

Universidad Nacional de Quilmes

Departamento de Ciencia y Tecnología
Ingeniería en Automatización y Control Industrial

CONTROL AUTOMÁTICO DEL EQUIPO UPDOWN

Olivieri, Ian Paulo

Director:

Pernia, Eric Nicolás

Co-director:

x

Jurado:

Safar, Felix

Juarez, José

y

z

Presentación: Septiembre de 2017
Quilmes, Buenos Aires, Argentina.

Resumen

Resumen del proyecto (1 carilla)

Índice de contenidos

1	Introducción	7
1.1	Marco temático	7
1.1.1	Esculturas cinéticas	7
1.1.1.1	Definición	7
1.1.1.2	Aplicaciones y estado actual del arte	8
1.1.2	Sistemas de iluminación	9
1.1.2.1	Equipos de luces	9
1.1.2.2	Consolas de control de luminaria	10
1.2	DMX	11
1.2.1	Definición e historia	11
1.2.2	Capa física	11
1.2.2.1	Cableado y conectores	11
1.2.2.2	Topología	12
1.2.2.3	Señal	12
1.2.3	Capa de enlace de datos	12
1.2.3.1	Subcapa de control de enlace lógico	12
1.2.3.2	Subcapa de control de acceso al medio	14
1.3	Updown	14
1.3.1	Definición	14
1.3.2	Descripción del sistema	15
1.3.2.1	Fuente de alimentación	15
1.3.2.2	Entrada y salida DMX	15
1.3.2.3	Dip-switch	16
1.3.2.4	Freno	16
1.3.2.5	Fin de carrera	16

1.3.2.6	Sistema motor	16
1.3.2.7	Placa de control	17
1.3.3	Resumen de entradas y salidas del sistema	17
1.3.3.1	Entradas	17
1.3.3.2	Salidas	17
1.4	Justificación del proyecto	18
1.5	Objetivos	18
1.5.1	REQ-01	18
1.5.2	REQ-02	18
1.5.3	REQ-03	18
1.5.4	REQ-04	19
1.5.5	Otros	19
2	Diseño	20
2.1	Descripción del capítulo	20
2.2	REQ-01	20
2.3	REQ-02	21
2.3.1	Esquema de control	21
2.3.2	Modelo de la planta	22
2.3.3	Procedimiento para el diseño de control	23
2.3.4	Relación entre cuentas de encoder y distancia	23
2.3.5	Velocidad máxima	24
2.4	REQ-03	24
2.4.1	Corte de correa	24
2.4.2	Fin de carrera	25
2.4.3	Pérdida de DMX	25
2.5	REQ-04	26

2.6	Firmware del updown	26
2.6.1	Aviso de confidencialidad	26
2.6.2	Convenciones	27
2.6.3	Librerías de bajo nivel	27
2.6.4	Librerías de alto nivel	28
2.6.5	Diagrama de módulos	29
3	Desarrollo	30
3.1	Descripción del capítulo	30
3.2	Firmware del updown - Librerías de bajo nivel	30
3.2.1	DigitalIO	30
3.2.1.1	Uso	30
3.2.1.2	Implementación	31
3.2.2	ADC	31
3.2.2.1	Uso	32
3.2.2.2	Implementación	32
3.2.3	PWM	32
3.2.3.1	Uso	32
3.2.3.2	Implementación	32
3.2.4	EXINT	32
3.2.4.1	Uso	33
3.2.4.2	Implementación	33
3.2.5	UART	33
3.2.5.1	Uso	33
3.2.5.2	Implementación	33
3.2.6	SUART	33
3.2.7	Tick	33

3.2.7.1	Uso	33
3.2.7.2	Implementación	33
3.3	Controlador	33
3.3.1	Relación entre cuentas de encoder y distancia	33
3.3.2	Determinación de la velocidad máxima	34
3.3.3	Obtención del período de muestreo	34
3.3.4	Obtención del modelo de la planta	34
3.3.5	Obtención del controlador	34
3.3.6	Ajuste del controlador	34
3.4	Dipswitch	34
3.5	Firmware del updown - Librerías de alto nivel	34
3.6	Firmware del updown - Función principal	34
4	Implementación	35
4.1	Descripción del capítulo	35
4.2	Configuración de la HOG	35
4.3	Validación	35
5	Conclusiones	36
5.1	c1	36

Índice de figuras

1.1	Ejemplo de escultura cinética movida por aire, por Anthony Howe. Fuente: LINK al video	7
1.2	Escultura cinética en el museo BMW. Fuente: LINK al video	8
1.3	Escultura cinética por parte de Build Up. Fuente: LINK al video	9
1.4	Shapeshifter, de High End Systems. Fuente: LINK al video	9
1.5	Consola Hog4, de High End Systems. Fuente: LINK a la imagen	10
1.6	Cable DMX con conector XLR5. Fuente: wikipedia	11
1.7	Conexionado en una red DMX. Fuente: wikipedia	12
1.8	Formato de la trama DMX. Fuente: LINK	13
1.9	Imagen del Updown, desarrollado por la empresa Blackout	14
1.10	Diagrama conceptual del equipo Updown	15
1.11	Diagrama del sistema motor	16
1.12	Diagrama del sistema motor	17
2.1	Esquema de control a implementar	22
2.2	Modelo mecánico de la planta	23
2.3	Diagrama de uno de los 3 divisores del dipswitch	26
2.4	Diagrama de módulos del firmware	29
3.1	Diagrama del módulo DigitalIO	32
3.2	Diagrama del módulo ADC	32

Introducción

1.1 Marco temático

1.1.1 Esculturas cinéticas

1.1.1.1 Definición

Las esculturas cinéticas (kinetic sculpture en inglés) son estructuras tridimensionales en donde el movimiento es una parte fundamental del conjunto. Para lograr el efecto de movimiento en el espacio estos sistemas se construyen con partes móviles que pueden cambiar de posición ya sea naturalmente por acción del viento, como se ve en la figura 1.1, o de manera forzada.



Figura 1.1: Ejemplo de escultura cinética movida por aire, por Anthony Howe. Fuente: [LINK al video](#)

1.1.1.2 Aplicaciones y estado actual del arte

Al ser obras que caen dentro del campo artístico suelen presentarse en museos y utilizarse para fines decorativos ya sea en parques o eventos. Sin embargo, el nivel de ingeniería y diseño que algunas de ellas requieren las tornan un interesante desafío intelectual y creativo.

Las aplicaciones puntuales de estructuras cinéticas a las que se hará foco en este informe, debido a la naturaleza del proyecto final, son aquellas en donde el efecto espacial se logra a través del movimiento en el eje vertical de objetos esféricos mediante motores.

Un ejemplo de aplicación de estas características se puede ver en la figura 1.2. Allí se muestra una escultura presentada en el Museo de BMW, en Munich, Alemania, en donde 714 esferas metálicas son coordinadas para formas figuras como olas, gotas, y hasta la silueta de un auto.

