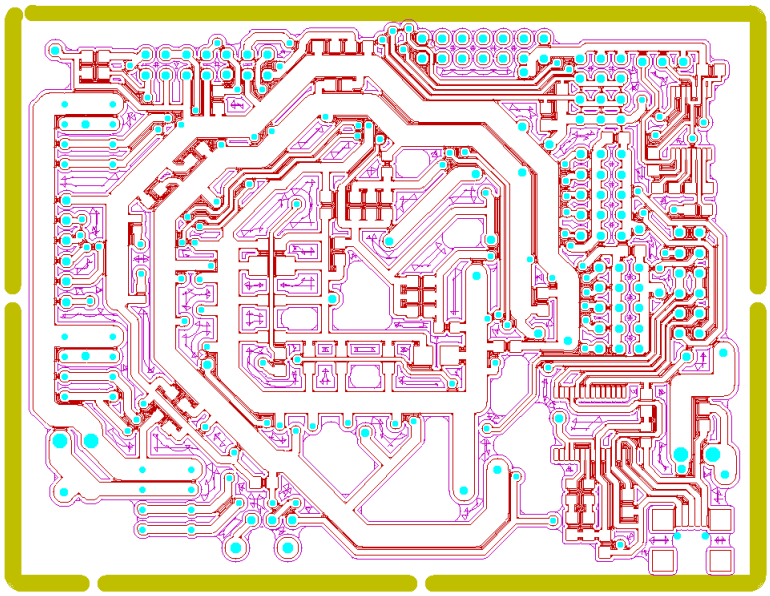


Figura 4.1.5: Instrucciones de corte para BOTTOM layer

límite exterior de la placa. Se le procesó como capa de corte, habiendo seleccionando una herramienta para el corte del borde, y catalogándolo como corte exterior. La herramienta seleccionada para esta etapa fue el *countour router* de 2 [mm]. Este corte se recomienda dejarlo abierto en 1 o más puntos, de manera que al cortar con la fresa, la placa no se separe y pueda dañarse o dañar a la maquina. Para esto el programa permite insertar espacios en el corte.

El listado de agujeros se debía importar de manera similar a como fue hecho con las pistas, sin embargo este archivo contenía información del diámetro de las perforaciones, pero no disponía del listado de herramientas disponibles en la LPKF. Por esta razón se debió seleccionar manualmente la herramienta que se utilizó para cada diámetro de perforación.

Una precaución que se debió tomar al importar cada capa de trabajo fue confirmar que la escala fuese la misma para en todos los archivos, ya que cualquier diferencia hacía la placa defectuosa. Por suerte, cualquier error en la escala es perceptible, ya que provoca diferencias de órdenes de magnitud en los tamaños. Otro punto importante fue que la interpretación decimal del archivo *gerber* fuese la misma con la que se crearon los archivo.

Finalizada las etapas en el CircuitCAM se obtuvo un archivo que contiene las capas ahora separadas por las operaciones de cada herramienta. En BoardMaster se ejecutaron estas instrucciones y de esta manera se fabricó la placa.

Durante el proceso de fabricación en la LPKF, se fueron ejecutando las instrucciones que iban cortando la superficie de la placa de cobre y formando placa. El orden recomendado para la fabricación de una placa es primero realizar las perforaciones, a continuación los cortes sobre el cobre para la formación de las pistas y finalmente el corte del borde exterior. La razón para este orden es que al realizar las perforaciones después de haber cortado las pistas, los *pads* quedan muy frágiles logrando que se despeguen de la superficie o se rompan al ser perforados. Con respecto al corte exterior, se debe realizar al finalizar todo, pues si se realizan las pistas una vez que ya se ha cortado el borde, la placa queda suelta y no permite sacar correctamente todo el material de cobre, incluso pudiendo provocar la ruptura de pistas.

En el primer intento de fabricación en la LPKF se obtuvo una placa fallida debido a que se utilizó una herramienta muy desgastada, lo que provocó que las pistas no quedasen totalmente aisladas y algunas pistas simplemente destruidas. Esto se remedió realizándola nuevamente pero con una herramienta en buen estado. De todas formas hay que notar que se apreció la diferencia

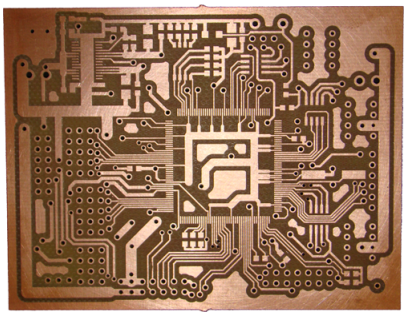


Figura 4.1.6: TOP *Layer* del prototipo

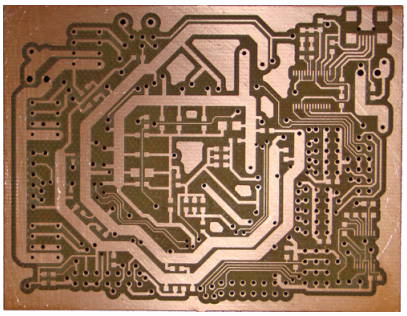


Figura 4.1.7: BOTTOM *Layer* del prototipo

entre los primeros cortes y los últimos pero esto es debido a la cantidad de material que debe fresar. Por esto, fue bueno utilizar dos herramientas, una gruesa para el desgaste grueso y una fina para los detalles. En las figuras 4.1.6 y 4.1.7 se muestran la placa resultante de la LPKF antes de soldarse.

4.1.5. Revisión y soldado

La placa recién salida de la LPKF no estaba lista para ser soldada, se le tuvo que remover los restos de cobre que quedaron sobre ella para evitar que la soldadura se adhiriera a estos restos formando posibles puentes, y se tuvo que eliminar la posible grasa que tenía el cobre debido a la manipulación. Para eliminar ambas cosas se le pasó una virutilla metálica, pero esto no removió todos los restos, por lo que con lupa y un bisturí se eliminaron los restos presentes entre las pistas.

A continuación, se insertaron las VIAS para unir las pistas de un lado de la placa al otro. En este caso se utilizó un sistema que viene con la LPKF, que son unos remaches de cobre que al ser puestos dentro del orificio y ser remachado, unen amas caras de la placa. El remachador tiene un ajuste de la presión utilizada para poner el remache, este se debe ajustar para que la VIA quede bien puesta, ni muy apretado (porque rompe las pistas) ni muy suelto (no hace bien contacto).

Lo siguiente fue el soldado de los componentes a la placa, para ello se utilizó soldadura con plomo y estaño, y como ayuda para el soldado se utilizó *flux*, este permite que la soldadura fluya más fácil, de esta manera fue posible soldar los pines de los encapsulados superficiales que de otra manera se formaban grumos de soldadura. Como regla general se recomienda soldar las cosas más pequeñas y bajas primero (resistencias, condensadores), luego circuitos integrados superficiales para luego terminar con las cosas altas, como son los conectores. Esto es porque las cosas más altas molestan para soldar las más bajas y en el caso de los *through hole* al dar vuelta para soldarse se caen si hay algo soldado de mayor altura. Hay que asegurarse de que se pongan en la dirección correcta los componentes y que queden bien alineados antes de soldar. La cantidad de soldadura no debe ser excesiva ya que puede provocar puentes accidentales, pero debe ser suficiente para que la resistencia de contacto sea baja.

4.1.6. Pruebas del prototipo

Para evitar daños a los componentes por conexiones incorrectas al soldar, se decidió soldar por bloques, y como el más peligroso en caso de estar mal era la alimentación de voltaje, se soldó el regulador y sus componentes, una vez soldados, se procedió a energizarlo y medir en las pistas si se tiene el voltaje correcto. Como medida de precaución, se alimento con 5 [V] pero con corriente limitada, para que en caso de haber algún corto circuito, evitar dañar las pistas y el regulado. Con el regulador de voltaje ya comprobado, el correcto funcionamiento y voltaje adecuado, se procedió a soldar el resto de los componentes.

Para probar los conversores de RS232 y de USB, en los *jumpers* de selección se ubicó el jumper entre las líneas de RX y TX de cada conversor, de esta manera se obtuvo el efecto de *echo* y con ello se corroboró el correcto de ambas líneas.

Una vez asegurada la comunicación serial, se empezaron las pruebas del DSC, para ello se utilizó los ejemplo que tiene Code Composer para el 28335, todos ellos ejecutados en RAM y cargados mediante el JTAG. Con las pruebas de la ejecución de los ejemplos se comprobó que el DSC estaba funcionando adecuadamente.

4.2. Implementación del *firmware*

El *firmware* es el programa que ejecuta el 28335, que controla las funciones del robot, permitiéndole caminar, recibir comandos, hacer mediciones de los sensores y otras más. Este programa debe ser muy estable y en lo posible no tener errores que podrían acarrear alguna mala orden al robot, provocando algún daño. El tiempo de latencia entre el momento en que las ordenes sean dadas y se efectúen debe ser mínimo para asegurar un comportamiento fluido y sin retrasos.

Para escribir el código y compilar para el 28335 se utilizó el Code Composer Studio 3.3 que es un conjunto de programas que incluyen el compilador, *linker* y herramientas de trabajo, además de permitir la utilización directamente del JTAG en su entorno de trabajo. Se utilizó una versión gratuita de prueba que tiene todas las funcionalidades por un tiempo limitado.

El JTAG que se utilizó es XDS510USB de Spectrum Digital, el cual tiene conexión USB con el PC, y sin necesidad de conexión de energía externa. El conector a la placa es el de 14-pin JTAG header. Otro programa no incluido en el CCStudio que se utilizó para la gradación del firmware en la *Flash* es el SDFlash, que es una utilidad de Spectrum Digital para ser utilizado con el JTAG o el *bootloader* serial.

En las siguientes subsecciones se explica su estructura y su funcionamiento, así como consideraciones especiales realizadas.