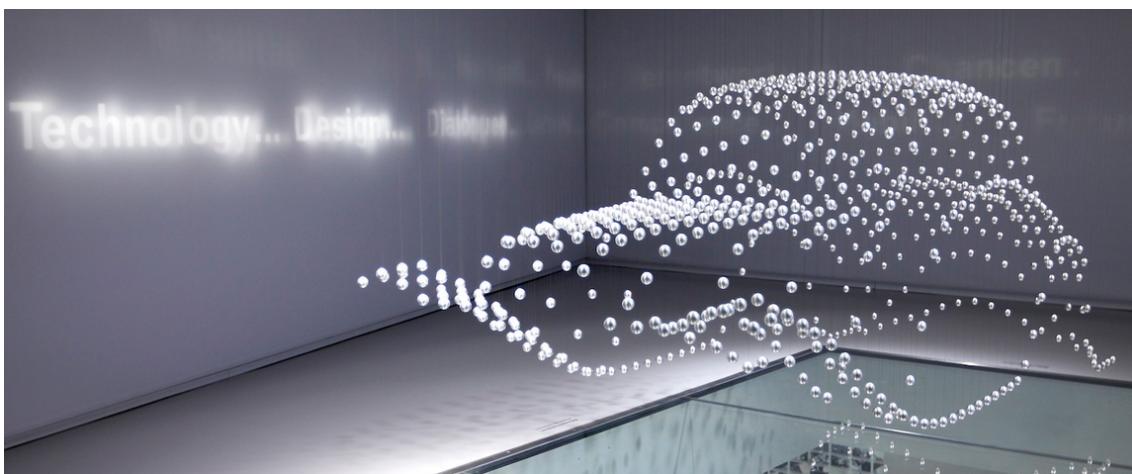


Figura 1.2: Escultura cinética en el museo BMW. Fuente: [LINK al video](#)

Otro ejemplo de aplicación se puede ver en la figura 1.3, en una obra presentada por la empresa Build Up en un centro comercial en Fukuoka, Japón. Allí se instalaron 1000 luminarias esféricas RGB dispuestas en una matriz de 25x40 para generar figuras tridimensionales como planos y gausseanas, entre otras. En este caso los efectos espaciales se logran coordinando el movimiento de cada esfera independientemente, cada una manejada por un equipo motorizado.

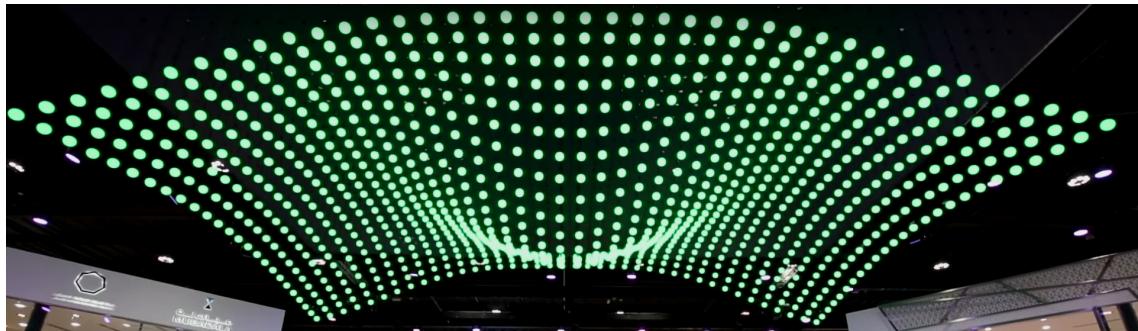


Figura 1.3: Escultura cinética por parte de Build Up. Fuente: [LINK al video](#)

1.1.2 Sistemas de iluminación

1.1.2.1 Equipos de luces

En cualquier espectáculo o evento la iluminación es una parte vital del show, y a medida que estos fueron evolucionando también lo hicieron los equipos de luces. Partiendo de aparatos fijos en donde solo se podía variar la intensidad de luz, se pueden conseguir hoy en día dispositivos complejos con decenas de parámetros controlables.

Un notable ejemplo es el **Shapeshifter**, figura 1.4, que cuenta con 7 módulos leds que pueden ser manejados independientemente.



Figura 1.4: Shapeshifter, de High End Systems. Fuente: [LINK al video](#)

En el caso del sistema visto en la figura 1.3, los parámetros controlables de los equipos son la posición, velocidad y colores de cada esfera.

1.1.2.2 Consolas de control de luminaria

Para controlar los sistemas de luces es necesario utilizar unas consolas especiales. Estas se comunican con las luminarias utilizando el estándar **DMX** y le indican a cada equipo el valor de sus parámetros en todo momento.

La manera más común para generar un efecto es indicando la progresión de uno o más parámetros desde un tiempo inicial a uno final. Al cambio de los parámetros entre 2 instantes de tiempo se las llama *cues*, o entradas, y cuyo conjunto forma los efectos. Dentro de las consolas que hay en el mercado para este tipo de control de equipos se pueden destacar las **consolas hog 4** de High End Systems, como la que se muestra en la figura 1.5.

Otra manera generarlos es a partir de equipos y softwares, como el **Madrix**, que tienen la capacidad de convertir videos a variaciones de parámetros, lo cual lo hace especialmente útil cuando se quieren crear **efectos lumínicos complejos**.



Figura 1.5: Consola Hog4, de High End Systems. Fuente: [LINK a la imagen](#)

1.2 DMX

1.2.1 Definición e historia

DMX, de *Digital MultipleX*, es un estándar de comunicación digital ampliamente utilizado para el control de sistemas de iluminación.

El estándar DMX512, donde 512 significa que se envían 512 piezas de información, fue creado por la *United States Institute for Theatre Technology* (USITT) en 1986 y transformado en DMX512/1990 tras una revisión de la USITT. En 1998 la *Entertainment Services and Technology Association* (ESTA) cuadró DMX dentro de los estándares ANSI, modificación que fue aprobada por el instituto (ANSI) en 2004. Finalmente, en 2008 DMX tuvo una nueva revisión y se llegó a la versión actual llamada "E1.11 – 2008, USITT DMX512-A", o simplemente DMX512-A. A pesar de esto, el nombre comúnmente conocido del estándar es simplemente DMX, aunque no es indistinto ya que hay diferencia de compatibilidad entre las diferentes versiones.

1.2.2 Capa física

1.2.2.1 Cableado y conectores

DMX emplea el estándar EIA-485 como capa física, por lo que emplea por lo menos 3 líneas; A, B y C, en donde A y B son los datos que se transmiten, y C es masa. Los conectores utilizados son los XLR, tanto de 5 como de 3 pinos. Un ejemplo del cable se puede ver en la figura 1.6



Figura 1.6: Cable DMX con conector XLR5. Fuente: wikipedia

1.2.2.2 Topología

La red de DMX consiste en un maestro y varios esclavos, conectados con una topología de bus multidrop (MDB) con nodos conectados entre sí, lo que normalmente se denomina como topología *daisy chain*. En otras palabras, todos los equipos a controlar tienen una entrada y una salida conectadas entre sí, de manera tal de que se puede conectar un equipo y apartir de este equipo conectar el siguiente, y así sucesivamente, como se ve en la figura 1.7. Esto permite que el conexionado sea simple y que la red pueda ser fácilmente extendida.

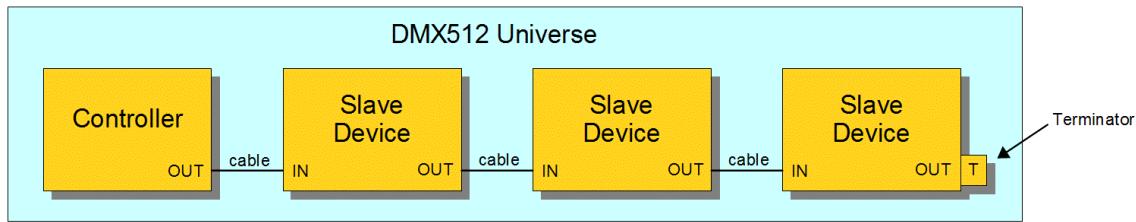


Figura 1.7: Conexión en una red DMX. Fuente: wikipedia

1.2.2.3 Señal

La señal de DMX es de tipo diferencial, como es indicado en EIA-485, de 5 Volts de pico, y los datos se envían asincrónicamente de manera serie con una tasa de transmisión de 250Kbits por segundo que equivale a una duración de bit de $4\mu s$.