4.2.1. Estructura del *firmware*

La estructura del *firmware* se divide principalmente en 3 partes, las cuales son: inicialización, *loop* principal e interrupciones. En la figura 4.2.1 se muestra un esquema del flujo de la ejecución del *firmware*, en esta aparecen claramente la etapa de inicialización y la del *loop* principal. Las interrupciones aparece en la figura 4.2.1 pero por ser funciones que se ejecutan asíncronamente, no están dentro del esquema del flujo del programa.

4.2.1.1. Inicialización

El bloque de inicialización tiene tres partes fundamentales, la primera es la inicialización de los periféricos. Éstos, al comenzar la ejecución del código, se inician desactivados y con sus configuraciones por defecto, por esta razón se deben activar de acuerdo al uso que se les dé. Una característica muy importante es que dejando desactivados los módulos que no se usan, se reduce

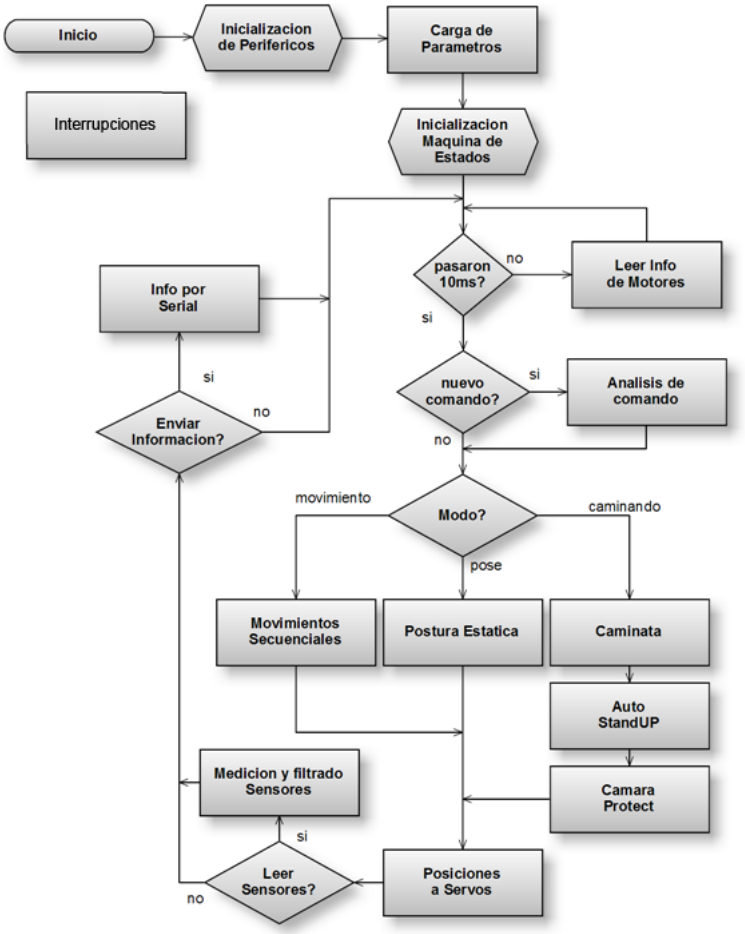


Figura 4.2.1: Esquema de flujo del *firmware*

considerablemente el consumo de corriente.

La segunda parte de la inicialización es la lectura de los parámetros del programa desde la memoria externa. Acá se almacenan las opciones para la ejecución del *firmware*, así como los parámetros de la caminata. Estos valores no pueden ir grabados en el *firmware* ya que si se quisiese modificar alguno habría que grabar el *firmware* nuevamente, lo cual no es muy recomendable y no permitiría un rápido ajuste de los parámetros.

La tercera parte es la inicialización de las variables de estado para el correcto funcionamiento del *loop* principal. En esta sección es donde se inicializan las interrupciones, ya que es inmediatamente antes de la ejecución del *loop* principal y tienen que ser habilitadas después de las demás inicializaciones, para no corromper variables de estado.

4.2.1.2. *Loop* principal

El *loop* principal es lo que se ejecuta una y otra vez para generar las funciones del robot. Tiene un periodo de 10 [ms], lo que implica que 100 veces por segundo se calculan las posiciones de los servo-motors y se les envían. Además se realizan las lecturas de de sensores dependiendo los parámetros de ejecución.

La primera parte del *loop* principal es la que regula si se ha cumplido el tiempo de 10 [ms] de cada ciclo. Esto se hace evaluando el estado de una variable que es modificada cada 10 [ms] por la función de interrupción activada por el módulo de *timers*. En caso de que no se haya cumplido el tiempo de un ciclo se hace una lectura de alguna información no crítica de los servo-motores, con esto se evita desperdiciar el tiempo de procesamiento, sin embargo no puede ser una información crítica, ya que el tiempo disponible no es suficiente para la lectura de todos los servo-motores regularmente. Estos datos pueden ser el voltaje del bus, temperatura de los motores, etc.

Lo primero que se hace al entrar al *loop* es revisar si durante el ciclo ha llegado algún comando válido, el cual es guardado en un *buffer* para luego ser analizado. En caso de ser válido, se interpreta su contenido y se cambian las variables de estado necesarias. Cada carácter del comando es recibido y guardado por la función de interrupción del SCI-A. Cuando se completa un mensaje se levanta un *flag* para avisar que hay un nuevo comando. Esta manera de analizar los comandos limita la frecuencia a la que se puede recibir comandos, ya que solo se pueden interpretar comandos cada 10 [ms], pero evita entrar en estados inestables debidos a la recepción de múltiples comandos durante un ciclo o modificación de variables en medio de la ejecución del loop principal.

La siguiente etapa del ciclo principal depende del modo en que se esté operando, hay 3 modos de operación los cuales son: movimientos secuenciales, postura estática y caminata.

En la postura estática el robot tiene un vector con las posiciones para todos los motores a la que deben llegar en un tiempo determinado, para esto se interpolan las posiciones de los motores desde la posición actual hasta la final en el tiempo requerido. Luego permanece en esa posición hasta que un nuevo comando sea interpretado. Este modo solo es usado en la confección de movimientos secuenciales y no tiene ninguna utilidad durante el funcionamiento normal.

En los movimientos secuenciales, el robot ejecuta un movimiento que está almacenado en la memoria, el cual está compuesto por un arreglo de posturas y una duración. A diferencia del modo de posturas estáticas, una vez que completa una postura continúa con la siguiente, formando una

secuencia de movimientos. Una vez que se termina la última, se vuelve al modo de caminata. Estos movimientos son los que hacen posible que el robot se levante y que patee la pelota.

El modo caminata es el encargado de generar los movimientos para desplazar el robot. En este modo se calculan las posiciones que deben tener los motores para poder crear el movimiento de piernas que hagan desplazarse al robot basado en los parámetros entregados en el comando de caminata. Este bloque es el que más consumo de cálculo tiene el *firmware*, y para todas estas pruebas se utilizó el mismo bloque de caminata que el *firmware* del HR18. La máquina de estados de este modo, es el encargado de controlar la inicialización de los motores, que corresponde a enviar los valores de ganancias y otros parámetros a los motores, así como de activar o desactivar el torque de ellos. A la salida de esta etapa se encuentran dos módulos los cuales son: *Auto StandUP* y *CamaraProtect*. Ambos módulos deben estar justo a la salida de la caminata ya que modifican su salida. *Auto StandUP* es el módulo que con la información del acelerómetro determina si el robot esta caído y en caso de ser así, elige que movimiento ejecutará en el siguiente bloque para levantar el robot. *CamaraProtect* también utiliza la información del acelerómetro pero sin filtrar, levantando la cabeza si el valor de la aceleración sube sobre cierto umbral para tratar de disminuir el daño durante una caída.

Estos tres modos anteriores tienen como salida la actualización del vector de posiciones de todas las articulaciones, que será enviado a los motores en la siguiente etapa, donde se genera un paquete que se transmite por el RS485 para que los motores cambien su referencia de posición.

La siguiente parte es la lectura de sensores. Dado que puede ser que no se quiera una frecuencia muy elevada de lectura de sensores, dependiendo de los parámetros del *firmware* se efectúa o no la lectura de los sensores. Aquí también se realiza el procesamiento de lo recibido de los sensores, como es el caso del acelerómetro para determinar si el robot esta caído o simplemente recibió un impacto. Para esto se implementó un filtrado de media móvil para eliminar los cambios bruscos.

$$Xf_t = \alpha X_t + (1 - \alpha) Xf_{t-1} \tag{4.2.1}$$

Donde α con rango de [0,1] corresponde al peso con que se actualiza la inclinación filtrada Xf_t con la lectura de la inclinación actual X_t y el valor anterior Xf_{t-1}.

La siguiente etapa del *loop* principal es la de envío de Información. En esta etapa se ve si

corresponde enviar alguna información por el serial, donde esta información puede ser desde el valor de una variable, como parte del servidor de Parámetros (que será explicado luego en la sección 4.2.4), envío del status del robot o alguna otra información. Estas transmisiones son asíncronas con respecto a los comandos y se configura su frecuencia de envío mediante parámetros que son cargados desde la memoria al iniciarse la ejecución. En la sección 4.2.3 se explica la estructura de estos mensajes. Finalmente se vuelve al comienzo del *loop* principal y se repite todo nuevamente.

Esta versión del *firmware* implementa las interrupciones de *timer*, I2C y SCI. La interrupción del *timer* controla la activación del *loop* principal para que se demore 10 [ms] por iteración. La interrupción del SCI es la encargada de recibir cada carácter e ir guardándolos para su posterior análisis y la del I2C que se encarga de pedir los datos al acelerómetro. Para mejorar el rendimiento de estas funciones de interrupción que deben tardar lo menos posible en atenderse, han sido copiadas a RAM al comienza la ejecución del *firmware* y de esta manera ahorrarse el tiempo de carga desde *Flash*.

4.2.2. Módulo SD

El módulo de tarjetas SD utilizado es producido por SparkFun el que utiliza el integrado DOSonCHIP CD17B10 de la compañía Wearable. Este módulo permite la lectura y escritura de memorias SD con formato FAT16/32 mediante una interfaz serial RS232 o una interfaz SPI. En este caso se utilizó mediante SPI ya que no se disponía de más puertos RS232. En la actual versión del *firmware* de este módulo no se utiliza el puerto SPI como un puerto SPI puro ya que requiere de líneas extras para funcionar. Para utilizar estas líneas se conectaron a los GPIO disponibles. [20]

El principal motivo para la utilización de memorias SD es la posibilidad de usar memorias removibles y modificar los parámetros directamente en la memoria y no conectar la placa de control al computador para modificar algún valor.

4.2.3. Comandos Seriales y RS232-USB

La comunicación serial con el 28335 es la manera que tiene para comunicarse con el exterior, y esta puede ser a través del conversor RS232 o a través del conversor USB serial. Internamente para el 28335 es indiferente cual de los medios esté utilizando, ya que para éste sólo es comunicación mediante el SCI-A.

Para generar un protocolo de comunicación simple de decodificar para el 28335 y para una persona, se diseñó uno sencillo de entender y de escribir. Básicamente se define el carácter " [" como carácter de comienzo de mensaje y el carácter "] " como fin de mensaje. La principal ventaja de este tipo de mensajes es la facilidad de reconocer el comienzo y final del mensaje.

El comando se definido como el primer carácter después de " [", dependiendo de este carácter se determina la cantidad de parámetros que necesita el mensaje. En caso de que el número de parámetros entregados no concuerde con los esperados, el comando se desecha. Dentro del mensaje, los valores van separados con " | " y dependiendo del comando estos pueden ser de diferentes tipos: *float*, *long*, *char*, *interger* y *unsigned interger*. Estos valores son transmitidos en representación de *strings* permitiendo compatibilidad entre procesadores que tengan distinta representación interna. Es verdad que esto no es lo óptimo, pero soluciona problemas de compatibilidad.

Para las respuestas del 28335 al sistema de alto nivel, se utiliza un sistema similar al anterior, salvo que el carácter de inicio de mensaje es " (" y el de fin de mensaje es ") ". Estos son principalmente utilizados por el servidor de parámetros que se explica en la sección 4.2.4.

Que el protocolo de comandos este definido de esta manera permite que una persona usando un terminal serial sea capaz de generar los comandos manualmente y de esta misma manera es simple entender las respuestas que se le entregan.

El SCI-A puede ser configurado hasta 1 [Mbps] y es soportado por ambos conversores a esa velocidad de transferencia. A esa velocidad por cada mensaje que se quisiese mandar en cada ciclo del *loop* principal, se pueden enviar 1000 caracteres. Usando esta misma estimación se deduce que incluso a tasas de transferencia de 50 [kbps] se pueden enviar 50 caracteres por ciclo. Dado que una orden de caminata no supera los 20 caracteres, queda claro que este estilo de comandos no es algo crítico que pudiese afectar el desempeño. Los comandos más largos pueden llegar a los 100 caracteres pero sólo se usan cuando se está diseñando movimientos.

4.2.4. Servidor de Parámetros

El servidor de parámetros permite modificar y observar variables de estado y parámetros del funcionamiento del *firmware* desde el exterior mediante la interfaz serial. Estos mismos parámetros son los que son iniciados desde el módulo de memoria SD al comienzo del programa.

Internamente, el programa tiene un arreglo de punteros genéricos que se inicializan con las direc-

ciones de las variables de interés. Además se dispone de un arreglo del mismo largo que contiene el tipo de variable que es. El índice en estos arreglos identifica a la variable en cuestión. Externamente desde un PC, usando este índice se puede consultar el valor de esta variable, o escribirle un valor nuevo.

El escribirle un valor nuevo a un parámetro permite probar cambios de manera muy rápida y eficiente incluso sin tener que reiniciar el *firmware*, así como permite observar el estado actual de algunas variables, lecturas de sensores y toda variable de interés que se haya agregado previamente a este arreglo.

Como internamente se almacena que tipo de variables es, el comando para modificar la variable es el mismo para todos sin importar que tipo de variable sea. Y así mismo cuando se informa un valor al exterior, no se aprecia que tipo de variable es internamente, ya se manda en su representación de *string*.

4.2.5. Acelerómetro Digital I2C

El acelerómetro digital utilizado es modelo el LIS3LV02DQ de la empresa ST tiene 3 ejes (X,Y,Z) que pueden medir hasta ±6g. Tiene comunicación digital mediante conector I2C y SPI. En este caso se seleccionó para ser utilizado mediante I2C. [19]

Básicamente el funcionamiento de este acelerómetro se basa en la escritura y lectura de registros en un mapa de memoria, este es el esquema típico en los dispositivos I2C. Una vez que parte la ejecución del *firmware*, en la etapa de inicializaciones se escribe en un registro para iniciar el acelerómetro, y para la lectura de los valores se leen los registros donde están almacenadas las mediciones.

Como la medición es mayor a 8 *bits* y se quiere leer además los 3 ejes del sensor, en vez de preguntar por cada eje cada vez se puede hacer una lectura en bloque de la memoria así ahorrando tiempo del protocolo I2C.

4.2.6. Giróscopo

El giróscopo que se seleccionó es uno análogo de dos ejes, el modelo es IDG-300 de InvenSense. Por ser análogo se requiere utilizar el módulo ADC para su conversión la cual se efectúa una vez en cada ciclo. [12]

4.2.7. Brújula

La brújula que se seleccionó es la HMC6352 de Honeywell de dos ejes y por lo que es posible saber la proyección del campo magnético terrestre proyectado sobre el plano formado por ambos ejes. La comunicación de ésta es I2C, lo cual es una ventaja ya que se pueden montar varios dispositivos al mismo tiempo al mismo módulo I2C. [3]

Como las mediciones son del campo magnético proyectado en un plano, lo ideal es tener el sensor de manera horizontal, y de esta manera la proyección es máxima. Como el robot puede estar en varias orientaciones debido a caídas, las mediciones solo son validas al estar de pie en torno a la vertical.

El funcionamiento del sensor es similar al del acelerómetro, ya que se debe iniciar y luego se deben leer los valores desde un registro.

4.2.8. Motores Dynamixel

Para comunicarse con los motores conectados al RS485, aun cuando se disponía de los tres conversores (SCI-A, SCI-B y SCI-C), para la realización de pruebas solo se utilizó el SCI-B donde se conectaron todos los motores.

Los motores se configuraron a 1 [Mbps] y los mensajes deben ser enviados como lo define el protocolo de los motores. A los motores se les configuró de manera que sólo responden cuando se les pregunte algo y no cada vez que les llegue un comando, esto es principalmente por que el tiempo de respuesta y el tiempo de estabilización del cambio del conversor a modo de lectura puede hacer más ineficiente la transmisión de los comandos y si algún motor responde provocaría colisión de mensajes.

Para la lectura de valores desde los motores, se debió tomar en cuenta el tiempo de cambio del conversor a modo de lectura, asegurándose de que el mensaje ya se haya transmitido completamente antes.

Una vez transmitido el comando, se debe cambiar el conversor a modo de lectura y esperar por la respuesta de parte del motor. Ya que esta respuesta puede no llegar, en caso de falla del motor o ausencia de este, se debe fijar un timeout para no dejar pegada la ejecución del código.

Cuando el mismo comando se debe transmitir los motores como en el caso de los comando de encendido o apagado de torque, se utilizó la transmisión de comandos *broadcast* ya que son más

rápidos que mandar un comando distinto por cada motor. Esto logra transmitiendo un mensaje a un ID especial.

4.2.9. Grabación del programa en Flash

Para la grabación del *firmware* a la *Flash* hay que tomar en cuenta tres cosas. Cuando se define el proyecto en el CCStudio, hay que definir si se va estar ejecutando desde la *Flash* o RAM, esto se hace definiendo la estructura de memoria que se desea ocupar en el archivo del *linker*. Esto es de mucha importancia ya que cuando se hicieron pruebas, ejecutando todo el código desde la RAM, como el proyecto es muy grande, no cabe entero. Por esta razón, en caso de querer probar alguna parte del código en especial es recomendable tener todo el código escrito de manera muy modular y deshabilitar las partes que no sean fundamentales para probar el código de interés. En caso de querer ver todo el código funcionando sólo se puede ejecutar desde la *Flash*.

Durante la grabación en la *Flash* se debe tener en consideración que es un proceso crítico, que si hay cualquier interrupción puede ser que se dañe irremediablemente el 28335. Por esto es recomendable el uso de una fuente de voltaje externo o una batería que esté en buen estado, y hacerlo en un lugar donde no haya ninguna interrupción accidental.

El programa SDflash básicamente lo que hace para grabar código a la memoria *Flash* es ejecutar un proyecto que debe ser compilado para las características de la placa, al que se en envía el programa compilado y lo graba. Se modificaron dos proyectos, uno para el uso con el JTAG y el otro para el SCI-A.

Capítulo 5

Discusión de resultados

5.1. Pruebas e Instalación en UCH1 - Tanker

Las pruebas realizadas fueron de dos tipos principalmente: las pruebas específicas de cada módulo del 28335 y las pruebas de las funciones del *firmware*. Las pruebas de cada módulo consistieron en configurarlos para las tareas que se necesitaban y de esta manera validar su correcto funcionamiento. Las pruebas de las funciones del *firmware* permitieron comprobar que realmente se comportaban tal cual para lo que fueron diseñadas.

5.1.1. FPU vs NO FPU

Se realizaron pruebas de cálculo con la FPU y sin la FPU utilizando las librerías de cálculo en el proyecto de CCStudio, estas pruebas solo se realizaron por no disponer de la librería de la FPU en un principio. Por esto, una vez que se puso a disposición pública, se utilizó esta librería lo que mejoró considerablemente el desempeño del código probado. La librería que se utilizó en un comienzo corresponde a la librería genérica para los controladores de la familia 28xx, que no incluyen soporte para FPU.

5.1.2. Motores Dynamixel

La prueba consistió en lograr comunicación con los motores mediante el RS485, para ello se debía configurar el SCI para estar al mismo *baudrate* que los motores, en este caso se utilizó 1

[Mbps] para las pruebas. El SCI tiene otros parámetros que se debieron configurar para adaptarse a la comunicación de los motores. Para comprobar la comunicación con los motores se implementaron funciones que mandar mensajes a los motores, y para escuchar respuesta de ellos. Para esto se debió configurar un GPIO que es el encargado de cambiar entre transmisión y recepción del conversor RS485, el timing entre transmisión y recepción también debió ser ajustado para asegurar la transmisión y recepción completa del mensaje.