1.2.3 Capa de enlace de datos

1.2.3.1 Subcapa de control de enlace lógico

La trama DMX, visible en la figura 1.8 consta de las siguientes partes:

- **Idle** y **MTBP** (Mark Time Between Packets): estado de HIGH en la línea que indica la ausencia de señal de DMX. En caso de que se supere el tiempo máximo de 1 segundo sin datos, se considera que hubo una pérdida en la conexión.
- **Break**: estado de LOW en la línea utilizado para separar las tramas entre sí.
- **MAB** (Mark After Break): estado en HIGH enviado luego de un Break. Este estado suele traer problemas de compatibilidad ya que fue cambiado de una duración de $4\mu s$ a $8\mu s$ en la versión de DMX512 de 1990.

- **Slots:** son datos con formato 8N2 (un bit de start (LOW), 8 bits de datos, 2 bits de stop (HIGH) y sin paridad), y pueden ser:
 - **SC** (Start Code): es el Slot 0, enviado luego de un MAB, para indicar el principio del payload. Todos los bits de datos equivalen a LOW en este estado.
 - **Channels:** contienen la información que quiere ser enviada a los equipos de DMX. El conjunto de los 8 bits de datos pueden tomar un valor de 0 a 255. En total pueden haber un máximo de 512 canales enviados.
- **MTBF** (Mark Time Between Frames): estado opcional de HIGH en la linea que puede agregarse antes del bit de start de cada canal.

Señal	Mínima	Máxima	Típica
Idle/MTBP	0	1 seg	No especificada
Break	88μs	1seg	88μs
MAB	-	-	8μs
Bit	-	-	1/250KHz = 4μs
Slot (11 bits)	-	-	44μs
MTBF	0	1 seg	No especificada

Tabla 1.1: Duración de las señales que componen la trama de DMX

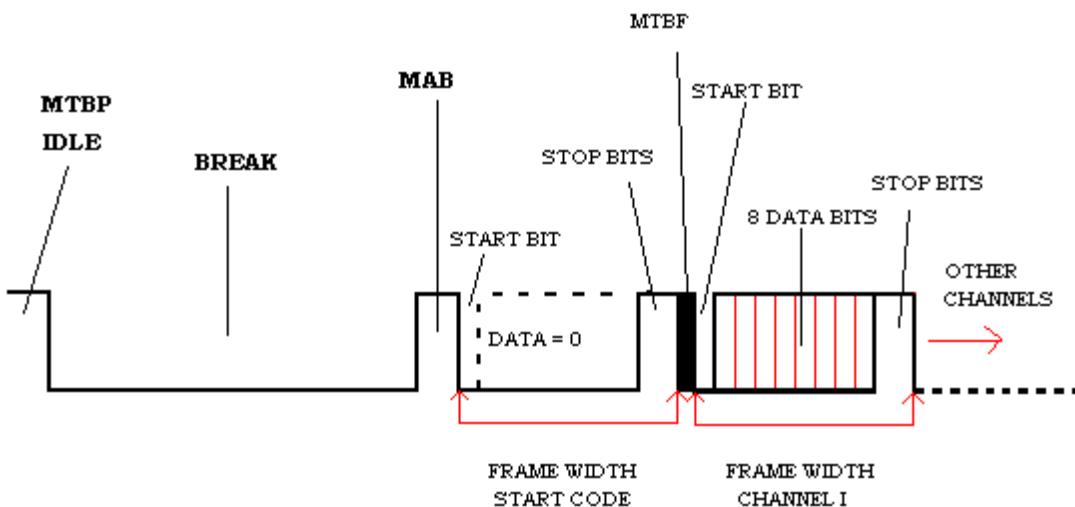


Figura 1.8: Formato de la trama DMX. Fuente: [LINK](#)

A cada trama con 512 canales se la llama "universo de DMX". Si se necesitan enviar más de 512 canales se utilizan más universos.

1.2.3.2 Subcapa de control de acceso al medio

DMX no implemente ningún mecanismo de control de acceso al medio debido a que el único transmisor es la red es el maestro y todos los esclavos reciben la misma información.

1.3 Updown

1.3.1 Definición

El **Updown**, que se puede ver en la figura 1.9, es básicamente una grúa que sube y baja una carga conforme a comandos recibidos por una equipo que utilice el protocolo DMX, como puede ser una consola de control de luminaria. La distancia máxima a la que la carga puede bajar es de entre 4 y 5 metros.

Este producto fue concebido en **Blackout**, una empresa productora y proveedora de tecnología cuyo objetivo es generar contenido audiovisual para grandes eventos. Blackout tomó interés en esculturas cinéticas como la de la figura 1.3 pero se encontró con el problema que el costo de importación de los equipos utilizados para tales fines, sumado a su precio unitario, era muy elevado. Por este motivo, decidió comenzar el desarrollo de un producto propio y nacional para alcanzar su objetivo.

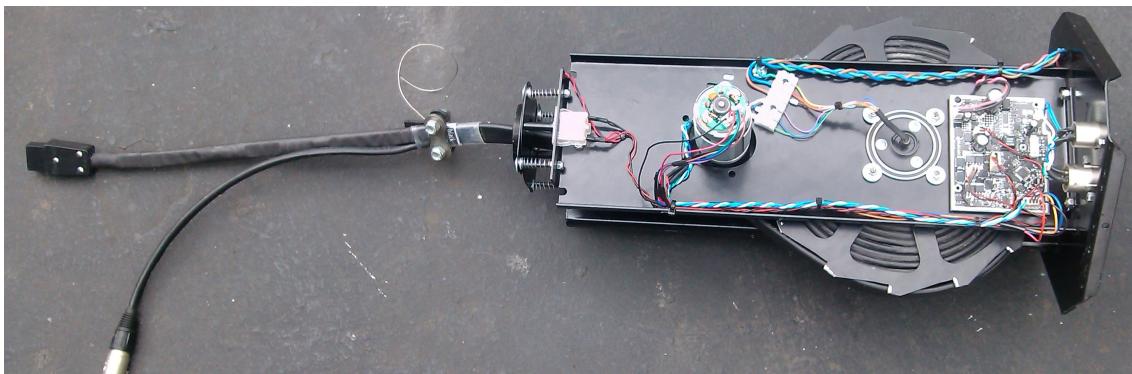


Figura 1.9: Imagen del Updown, desarrollado por la empresa Blackout

1.3.2 Descripción del sistema

En la figura 1.10 se presenta el diagrama general del equipo updown. Allí se pueden identificar todos elementos electrónicos y mecánicos que conforman el sistema, los agentes externos del sistema, y la comunicación entre ellos.

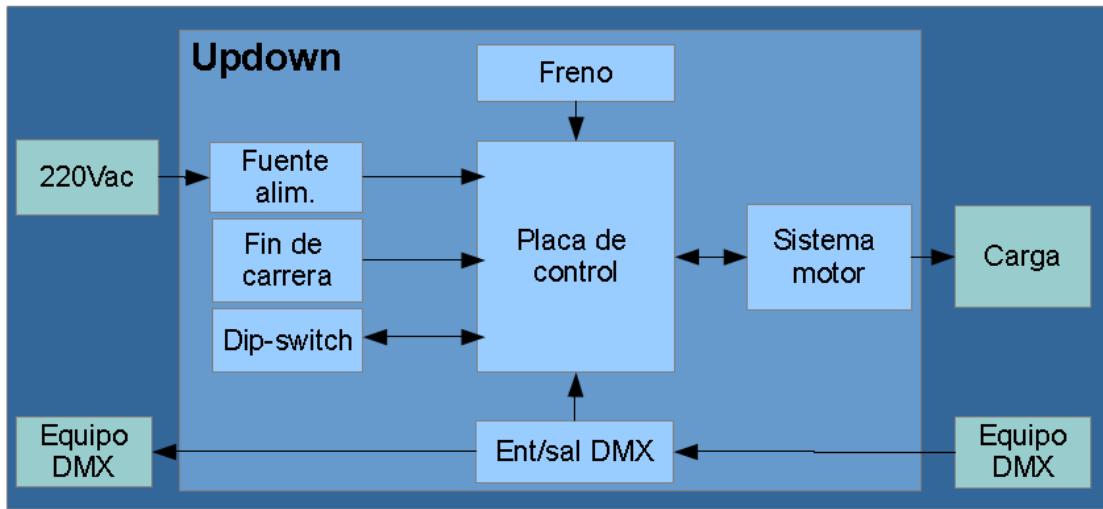


Figura 1.10: Diagrama conceptual del equipo Updown

1.3.2.1 Fuente de alimentación

La fuente de alimentación es la encargada de proveer la potencia necesaria para el funcionamiento del sistema motor, y de placa de control junto con sus periféricos. Esto lo logra convirtiendo la tensión de red de 220Vac a una señal continua de 24Vcc, entregando 4.5A máximo.