La primera prueba fue generar un mensaje de *broadcast* a los motores que encendieran el *led* de status que posee el motor. Una vez conseguido esto, se empezaron a generar otros comandos y a recibir las respuestas de los motores. Se observó que no siempre se obtiene respuesta de los motores, aunque la tasa de error era muy baja. Esto ocurría cuando se le enviaban órdenes muy seguidas a los motores, y aumentaba a medida que se le reducía el tiempo entre órdenes. Por esta razón la programación de la recepción de respuestas desde los motores se implementó robusta para manejar el caso en que no se obtuviera respuesta y no dejar bloqueado el código.

La prueba final del manejo de los motores fue la de corroborar si era posible de enviarle posiciones a cada uno de los motores cada 10 [ms], tarea que se consiguió con éxito sin problemas.

5.1.3. Módulo tarjeta SD

Este módulo se comunica mediante SPI con el 28335, pero además requiere de otras líneas de comunicación extra que fueron manejadas mediantes GPIO. Las líneas de *reset*, *dirección* y *busy* son las líneas de control para manejar el funcionamiento del módulo y el SPI sólo se utiliza para la transferencia de datos. Este modulo SD, pese a incluir SPI, este protocolo no esta bien implementado en la actual version del *firmware* del CD17B10 y la documentación que se entrega de él es muy deficiente, ya que toda la información se hace referente a comunicación serial y nada al sobre el SPI.

La elección de este módulo fue una mala elección, ya que las buenas características que presenta sólo se cumplen para la comunicación serial y no para la SPI, información que en la documentación no se daba. El módulo suele fallar en la lectura, teniendo que repetir la lectura de los datos. Este fallo es aleatorio, por lo que en la versión final de la placa fue reemplazado por una memoria I2C.

5.1.4. Acelerómetro I2C

Las pruebas al acelerómetro comenzaron por configurar correctamente el módulo I2C del 28335. Luego de entender el funcionamiento del protocolo, se generó un mensaje válido para el dispositivo. Al inicializar el dispositivo hay que realizar una escritura en el registro de *enable*, y para obtener la información del sensor se realizan una lecturas a registros. La lectura de cada eje del sensor está contenida en dos registros diferentes que se deben leer para luego concatenarlo en un *integer* de 16 *bits*, esto gracias a que se utiliza la representación de *Little Endian* (el sensor tiene opción de *Little Endian* o *Big Endian*). Gracias a que los registros de los 3 ejes están contiguos se puede simplemente leer los 6 registros con un mismo comando al acelerómetro.

Luego de haber realizado pruebas exitosas de lecturas donde se determinaron los umbrales a los que se puede considerar caído el robot, se programaron las funciones de escritura y de lectura mediante las interrupciones I2C del 28335 y así no dejar detenida la ejecución del resto del código mientras se obtienen los valores de este sensor.

5.1.5. Brújula

La prueba de la brújula fue menos compleja, ya que la comunicación por I2C ya se había realizado para el acelerómetro, por lo que solo hubo que cambiar la dirección del dispositivo y la de los registros. Los resultados fueron buenos y se pudo medir con la brújula la dirección del norte estando en orientación horizontal. Cuando se orientaba fuera del plano horizontal, los valores que se obtuvieron no fueron precisos, ya que se produce la proyección del campo magnético sobre el plano del sensor.

5.1.6. ADC

Para la prueba de los conversores análogos se ejecutaron ejemplos de que venían con el CCStudio. En ellos se probó el correcto funcionamiento de algunas entradas análogas mediante un divisor de voltaje hecho potenciómetro. No se probó el giróscopo por que la versión de la caminata que utilizó no se le incluyó la compensación por giróscopo.

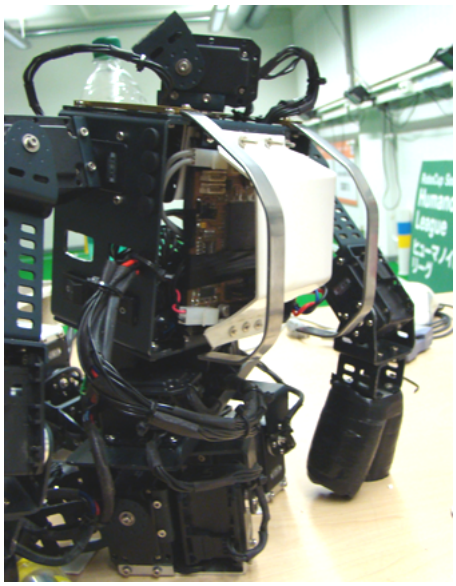


Figura 5.1.1: Prototipo instalado en UCH1

5.1.7. Instalación y pruebas en Tanker

Una vez ya probados los módulos por separado y comprobado su correcto funcionamiento, se unieron al *firmware*, donde se hicieron pruebas del *firmware* completo. Para esto fue necesario conectar el robot UCH1 para realizar las pruebas de funcionamiento del robot.

Ya se había probado el control de varios motores simultáneamente. Una vez conectado a todos los motores del robot hubo que probar si era capaz de realizar movimientos coordinados. Para esto se ajustó la dirección de cada motor ya que dependiendo en cual posición se encuentre el motor, es el sentido de giro en que funciona, esto se representa con un parámetro para cada motor que es el sentido de giro, es un 1 o un -1. Además existe un valor de *offset* del motor, el cual se ajusta para ubicar el motor en posición 0°, definida como la posición de cada articulación cuando el robot está completamente estirado con los brazos formando una cruz. Con esta pose del robot se realizó la prueba para comprobar que la conversión desde ángulo en grados, a formato para el comando del motor funcionaba.

En la figura 5.1.1 se aprecia instalada la placa de control en el robot UCH1. Este robot no posee un lugar apropiado en su interior donde montar la placa, por este motivo tuvo que ser montada en la espalda del robot de manera que se pudiera ubicar más firmemente y acceder más fácil a ella. Cabe notar que esta placa corresponde a un prototipo para probar su funcionamiento y no es la

versión final. La versión final cumple con la restricción de tamaño para el robot HR18.

Ya teniendo la configuración de los *offsets* y las direcciones de los motores se probó el resto de las partes del *firmware*. Para realizar pruebas de las poses estáticas basta con tener un vector con todas las posiciones a los motores y enviárselas. El siguiente paso fue realizar un movimiento de secuencial de poses estáticas. Esto se hizo interpolando las posiciones de los motores de dos poses estáticas, esto genera un movimiento continuo al cual se le controla la velocidad mediante la duración entre las dos posturas. El *firmware* demostró ser capaz de realizar movimientos secuenciales así que se comenzó a diseñar movimientos que pudiesen levantar al robot hasta su postura erguida. Aquí se encontraron los primeros problemas con el robot Tanker, ya que mecánicamente el robot Tanker no era capaz de ponerse de pie ya que tenía poses que no puede realizar con las piernas y unido a que los motores RX-28 que tenía instalados para las rodillas y abdomen eran muy débiles para esas ubicaciones, se logró hacer sólo un movimiento que permitía levantar el robot cuando éste se encontraba caído de frente, el cual exigía a los motores más allá de lo apropiado. Para suplir el hecho que por geometría no le permitía levantarse de espalda, se realizó un movimiento que hacia rodar el robot por el suelo dándolo vuelta y ocupando el movimiento frontal. Pese a las limitaciones explicadas, el sistema de control era capaz de realizar la tarea pedida.

Lo siguiente en probar fue el módulo de caminata. Hay que recordar que el código de la caminata estaba extraído del *firmware* original del HR18 por este motivo hubo que adaptar algunas partes para su utilización. Cuando se probó si el robot era capaz de caminar se acrecentaron más aun las deficiencias mecánicas de Tanker. Éste, al levantar un pie, por tener rodillas débiles y el centro de masa muy alto, se le doblaban las rodillas o se tropezaba con los pies al no poder levantarlos apropiadamente, levantarlo no era el problema sino que al levantar una rodilla la otra bajaba, para dar un paso. Tanker no logró caminar apropiadamente debido a su estructura física pero sí realizaba los movimientos de caminata, esto quería decir que el código de la caminata y el envió de órdenes a los motores era estaba funcionando bien, y los impedimentos mecánicos eran el problema. Estas pruebas debieron haber sido realizadas directamente en un HR18, pero en los momentos en que se estaban haciendo estas pruebas, estos robots estaban siendo usando exhaustivamente como preparación para la RoboCup 2008.

Luego de comprobar el mal desempeño mecánico de Tanker, al robot se le reemplazaron las rodillas y el abdomen por motores RX64. Con estos cambios, los movimientos para levantar el

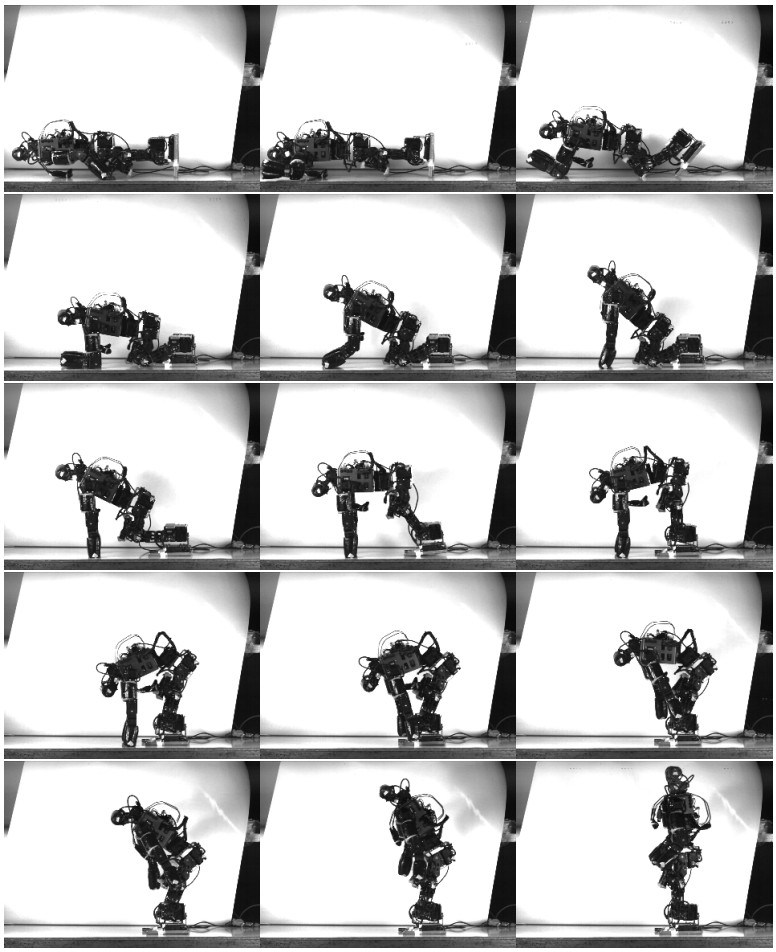


Figura 5.1.2: Secuencia de Tanker levantandose

robot mejoraron mucho porque los motores no se les exige al máximo. En la figura 5.1.2 se muestra la secuencia de Tanker levantándose. Pese a estas mejoras la caminata no mejoró complemente, ahora sí pudo realizar algunos pasos, pero igual se caía. La razón de esto puede ser la falta de sintonización de parámetros de la caminata al robot Tanker ya que esta caminata fue diseñada para el robot HR18.

Con el robot ya funcionando, pero no caminando, se probó la función de *Auto StandUP*, la cual funcionó excelentemente. Esto permite derribar el robot y que este detectara su caída y se levantara automáticamente. En los HR18, era el sistema de alto nivel el encargado de detectar que el robot estaba caído y seleccionar el movimiento para levantarlo. Esto provocaba errores ya que trataba de levantarlo varias veces aun cuando ya se hubiese levantado ya. La velocidad para detectar la caída, es ajustable mediante el parámetro del filtro del acelerómetro llamado α en 4.2.1.2.

La otra función que se probó fue la de *CamaraProtect*. Aun cuando la estructura de protecciones de Tanker protege la cámara de manera que nunca toca el suelo en un impacto, se probó que es posible alcanzar a levantar la cabeza antes de un posible impacto en el suelo pues se tiene una velocidad de reacción muy alta, dado por la velocidad de adquisición de los sensores y la velocidad de respuesta de los motores. Esto puede ser una mejora a los robots HR18, en los cuales su cabeza sí impacta con el suelo al caer.

Para la prueba del servidor de parámetros, se implementó una interfaz capaz de modificar estos valores y de escucharlos externamente en un PC. Esto permitió modificar parámetros durante las pruebas sin tener que cambiar la información existente en la tarjeta SD. El desempeño del servidor fue excelente y resulta ser una herramienta indispensable en futuras implementaciones del *firmware*.

La implementación del comando de movimiento de cabeza, a diferencia de como esta implementado en HR18, que recibe cada ángulo por separado, recibe los dos ángulo que lo definen simultáneamente. Además para mejorar el desempeño, la orden a la posición no emplea interpolaciones, ya que cuando se utiliza para realizar *tracking* no se usa la imagen hasta que se está en la posición solicitada, por lo que no se le restringió la velocidad. Las pruebas que se realizaron fueron de tracking a la pelota usando el sistema de alto nivel que reconocía la pelota capturando imágenes con la cámara instalada en la cabeza y enviado las ordenes directamente a la placa mediante RS232. El desempeño del *tracking* fue mejor que el obtenido en los HR18 ya que puede recibir más instrucciones por segundo bajo las mismas condiciones.

Del desempeño obtenido de las pruebas se puede, determinar varios cambios que deben tenerse en cuenta para la versión final:

- El rendimiento y funcionalidad el TMS320F28335 quedaron demostrados con las pruebas del *firmware*, y aun tiene capacidad de procesamiento que no se está utilizando,y ademas se puede aumenta de 140 [MHz] a 150 [MHz], lo que permitiría implementar alguna otra funcionalidad al código. Los módulos de periféricos que dispone lo hacen una excelente opción como controlador. La cantidad de corriente consumida en pleno consumo nunca superó los 200 [mA] cumpliendo con los requerimientos de energía y el ancho de las pistas de alimentación de potencia no se apreciaron afectadas por la corriente. Quedó demostrado que realizar desarrollo con productos Texas Instruments es una muy buena decisión, ya que ofrecen un muy buen soporte de sus productos, pese a que el 28335 estaba aun en etapa experimental al momento

de empezar a trabajar con éste. La documentación contenía errores que fueron reportados y modificados a la siguiente revisión. Ya una vez que se tenía listo prototipo, 28335 ya estaba en producción activa y la documentación estaba actualizada.

- El tamaño de la placa debía ser más pequeña, ya que el tamaño del prototipo pese a no ser muy grande, no pudo entrar en el interior de Tanker o de HR18. Para esto se deben limitar los periféricos que se utilizarán para así reducir el espacio. Además una buena alternativa es la de separarla en dos placas, al igual que la original del HR18 y así poder disponer de espacio y facilidad de conexión. Los módulos que se mantuvieron son los 2 SCI para ser utilizados en RS485, un SCI para el RS232 y USB, no se mantuvo el puerto de expansión es decir se descartó el McBSP y la entrada de SCI externa. Se dispuso los ADC con un divisor de voltaje para poder ser utilizado con los sensores del HR18 que son análogos en 5[V], puertos GPIO, puerto SPI y puerto I2C con múltiples conectores para utilizar varios dispositivos. No se olvidaron los tornillos de montaje que tienen las mismas posiciones que las placas actuales del HR18 para que sea montable fácilmente.

- Después de observado el pésimo desempeño del módulo de tarjeta SD, se decidió reemplazarlo con una memoria I2C de 1 [Mbit]. Esto también está motivado por el buen desempeño que se obtuvo el servidor de parámetros ya que gracias a esto sería muy sencillo modificar los valores de estos parámetros.

- Debido a que al momento de probar códigos se observó que en caso de ejecutar todo desde la RAM interna. Se decidió agregar un *chip* externo de SRAM modelo CY62167DV30LL-55ZXI de la empresa Cypress Semiconductor Corp, ésta es una SRAM de 16 [Mbit] de ancho 16 *bits* y en encapsulado TSOP48, que esta va conectada directamente al módulo XINT del 28335. [1]

- Los *jumpers* para la selección entre RS232 y USB fue reemplazado por un *dip switch* superficial, el que permitirá seleccionar en menor espacio que conversor se usa. Así mismo el conector mini-B USB fue reemplazado con un conector de *pinheader* ya que esto ayuda a reducir espacio.

5.2. Versión Final

Tomando en consideración los resultados obtenidos de las pruebas del prototipo se rediseño la placa para las nuevas características seleccionadas, ésta fue realizada con el *software* de CadSoft Eagle 4.11. La razón de porqué se cambio de *software,* es para poder realizar una comparación entre programas realizando proyectos similares y además por la facilidad de uso que da este ultimo, se decidio que para proyectos no muy grandes es mucho más rápido el trabajar con Eagle pero para manejar proyectos grandes OrcCad funciona mejor. Nuevamente se tuvo que realizar el trabajo de crear la mayoría de las librerías , y se redibujaron los esquemáticos como se deseaba que quedasen con los cambios nuevos. Como se quería reducir el tamaño, se decidió que esta placa sería fabricada con cuatro capas, en vez de dos, esto implica que no se podrá producir en la LPKF, sino que se deberá mandar a hacer a una empresa externa. De estas cuatro capas, son dos las dedicadas completamente para el transporte de alimentación, y la otras dos son utilizadas para las pistas de datos. En la sección de Anexos se presentan los *layouts* y esquemáticos de la versión final.

El tamaño final de las placas es de 79 [mm] x 50 [mm]. Ambas tienen las mismas medidas e incluyen los orificios para el montaje que corresponden a los mismos que dispone el robot HR18. El tamaño del primer prototipo es de 90 [mm] x 70 [mm], con esto se tiene que la placa nueva es un 62 % del área del prototipo, pero ahora son dos. La ventaja de esto es que están separadas para poder ser montadas. Además se agregaron mas conectores para motores por en cada bus, ya que en el prototipo se demostró que eran muy pocos. Originalmente eran sólo dos conectores por buses, ahora se agregaron 5 por bus, esto es lo que provoca que la placa extra sea tan grande pero además permite alejar de la placa de control los conectores de los motores.

La construcción de la versión final de la placa de control está pendiente en su fabricación pero será durante este año (2008).

Capítulo 6

Conclusiones

Se logró desarrollar una placa de control para robots móviles, en este caso probada especialmente en un robot bípedo. Posee una capacidad suficiente para hacer los cálculos de cinemática directa e inversa y además hacer la adquisición de los sensores y otras tareas como el cálculo de trayectorias de una caminata. El estar usando un controlador de ultima generación que incluye características de los DSP y de controladores, asegura que se realizó un trabajo que va a poder prestar un buen servicio en las aplicaciones y podra ser usado por un buen tiempo, siendo una elección viable y de costo no tan elevado si se produjese en cantidades (estimado USD150 en enero de 2008).

La utilización de un procesador con FPU para aplicaciones donde se requiere mucho cálculo resultó en una mejora del desempeño considerable, comparado con el mismo código corriendo sin FPU y con la placa HC3, esto mejora al punto que queda tiempo libre para realizar otras cosas en las ventanas de 10 [ms].

El objetivo de construir un prototipo a través de todas las etapas de desarrollo de una placa electrónica de control, fue cumplido exitosamente. La calidad de las pistas al realizar la placa en la LPKF por segunda vez (la primera vez las herramientas fallaron) fue buena, y permitió que se pudieran soldar todos los componentes superficiales sin ningún problema. La fabricación de un prototipo tan complejo en la LPKF provoca un desgaste considerable en las herramientas. Se pudo observar el deterioro de la herramienta desde el comienzo del proceso de fabricación y al terminar éste.