1.3.2.2 Entrada y salida DMX

La señal DMX proveniente del master de la red, otro Updown, o cualquier otro equipo esclavo dentro del mismo universo DMX ingresa al sistema para ser procesada por la placa de control. Su función es indicarle al equipo la referencia de posición y velocidad, y los parámetros que correspondan a la carga.

Además, para cumplir con el estándar DMX, la señal de entrada se replica mediante un puente a una salida para que otro equipo pueda ser conectado a la red.

1.3.2.3 Dip-switch

El dipswitch consta con varios divisores resistivos, 10 switchs y un led indicador. Su función es seleccionar el canal inicial de DMX con el que el equipo trabajará.

1.3.2.4 Freno

El freno consta de una bobina, desactivada por defecto, que mueve una varilla metálica que traba el carrete que contiene la polea. Al activar la bobina la varilla libera al carrete, permitiendo el libre movimiento de la carga.

1.3.2.5 Fin de carrera

El fin de carrera consta de un par de pulsadores en la base del equipo. Eventos como enrollar completamente la polea activa alguno de los pulsadores, dandole aviso a la placa de control.

1.3.2.6 Sistema motor

El elemento principal del sistema motor es un motor de continua de 24V con un encoder AB en su eje. Este motor mueve un carrete en donde se enrolla la polea que sostiene la carga. El carrete, llamado disco, tiene unas marcas que son sensadas por un encoder en la placa de control para tener una segunda referencia de posición.

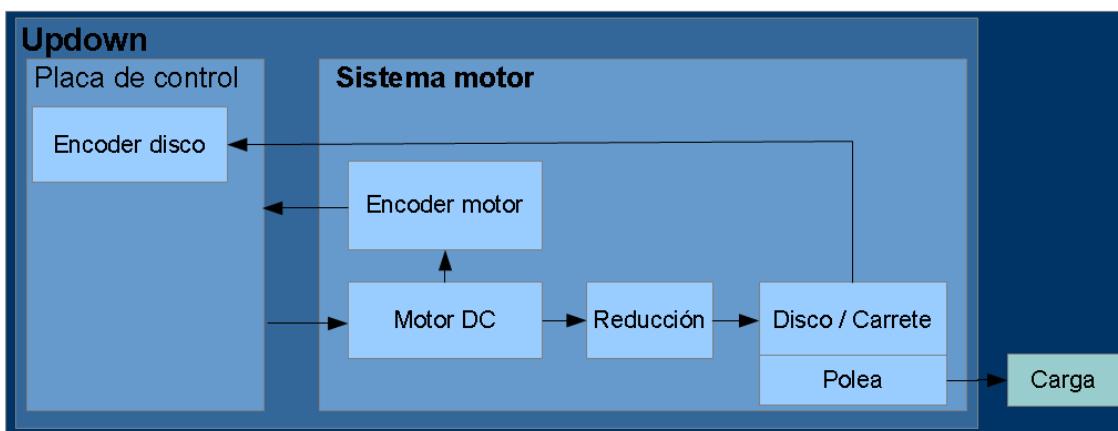


Figura 1.11: Diagrama del sistema motor

1.3.2.7 Placa de control

La placa de control contiene electrónica para el manejo de las entradas y salidas del sistema, y un controlador para manejar todos los periféricos siguiendo la lógica deseada.

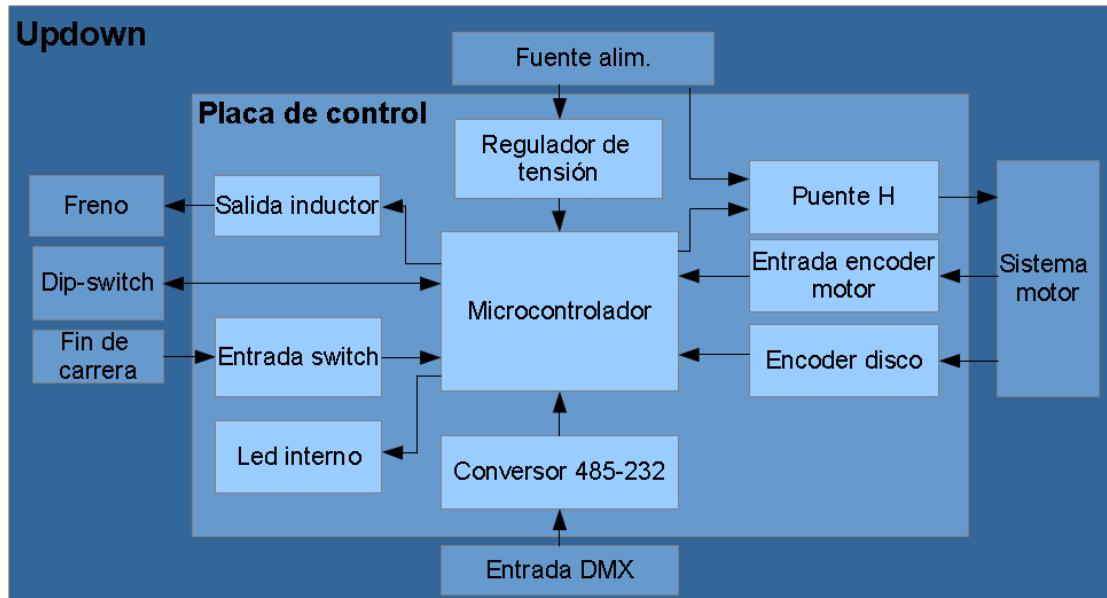


Figura 1.12: Diagrama del sistema motor

1.3.3 Resumen de entradas y salidas del sistema

1.3.3.1 Entradas

Señal de DMX, Fin de carrera, Dip-switch, Encoder AB de disco, Encoder AB de motor.

1.3.3.2 Salidas

Freno, Motor, Led interno (led indicador de la Placa de control), Led externo (led indicador del Dipswitch).

1.4 Justificación del proyecto

Durante el desarrollo del updown la empresa se dió cuenta que con los recursos que contaban, tiempo en particular, no podían concretar los requerimientos que buscaban que el updown tuviera. En esta instancia la mayoría de la mecánica y la electrónica estaba hecha, solo faltaba resolver: el desarrollo de un dipswitch para el direccionamiento de DMX, la programación del microcontrolador para manejar todo el sistema y generación software de prueba para el armado de nuevos equipos.

Por este motivo, Blackout se vió en la necesidad de buscar a alguien externo a la empresa para darle fin al proyecto, lo que presentó la oportunidad de realizar el trabajo de concluir el desarrollo del equipo, que es en lo que este proyecto final se basa.