La incapacidad de caminar del robot Tanker se debe a problemas mecánicos presentes en el

robot que son ajenos al trabajo de esta memoria, ya que la capacidad de cómputo del 28335 quedó corroborada en generar las trayectorias aun cuando éstas no hubiesen sido capaces de mover adecuadamente el robot.

Pese a utilizar sensores de bajo costo para el acelerómetro, este cumplió con las necesidades planteadas, con un desempeño excelente. El uso de sensores que se comuniquen digitalmente con el procesador permitió no preocuparse tanto del ruido que puede aparecer en la adquisición en caso de que utilicen los conversores ADC. Esto es muy bueno porque permite la ubicación de los sensores en posiciones necesarias que no siempre están cerca del procesamiento. De las interfaces digitales utilizadas, la que se encontró con mejor desempeño es la I2C ya que permite transferencias veloces, y permite múltiples dispositivos conectados al mismo bus.

El módulo SD seleccionado resultó ser tan deficiente y poco robusto que no cumplió con las expectativas mínimas y por esto necesitó ser reemplazado en la versión final. La deficiencia de este sistema fue compensado con el servidor de parámetros, el cual tuvo un desempeño muy bueno permitiendo sacar estadística de funcionamiento, como una estimación del tiempo libre que le quedaba al procesador que de otra manera no se podría haber hecho y que gracias a esto se puede utilizar este tiempo para realizar lecturas del estado de los motores.

Aun cuando la comunicación serial no es la más eficiente para la comunicación con la plataforma de control, sí es la manera más simple de implementar. Queda pendiente para otros trabajos, el encontrar un conjunto de alto nivel y control de bajo nivel que tengan una comunicación eficiente y más veloz. El problema de hacer algo así es que tiene que ser un conjunto de ambas cosas, lo que deja de ser una plataforma general para el control de robots.

El uso de RS485, de la manera en que está ocupada en los motores Dynamixel es muy eficiente y permite pensar nuevas maneras de expansión, ya que se pueden construir dispositivos que utilicen el mismo protocolo que los motores, y sirvan para adquirir información o generar sistemas más grandes gracias a que pueden ser distribuido, más aun, se pueden conectar otros procesadores a esta misma red. Un posible trabajo a futuro es la creación de un sistema modular de procesamiento, que permita unir dispositivos y que además el *firmware* sea un conjunto de librerías que se agreguen dependiendo de la aplicación en la que serán utilizados.

Dentro de este mismo trabajo, la creación de un módulo de caminata propio y la optimización del código mediante la utilización de todas las ventajas que presenta el *hardware* del 28335, son

cosas que quedan pendientes para trabajos futuros.

Bibliografía

[1] Cypress Semiconductor Corp. Datasheet cy62167dv30ll-55zxi 16-mbit (1m x 16) static ram. Technical report, Cypress Semiconductor Corp, 2004.

[2] FTDI. Datasheet ft232r usb uart i.c. Technical report, FTDI, 2005.

[3] Honeywell. Datasheet digital compass solution hmc6352. Technical report, Honeywell, 2006.

[4] Texas Instruments. Tms320x28xx, 28xxx dsp serial communication interface (sci) reference guide. Technical report, Texas Instruments, 2004.

[5] Texas Instruments. Tps767d301-q1 dual-output low-dropout voltage regulators. Technical report, Texas Instruments, 2004.

[6] Texas Instruments. Tms320x28xx, 28xxx inter-integrated circuit (i2c) module. Technical report, Texas Instruments, 2005.

[7] Texas Instruments. Tms320x28xx, 28xxx dsp serial peripheral interface (spi) reference guide. Technical report, Texas Instruments, 2006.

[8] Texas Instruments. Datasheet max3227 3-v to 5.5-v single-channel rs-232 line driver/receiver. Technical report, Texas Instruments, 2007.

[9] Texas Instruments. Tms320c28x floating point unit and instruction set reference guide. Technical report, Texas Instruments, 2007.

[10] Texas Instruments. Tms320f2833x multichannel buffered serial port (mcbsp) reference guide. Technical report, Texas Instruments, 2007.

[11] Texas Instruments. Tms320f28335 data manual. Technical report, Texas Instruments, 2008.

[12] InvenSense. Datasheet integrated dual-axis gyro idg-300. Technical report, InvenSense, 2006.

[13] Isao Parra Javier Testart Rodrigo Asenjo Pablo Hevia Carolina Vélez Felipe Larraín Javier Ruiz-del Solar, Paul Vallejos. Uchile roadrunners 2008 team description paper. Technical report, Universidad de Chile, Department of Electrical Engineering, Chile, 2008.

[14] Robert Kratz Sebastian Petters Hajime Sakamoto Maximilian Stelzer Dirk Thomas Oskar von Stryk Martin Friedmann, Jutta Kiener. Team description paper: Darmstadt dribblers & hajime team (kidsize) and darmstadt dribblers (teensize). Technical report, Technische Universitšat Darmstadt, Department of Computer Science, Germany, 2006.

[15] MAXIM. Datasheet rs-485/rs-422 transceivers. Technical report, MAXIM, 2003.

[16] ROBOTIS. Datasheet dynamixel dx-117. Technical report, ROBOTIS, 2006.

[17] ROBOTIS. Datasheet dynamixel rx-28. Technical report, ROBOTIS, 2007.

[18] ROBOTIS. Datasheet dynamixel rx-64. Technical report, ROBOTIS, 2007.

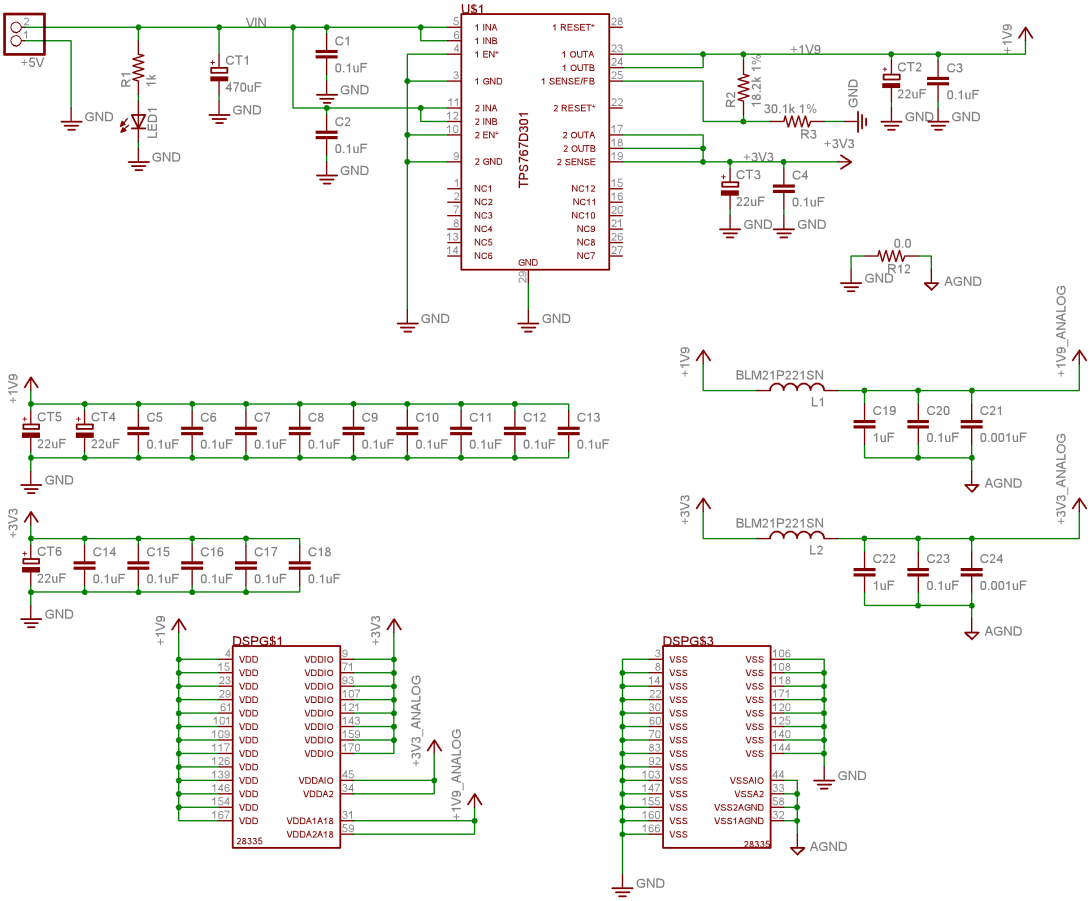
[19] ST. Datasheet lis3lv02dq mems inertial sensor 3-axis - 2g/6g digital output low voltage linear accelerometer. Technical report, ST, 2005.

[20] Wearable. Datasheet dosonchip cd17bxx memory card interface with file system. Technical report, Wearable, 2008.