1.5 Objetivos

Para dar conclusión al proyecto se deben cumplir con los requerimientos no resueltos del producto. Estos son:

1.5.1 REQ-01

Contar con el software necesario para el manejo de todas las entradas y salidas del sistema.

1.5.2 REQ-02

Lograr que las cargas manejadas por los equipos se muevan a la velocidad y posición indicadas mediante una consola DMX.

1.5.3 REQ-03

Manejar errores y excepciones de hardware para lograr que el producto sea seguro, siendo que será instalado en eventos con un alto nivel de concurrencia.

Dentro de estos errores se encuentran: corte de correa, pérdida de señal DMX y accionamiento indebido del fin de carrera.

1.5.4 REQ-04

Concluir el desarrollo del dipswitch.

1.5.5 Otros

Además, de ser necesario para el cumplimiento del resto de los requerimientos, se deberán plantear, y de ser posible realizar, los cambios mecánicos y electrónicos que hagan falta.

Diseño

2.1 Descripción del capítulo

En este capítulo se analizarán los requerimientos presentados en la sección 1.5 del capítulo 1 con el objetivo de determinar qué se planea hacer para cumplirlos.

2.2 REQ-01

Para cumplir este requerimiento se debe contar con herramientas de software que permita el manejo de: freno, leds indicadores, fin de carrera, dipswitch, encoders AB de disco y motor, motor y DMX. Además, como la placa de control no cuenta con un puerto de debug es necesario tener algún canal de comunicación alternativo, como un puerto serie por software. También será necesario temporizar ciertas partes del programa, por lo que se deberá poder manejar un timer.

El microcontrolador utilizado en la placa de control es el Atmega328p, y cuenta con periféricos para manejar todas las entradas y salidas del sistema por lo que es una buena elección para el proyecto. El único inconveniente que presenta es que es un microcontrolador de 8 bits con relativamente baja memoria y sin optimizaciones para operaciones de punto flotante.

En sistemas embebidos, la manera de manejar el hardware del microcontrolador, entre otras cosas, es a través de librerías. Estas son un conjunto de funciones o procedimientos externos a la aplicación a desarrollar que le permiten al usuario manejar ciertos aspectos del sistema de forma más fácil.

Una opción sería utilizar librerías ya desarrolladas, entre las cuales se destacan las de Arduino. Arduino, una plataforma de programación de sistemas embebidos muy popular, utiliza en sus placas microcontroladores de marca Atmel. En particular, el *Arduino UNO* utiliza el Atmega328p, por lo que existen funciones provistas por Arduino para el manejo de los periféricos del microcontrolador que se utiliza en el proyecto.

La otra opción sería hacer librerías a medida para el proyecto, desarrollando las funciones necesarias para el manejo de los periféricos que se necesiten para que se comporten exactamente como uno desea.

En la tabla 2.1 se presenta una comparación entre ambas opciones.

El mayor problema con las librerías de Arduino es que para que cuadren en proyectos grandes como estos requieren modificaciones, y las funciones provistas deben ser analizadas para verificar que no se usen elementos como delays bloqueantes ni operaciones

de punto flotante indiscriminadamente, ya que deterioran el rendimiento.

Como el objetivo es hacer que el sistema sea lo más confiable posible, y teniendo en cuenta que el tiempo de desarrollo no es un factor crítico, **se desarrollarán librerías propias para manejar los periféricos.**

	Arduino	Propia
Pros	<ul style="list-style-type: none"> - Listas para usar - Altamente testeadas - Mantenidas por una comunidad 	<ul style="list-style-type: none"> - Bajo uso de recursos - Confiabilidad - Predictibilidad
Cons	<ul style="list-style-type: none"> - Genéricas (poco optimizadas) - Necesitan modificaciones para ser útiles - Requieren análisis 	<ul style="list-style-type: none"> - Tiempo de desarrollo alto por defecto - Investigación y estudio

Tabla 2.1: Comparación entre el uso de librerías de Arduino vs el desarrollo de unas propias

2.3 REQ-02

2.3.1 Esquema de control

Para cumplir con este requerimiento la carga se debe mover en respuesta a una referencia de posición y otra de velocidad indicadas por una consola DMX. Para esto se utilizará el esquema de control mostrado en la figura 2.1, siendo: p la posición, v la velocidad, C_p el controlador de posición, C_v el de velocidad, ϵ_p y ϵ_v los errores de posición y velocidad, u la acción de control, G la planta, y rp_dmx y rv_dmx las referencias de posición y velocidad dadas por la consola DMX.

La elección de este esquema fue debido a su simpleza: un controlador de velocidad se encarga de mantener la velocidad igual a la referencia, mientras que uno de posición le indica al de velocidad que debe frenar cuando la posición objetivo está por ser alcanzada. Para evitar que el controlador de posición le indique al de velocidad una referencia mayor a la permitida, se utiliza un limitador en la referencia de velocidad. Similarmente, como la acción de control será el ciclo de trabajo del pwm que mueve al motor, se utiliza un segundo limitador para evitar que el ciclo de trabajo supere el 100 %.

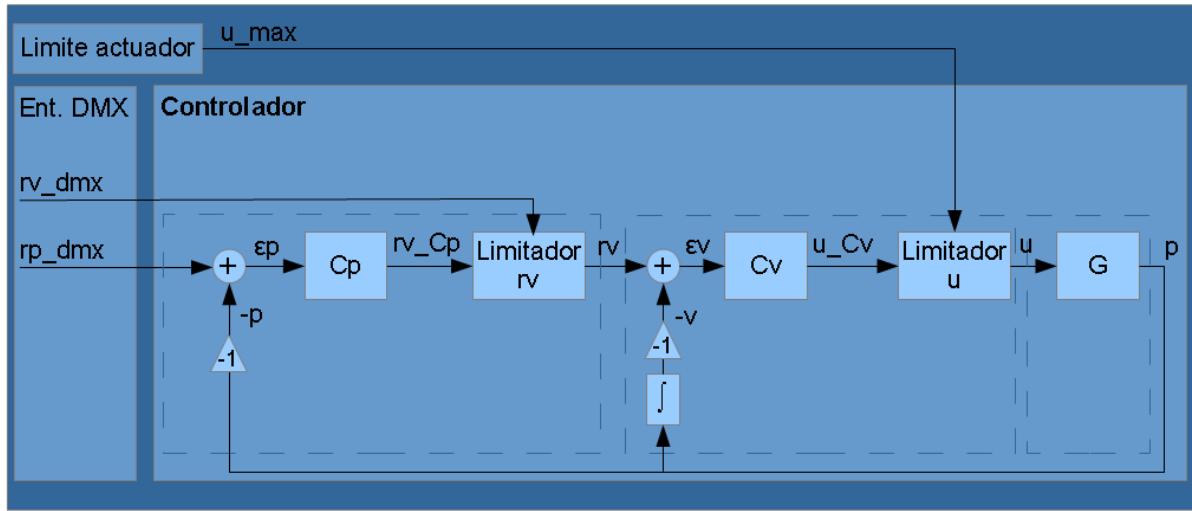


Figura 2.1: Esquema de control a implementar

2.3.2 Modelo de la planta

Para este proyecto la planta será considerada como el conjunto motor-transmisión-carga. Un esquema mecánico de la planta se muestra en la figura 2.2. Su funcionamiento es el siguiente: parte de la potencia eléctrica que ingresa al motor se transforma a mecánica y se utiliza para mover el piñón que se encuentra conectado al eje del motor. El piñón, a través de una cadena ANSI 25, transfiere su energía a la corona, que hace girar un disco adosado a ella. Este disco, o carrete, contiene en su interior el cable que sostiene a la carga, enrrollandolo o desenrrollandolo para subirla o bajarla. Como la carga del updown puede necesitar ser alimentada y recibir señal de DMX, el cable utilizado contiene en su interior contiene 2 cables con ambas señales.

Ahora, para diseñar el controlador se debe encontrar un modelo matemático de la planta. Una opción es determinar las ecuaciones diferenciales que gobiernan el sistema aplicando la segunda ley de newton, y determinando el modelo matemático de un motor de continua mediante identificación. El problema es que por motivos constructivos del equipo existen muchas piezas que generan roces en diferentes partes del recorrido de la carga, lo que quiere decir que hay una perturbación cuya dinámica es totalmente desconocida y que depende de varios factores, como la distancia y el peso de la carga. Por lo tanto, se abordará el problema con un método más simple: determinar el modelo de todo el conjunto mediante identificación, midiendo la respuesta del sistema ante ciertas entradas. Como el objetivo es que varios updown se comporten lo más parecido posible, se realizarán las pruebas para por lo menos 2 updown.

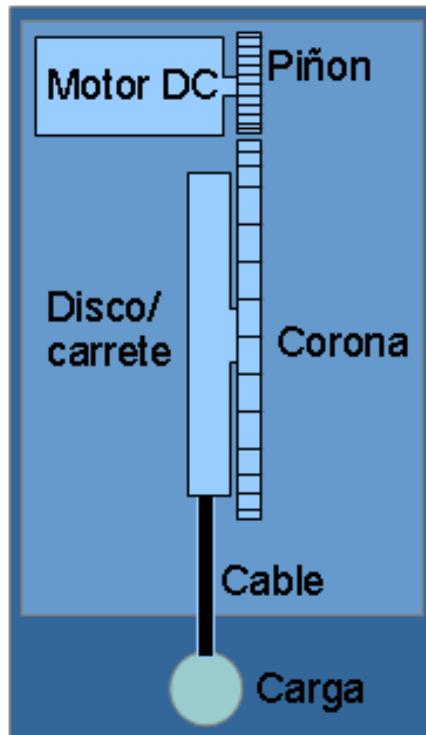


Figura 2.2: Modelo mecánico de la planta

2.3.3 Procedimiento para el diseño de control

Primero se determinará el período de muestreo y actuación inyectándole un escalón unitario a planta.

Luego, se obtendrá el modelo de la planta mediante identificación paramétrica, haciendo uso de un método de estimación de tipo offline (no recursivo).

Luego se diseñarán los controladores de posición y velocidad en base al modelo de la planta obtenido

Finalmente se validarán los controladores hayados haciendo pruebas directamente sobre el updown.

2.3.4 Relación entre cuentas de encoder y distancia

A medida que el motor gira desenrolla el cable para subir o bajar la carga. El problema es que dependiendo cuán enrrollada está el cable en el disco un mismo número de cuentas de encoder se puede traducir en distintas distancias recorridas por la carga. Por ejemplo, si el cable está completamente enrollada el radio es mayor, por lo que si giro 360 grados el largo desenrollado será uno, mientras que si el cable se encuentra

parcialmente desenrollada el radio es menor, por lo que la distancia descendida en un giro de 360 grados del carrete será otra (menor que la anterior).

Para determinar esta relación se harán marcas cada un metro en el cable, tomando como 0 cuando esta está completamente enrollada, y para cada una se anotará el valor del encoder del motor, que es el que más resolución tiene. Con estos datos se construirá una tabla, se encontrarán varios polinomios que fitteen los datos y se determinará cuál de ellos es el que se implementará en software.

2.3.5 Velocidad máxima

Una tarea a resolver es determinar cuál es la velocidad máxima posible para una carga determinada.

Para determinar este valor se medirá la velocidad para una carga de 3Kg en subida aplicando sobre el motor un pwm con ciclo de trabajo de 100%.

La posición máxima posible queda dada por condiciones de diseño: 4 metros

2.4 REQ-03

Los posibles errores detectables que se pueden tener en el equipo son:

2.4.1 Corte de correa

La transmisión de potencia entre el piñón en el eje del motor y la corona en el disco se hace a través de una correa. El problema es que durante las pruebas, la empresa observó que la correa podría cortarse.

La manera que tiene el equipo de detectar este error es mediante un segundo encoder AB que mide el giro del disco. Como la relación de vueltas entre el disco y el motor es proporcional, cualquier desviación grande de esta proporcionalidad indica que uno se está moviendo más rápidamente que el otro. Esto podría interpretarse como que la correa fue cortada o que alguno de los encoders dejó de funcionar, 2 errores válidos de hardware.

Para poder detectar estos errores lo primero que se hará es determinar la relación entre las cuentas del encoder del disco y las del motor. Luego verificará en el firmware que esta se cumpla en todo momento, bajo un cierto nivel de error aceptable. En caso de no

cumplirse se accionará el freno, se detendrá el equipo completamente y se indicará sobre la existencia del error mediante el led de la placa de control (led interno).

2.4.2 Fin de carrera

La función principal del fin de carrera, un par de pulsadores en la base del equipo, es indicar cuándo el cable que sostiene la carga está completamente enrollada para determinar la posición 0 o "home" del equipo, dado que este puede estar en cualquier posición al ser encendido.

La función secundaria del fin de carrera es detectar posibles eventos indeseados. Por ejemplo, en su punto más bajo la carga se encontrará a 4 metros por debajo del 0 del updown, por lo que podría comenzar a oscilar debido a fuertes vientos o personas que muevan la carga. Por cómo está construido el fin de carrera si la carga oscila más allá de un ángulo de aproximadamente 30 grados el fin de carrera se accionará dando aviso de esto.

Para poder detectar estos errores se verificará que el fin de carrea no sea presionado fuera de la rutina de calibración de (homing). En caso de que sea presionado en estas condiciones se detendrá el equipo, se esperará un tiempo a que la oscilación baje, y intentarán reanudar las operaciones normales. En caso de que el fin de carrera siga presionado se accionará el freno, se detendrá el equipo completamente y se indicará sobre la existencia del error mediante el led de la placa de control (led interno).

2.4.3 Pérdida de DMX

El estándar DMX establece que el tiempo máximo entre un paquete de datos y otro, tiempo de IDLE o MTBP (ver tabla 1.1), es de 1 segundo. Esto quiere decir que si por 1 segundo no me llegaron nuevos datos se considera que se perdió la comunicación con el dispositivo DMX.

Para detectar la pérdida de señal de DMX se verificará constantemente que el último paquete haya llegado hace menos de 1 segundo. En caso de no cumplirse se asumirá que las referencias de posición y velocidad son 0 (equipo parado). Además, por una convención en equipos DMX, se indicará mediante el led en el dipswitch (led externo) cuándo el equipo recibe correctamente la señal de DMX y cuándo no, encendiéndolo y apagándolo rápidamente en el primer caso y lentamente en el segundo.

2.5 REQ-04

El propósito del dipswitch en el equipo es poder seleccionar el canal principal de DMX a partir del cual leer las referencias de posición y velocidad, dentro de los 512 canales que tiene un universo DMX. Para lograr esto cuenta con 10 switchs, 9 que determinan el canal de DMX ($2^9 = 512$) y uno extra para ampliaciones futuras. Estos conmutadores conectan o desconectan resistencias de unos divisores de tensión resistivos, variando el valor entregado por cada uno de ellos. Como el dipswitch cuenta con 3 divisores se tienen 3 salidas analógicas, que dan información de qué switchs están activados y cuales no. El esquema de uno de los divisores se puede ver en la figura 2.3. Para completar con el desarrollo del dipswitch se deben encontrar 5 valores de resistencias (R1,R2,R3,R4 y R).