ANEXOS

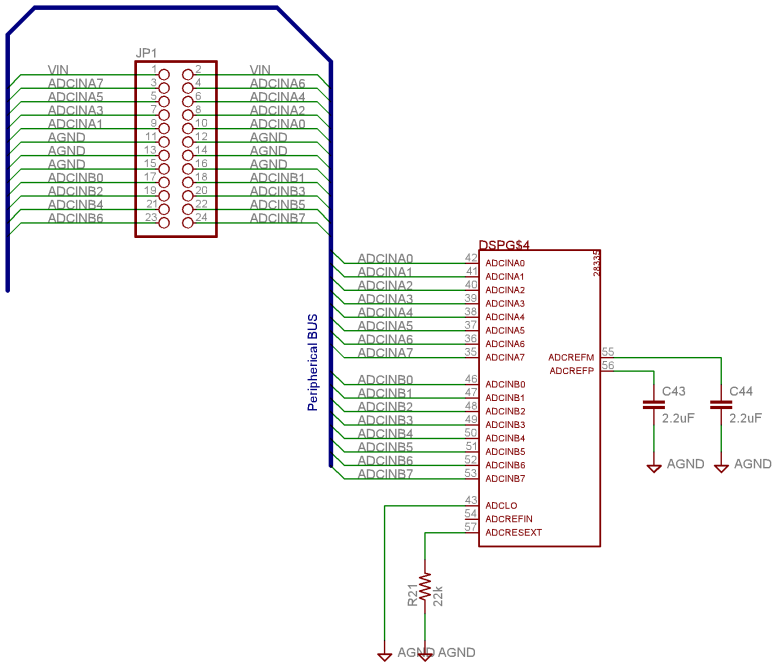
A. Esquemáticos Version Final IC1.1

Esquemático alimentación



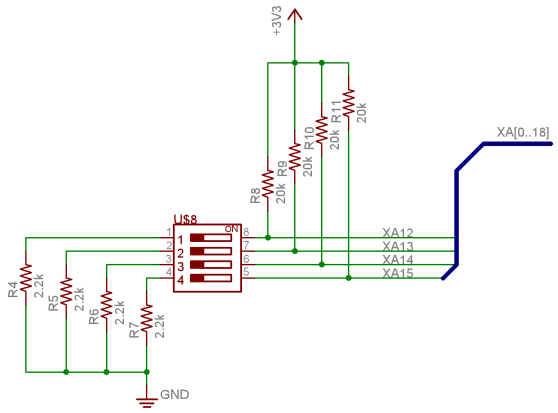
Esquemático de alimentación versión final

Esquemático análogos



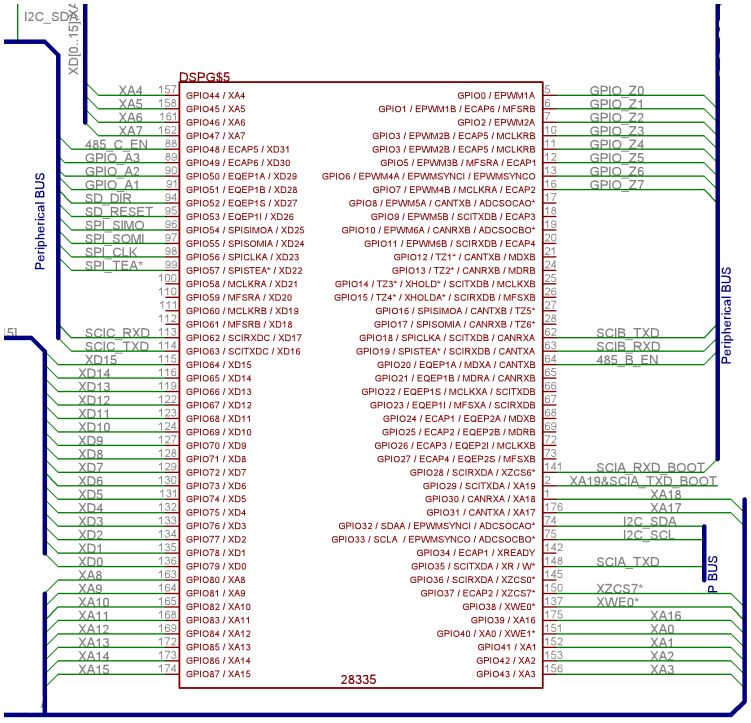
Esquemático de análogos versión final

Esquemático *booteo*



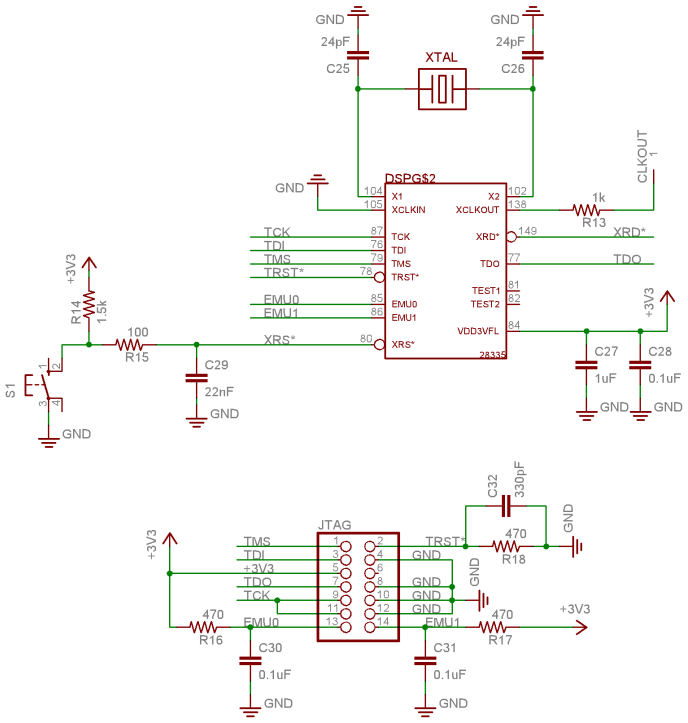
Esquemático de la selección de *booteo* versión final

Esquemático GPIOS



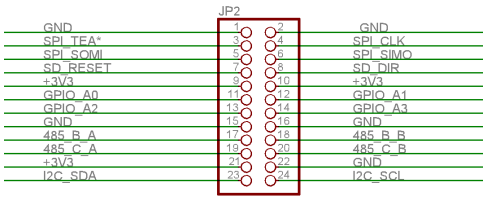
Esquemático de los buses IO versión final

Esquemático JTAG



Esquemático del JTAG versión final

Esquemático conector a placa de extensión



Esquemático del conector a la placa de extensión versión final

Esquemático RAM

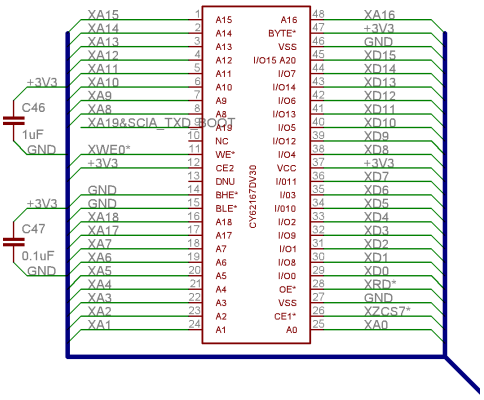
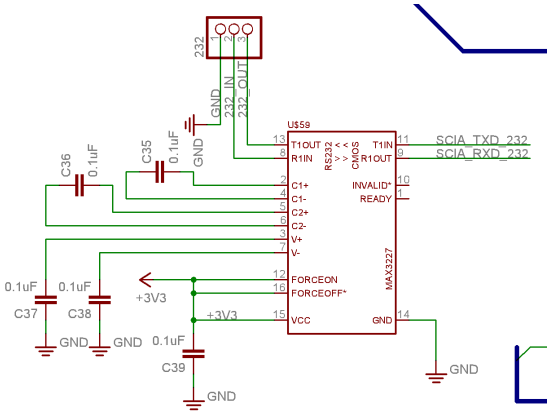


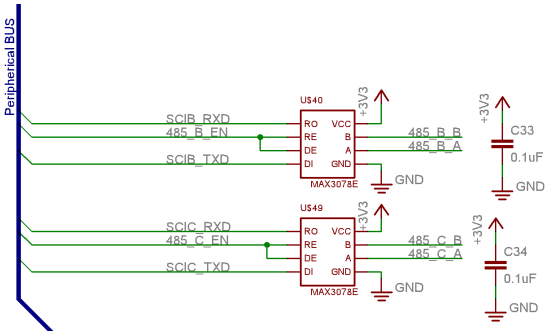
Figura 6.0.1: Esquemático de la RAM versión final