Para hallar estos 5 valores se simulará el divisor de tensión en Matlab, probando con distintas combinaciones de resistencias hasta que los valores analógicos de la salida para cada combinación de resistencia estén lo suficientemente separados.

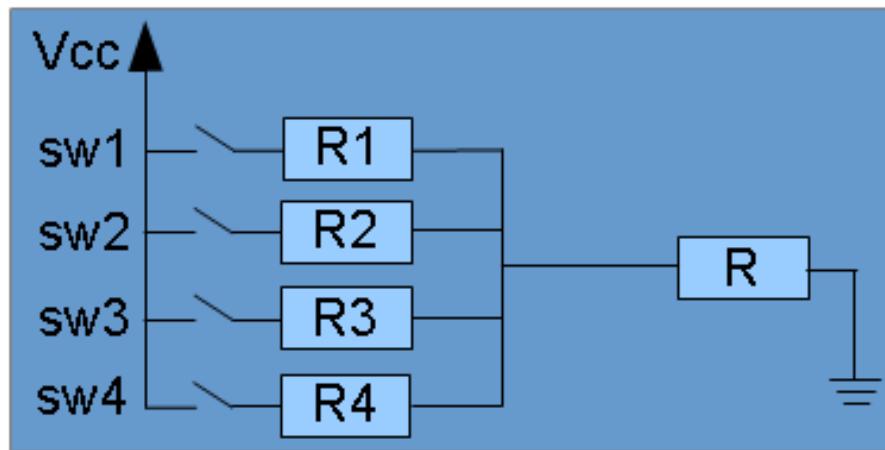


Figura 2.3: Diagrama de uno de los 3 divisores del dipswitch

2.6 Firmware del updown

2.6.1 Aviso de confidencialidad

Debido a políticas de confidencialidad de la empresa Blackout, el código fuente del firmware desarrollado para este proyecto no puede ser distribuido libremente. Es por este motivo que únicamente se describirá cómo se solucionaron los problemas, sin analizar el código puntualmente.

2.6.2 Convenciones

Con el objetivo de maximizar el encapsulamiento, las funciones dentro de los módulos se dividirán en internas y externas. Las internas podrán ser accedidas únicamente por otras funciones dentro del módulo, mientras que las externas también podrán ser accedidas por funciones de otros módulos. Para implementar esto se hará uso del sistema de separación de información en archivos .c (código) y .h (headers). Los .h tendrán las declaraciones de funciones externas y definiciones para que, al ser incluido por otros módulos, tengan acceso a las mismas.

En cuanto a variables, solo se utilizarán variables internas al módulo. En caso de que tengan que poder ser accedidas externamente se implementarán setters (para escribir un valor en la variable) y getters (para escribir el valor de la variable) según corresponda. Las funciones tendrán su nombre en formato *camelCase* (primera letra minúscula y el resto de las primeras letras de las palabras utilizadas en el nombre en mayúscula), los Define en *MAYUSCULA*, y los módulos en formato PascalCase (como el camel case pero con la primera letra en mayúscula). Finalmente, todas las funciones y definiciones que pueden ser accedidas desde afuera de un módulo llevarán como prefijo el nombre del módulo seguido de un guión bajo.

Además, debido a políticas de Blackout, todo el software desarrollado debe ser entendible por todo el equipo de trabajo. Esto quiere decir que una de las metas para las funciones de las librerías a crear es que permitan un flujo entendible, y que los nombres de las funciones sean parecidas a las de Arduino ya que lo que se acostumbraba a utilizar en la empresa para la programación de sistemas embebidos.

Para lograr este objetivo todas los módulos desarrollados tendrán una función de inicialización llamada "init", y de lectura y escritura, según corresponda, llamadas "read" y "write", respectivamente.

A continuación se presentan algunos ejemplos de las convenciones que se usarán. Función de inicialización del módulo 1: Modulo1_init(); Define del largo de un buffer genérico del modulo 3: Modulo3_BUFFGERENERICO.

2.6.3 Librerías de bajo nivel

Como se mencionó en la sección 2.2, se desarrollarán librerías para manejar los periféricos del Atmega328p y así poder controlar el hardware del sistema. Esto implica analizar la [hoja de datos del Atmega328p](#) e interactuar con los registros, un lugar de memoria utilizado para guardar información, que correspondan.

Asociadas a las librerías viene de la mano el concepto de módulo, que tiene que ver con agrupar funciones similares en un mismo paquete para desacoplar dependencias y separar al sistema en varios subsistemas que sean lo más independientes que se pueda. Esto ayuda a que el programa sea escalable y mantenable, por lo que todo el software

desarrollado se hará de manera modular.

Entonces, para manejar los periféricos se creará una librería de bajo nivel que constará de los siguientes módulos:

- **Entradas y salidas digitales**, para leer el estado del fin de carrera y comandar el freno y los leds indicadores interno y externo. Este módulo se llamará *DigitalIO*.
- **Entradas analógicas**, para leer las 3 salidas analógicas del dipswitch y poder determinar la posición de los selectores. Este módulo se llamará *ADC*, por *Analog to Digital Converter*.
- **Interrupciones externas** por cambio de estado de un pin, para llevar cuenta de los cambios de estado de los encoders AB y así saber la posición relativa. Este módulo se llamará *EXINT*, por *EXternal INTerrupts*.
- **PWM**, para el manejo del motor de continua. Este módulo se llamará *PWM*, por *Pulse Width Modulation*.
- **UART**, para la lectura de la señal DMX. Este módulo se llamará *UART*, por *Universal Asynchronous Receiver/Transmitter*
- **UART por software**, una UART implementada en pines genéricos que se utilizará para el debuggeo de la placa. Este módulo se llamará *SUART*, por *Software UART*.
- **Base de tiempo**, un timer utilizado para la temporización tareas y eventos. Este módulo se llamará *Tick*.

2.6.4 Librerías de alto nivel

Hay que tener en cuenta que no es lo mismo obtener un valor analógico leyendo un *registro* (lugar para el almacenamiento de datos) del microcontrolador, que determinar qué conmutadores del dipswitch están activados leyendo 3 valores analógicos. Por lo tanto, se necesita una librería que sea más abstracta que la del manejo de periféricos para poder utilizar el hardware de esta aplicación en particular.

Entonces, se creará una librería propia de la aplicación que constará de los siguientes módulos:

- **Dipswitch**, que incluye la lectura de las llaves del dipswitch, y la escritura del led indicador externo.
- **Grua**, que incluye el manejo del motor, del led interno, del freno y del fin de carrera.

- **DMX**, para la recepción de datos de la consola DMX.
- **Encoder**, para el manejo de los encoders AB del disco y del motor.
- **Controlador**, para la implementación del sistema de control

2.6.5 Diagrama de módulos

De estas 2 librerías se obtiene el diagrama de módulos presentado en la figura 2.4, el cual muestra su relación, dependencia y alcance.

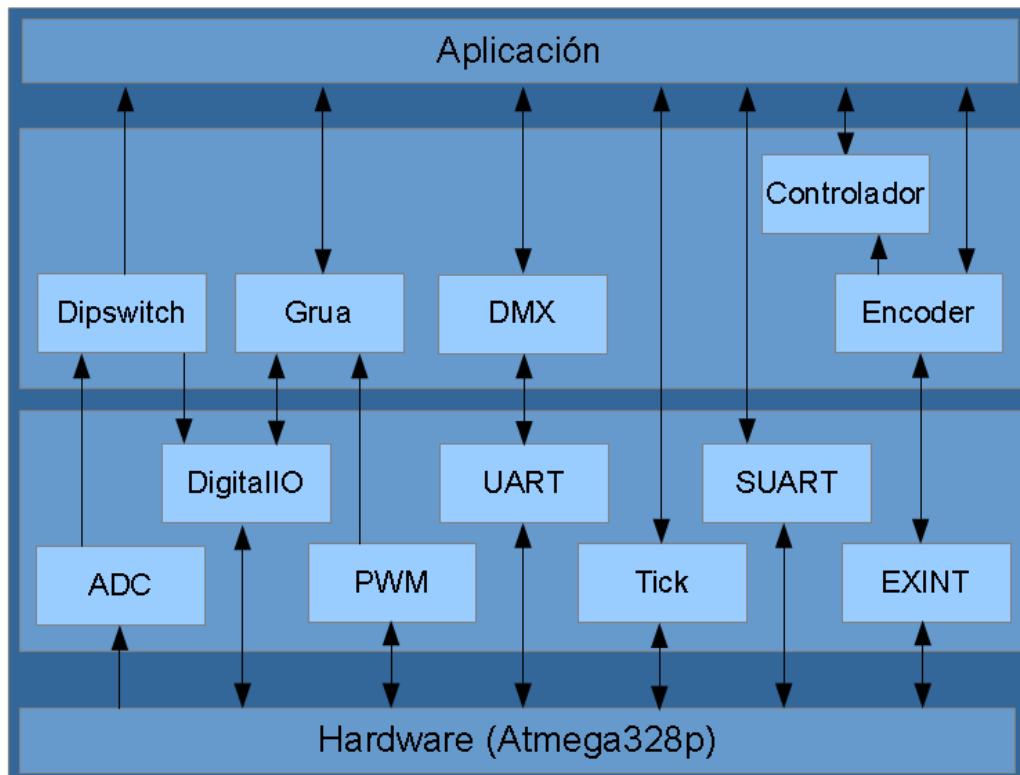


Figura 2.4: Diagrama de módulos del firmware

Desarrollo

3.1 Descripción del capítulo

En este capítulo se desarrollarán las soluciones propuestas en el capítulo de diseño. Esto implica detallar el proceso realizado, describir los cambios que se debieron hacer en caso de que el planteo inicial no haya funcionado, análisis de resultados, etc. Cada sección será redactada en el orden que se hicieron, debido a que unas partes dependen de otras para poder desarrollarse.

3.2 Firmware del updown - Librerías de bajo nivel

3.2.1 DigitalIO

El desarrollo del módulo de entradas y salidas digitales (DigitalIO) está basado en el capítulo 18 de la hoja de datos del Atmega328p.

FIJATE SI NO ES MEJOR EMPEZAR POR LO DE QUE EL ACCESO SE HACE CON 3 FUNCIONES, Y DE AHI DECIR QUÉ HACE CADA FUNCIÓN

3.2.1.1 Uso

En microcontroladores de la marca Atmel, como es el de este proyecto, las entradas y salidas digitales manejan mediante 3 registros: el DDRn (Data Direction Register n), PORTn (Port n Data Register) y PINn (Port n Input Pins Address). La "n" en los registros se refiere al registro específico a ser accedido, que en el caso de este microcontrolador puede ser B, C o D.

Para configurar un pin, que es una pata del microcontrolador, como entrada o salida digital primero se elige la dirección del mismo, o sea, si se utilizará como entrada O como salida digital. Para esto se utiliza el registro DDRn, en donde escribir un bit de este registro en 1 significa configurar el pin asociado a este bit como salida o Output, mientras que si se escribe en 0 significa configurar al pin como entrada o Input.

Si el pin fue configurado como salida se setea su valor mediante el registro PORTn. Un 1 en un bit de este registro significa poner en HIGH (5 Volts) la salida asociada a ese bit, mientras que un 0 es un estado LOW (0 Volts).

Si el pin fue configurado como entrada se lee su valor del registro PINn, que puede ser 0

o 1 (HIGH o LOW). A un pin configurado como entrada se le puede, además, habilitar una resistencia pull-up interna del microcontrolador, haciendo que por defecto el canal esté en 1. El pull-up se encuentra deshabilitado por defecto, pero puede ser habilitado escribiendo un 1 en el bit análogo del registro PORTn.

Por ejemplo, si se quiere configurar el bit 3 del puerto B como entrada con pull-up y el bit 6 del puerto D como entrada en lenguaje C, se tiene:

```

1  /* —— Configuracion bit 3 puerto B como Input — Pullup — */
2  DDRB &= ~(1 << 3); // Configuracion como entrada
3  PORTB |= (1 << 3); // Habilitacion pullup
4
5  valorBit3PuertoB = PINB & (1 << 3); // Ejemplo lectura del bit
6
7  /* —— Configuracion bit 6 puerto D como Output — */
8  DDRD |= (1 << 6); // Configuracion como salida
9
10 PORTD |= (1 << 6); // Ejemplo escritura de un 1 en el bit

```

3.2.1.2 Implementación

El acceso al módulo se realiza a través de 3 funciones: una de inicialización (init), una de lectura (write) y otra de escritura (read), presentadas en la figura 3.1.

Para facilitar la lecto-escritura y configuración de los pines se mapearon los bits de los puertos B,C y D a números, como se muestra en la tabla 3.1. El criterio de enumeración fue basado en los pines del Arduino UNO.

Puerto	Bit	Numero mapeado
D	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
B	0, 1, 2, 3, 4, 5	8, 9, 10, 11, 12, 13
C	0, 1, 2, 3, 4, 5	14, 15, 16, 17, 18, 19

Tabla 3.1: Mapeo de bits de los puertos a número

3.2.2 ADC

El desarrollo del módulo ADC está basado en el capítulo 28 de la hoja de datos del Atmega328p.



Figura 3.1: Diagrama del módulo DigitalIO

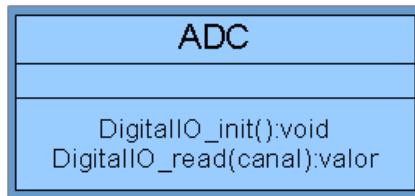


Figura 3.2: Diagrama del módulo ADC

3.2.2.1 Uso

3.2.2.2 Implementación

3.2.3 PWM

El desarrollo del módulo PWM está basado en el capítulo 20 de la hoja de datos del Atmega328p.

3.2.3.1 Uso

3.2.3.2 Implementación

3.2.4 EXINT

El desarrollo del módulo de interrupciones externas (EXINT) está basado en el capítulo 17 y 18 de la hoja de datos del Atmega328p.

3.2.4.1 Uso

3.2.4.2 Implementación

3.2.5 UART

El desarrollo del módulo UART está basado en el capítulo 24 de la hoja de datos del Atmega328p.

3.2.5.1 Uso

3.2.5.2 Implementación

3.2.6 SUART

El desarrollo del módulo SUART está basado en el capítulo 19 de la hoja de datos del Atmega328p.

3.2.7 Tick

El desarrollo del módulo Tick está basado en el capítulo 22 de la hoja de datos del Atmega328p.

3.2.7.1 Uso

3.2.7.2 Implementación

3.3 Controlador

3.3.1 Relación entre cuentas de encoder y distancia

Como se mencionó en la sección [2.3](#), subsección 3,

3.3.2 Determinación de la velocidad máxima

3.3.3 Obtención del período de muestreo

Como se mencionó en la sección [2.3](#), subsección 3,

3.3.4 Obtención del modelo de la planta

3.3.5 Obtención del controlador

3.3.6 Ajuste del controlador

3.4 Dipswitch

3.5 Firmware del updown - Librerías de alto nivel

3.6 Firmware del updown - Función principal

Implementación

4.1 Descripción del capítulo

En este capítulo se pondrá a prueba todo lo desarrollado en el capítulo anterior. Para esto se describirán los pasos que se hicieron para hacer las pruebas y cómo se validaron.

4.2 Configuración de la HOG

4.3 Validación

Conclusiones

5.1 c1