Esquemático RS232



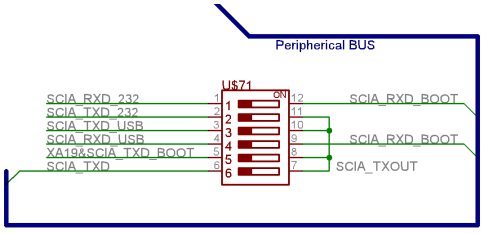
Esquemático del RS232 versión final

Esquemático RS485



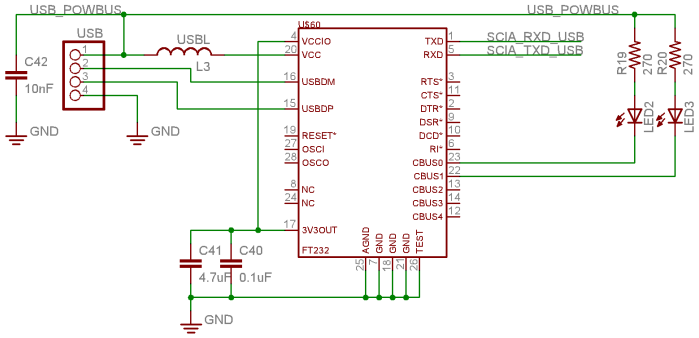
Esquemático del RS485 versión final

Esquemático selector RS232 y USB



Esquemático del selector RS232 y USB versión final

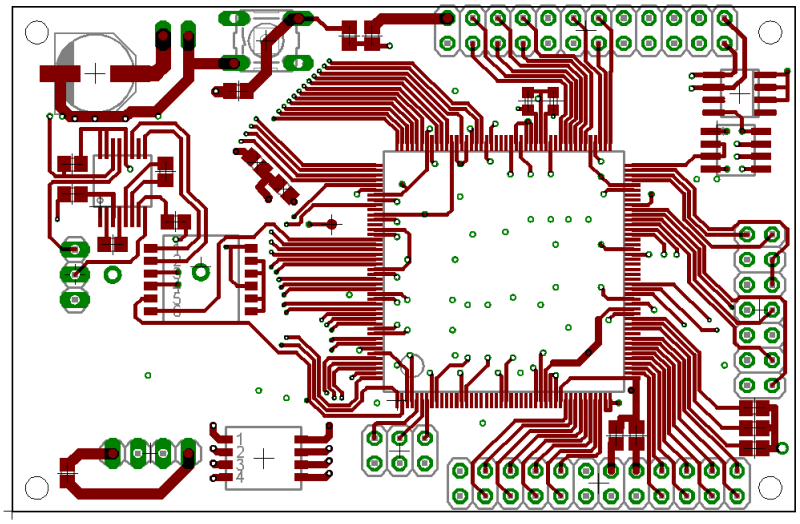
Esquemático USB



Esquemático USB versión final

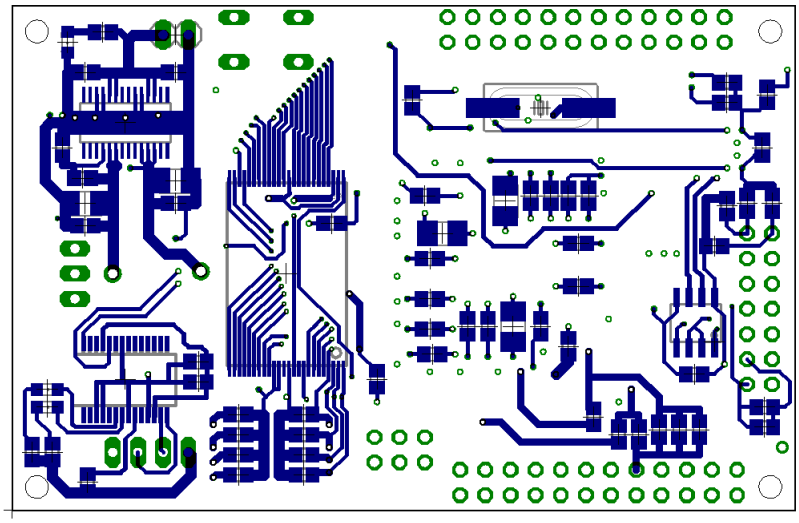
B. Layers del Board Final

TOP Layer



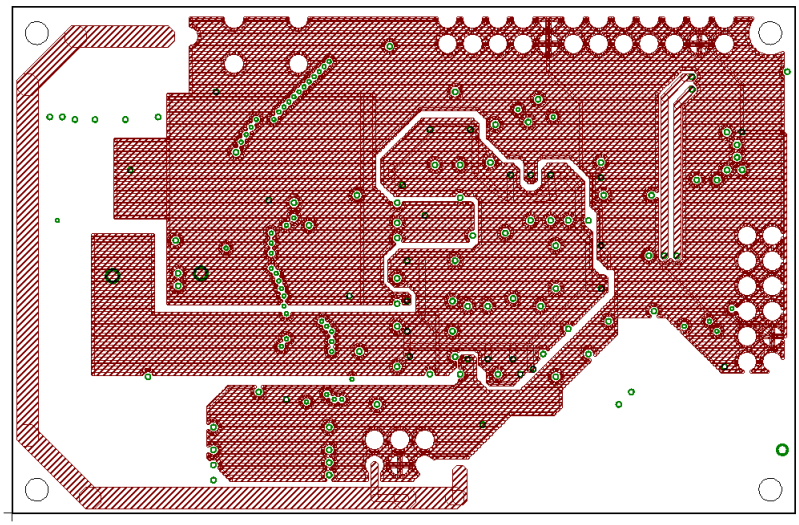
TOP Layer de placa versión final

BOTTOM Layer



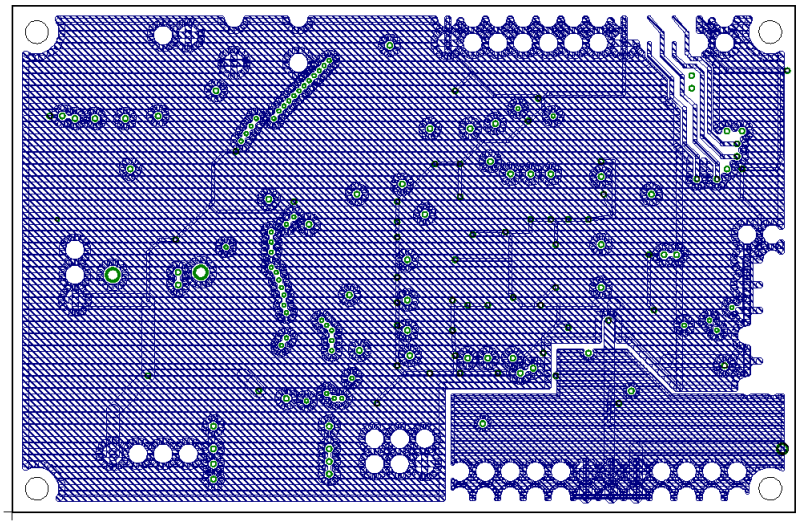
BOTTOM Layer de placa versión final

POWER Layer



POWER Layer de placa versión final

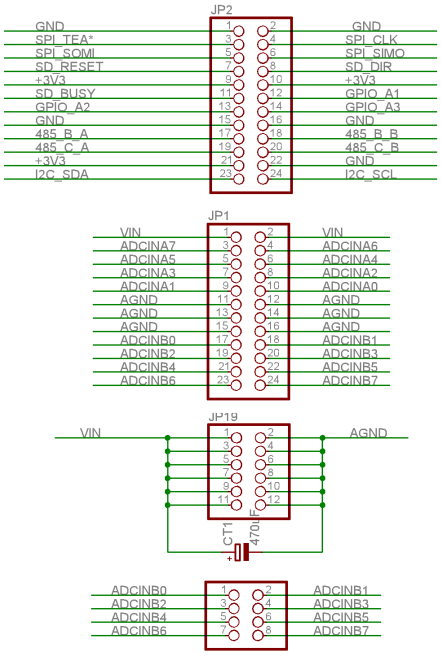
GND Layer



GROUND Layer de placa versión final

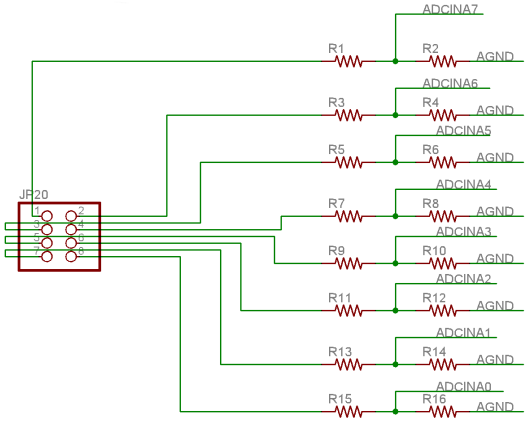
C. Esquemáticos de placa de Extensión de Periféricos

Esquemáticos análogos y conector extensión



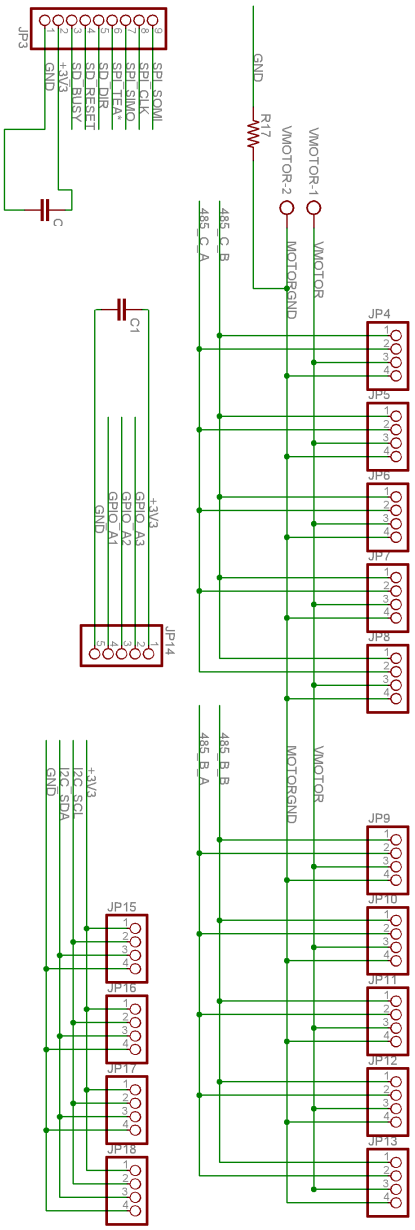
Esquemático de los análogos y conector extensión

Esquemáticos entrada análoga



Esquemático de entradas ADC

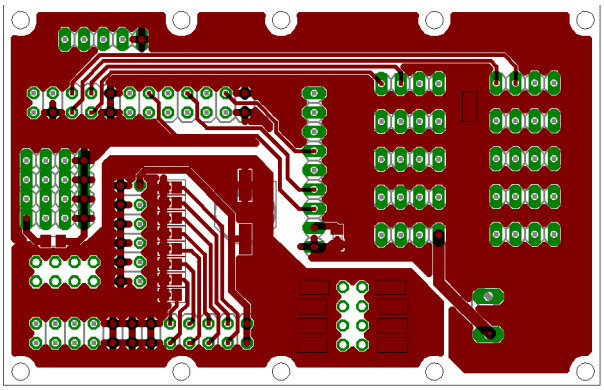
Esquemáticos entrada análoga



Esquemáticos de motores, I2C, SPI y GPIO

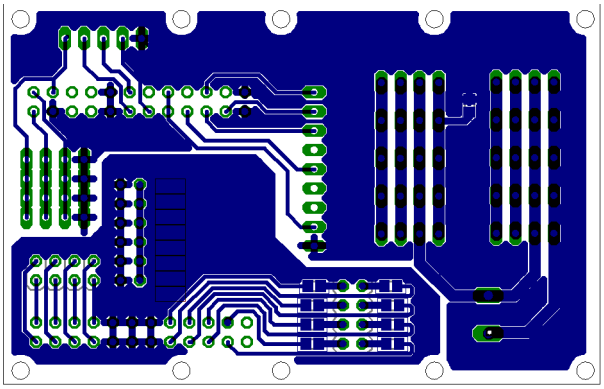
D. Layers de placa de Extensión de Periféricos

TOP Layer



TOP Layer de placa de extensión

BOTTOM Layer



BOTTOM Layer placa de extensión