ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE UN PROTOTIPO DIDÁCTICO DE PUENTE GRÚA DE POSICIONAMIENTO MANUAL Y AUTOMÁTICO CON PINZA DE SUJECIÓN PARA EL IZAJE DE OBJETOS, CONTROLADO INALÁMBRICAMENTE DESDE UNA TABLET MEDIANTE BLUETOOTH UTILIZANDO EL MICROCONTROLADOR

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTROMECÁNICA

AMENDAÑO CLABÓN LUIS RAMÓN
luis.amendano@est.epn.edu.ec

SALCEDO BURGOS ESTEFANÍA CAROLINA
estefania.salcedo@est.epn.edu.ec

DIRECTOR: ING. ALFREDO ARCOS
alfredo.arcos@epn.edu.ec

QUITO, Julio, 2015

DECLARACIÓN

Nosotros, Estefanía Carolina Salcedo Burgos y Luis Ramón Amendaño Clabón, declaramos bajo juramento que el trabajo aquí escrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Estefanía Carolina Salcedo BurgosLuis Ramón Amendaño Clabón

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por **Estefanía Carolina Salcedo Burgos** y **Luis Ramón Amendaño Clabón**, bajo mi supervisión.

_____

Ing. Alfredo Arcos Lara

DIRECTOR DE PROYECTO

AGRADECIMIENTO

Queremos expresar nuestros más profundos y sinceros agradecimientos a Dios, por ser nuestra fortaleza y darnos cada día una razón más para vivir.

A todos los profesores y personal administrativo de la Escuela de Formación de Tecnólogos, quienes durante todos estos años supieron brindarnos su amistad y sabios conocimientos, de manera especial al Ing. Alfredo Arcos, por su acertada orientación en este proyecto.

Al Ing. Marco Quizanga (†), que aunque ya no se encuentre entre nosotros, lo llevaremos siempre en nuestros corazones porque supo ser un pilar fundamental en nuestra formación y un ejemplo de vida a seguir.

A nuestros amigos, compañeros y todos aquellos con quienes entablamos un fuerte lazo de amistad, en especial a Paúl, Andrés, Richi, Edison (Master), quienes nos brindaron su amistad y apoyo incondicional en los buenos y malos momentos, nos ayudaron a levantar e hicieron ver nuestros errores. No dejaremos de lado a aquellos amigos que decidieron tomar otro rumbo, aquellos que cayeron, se levantaron y aún se mantienen en pie de lucha. Sin lugar a duda nuestra segunda familia.

DEDICATORIA

Este trabajo lo dedico a mis padres, César Salcedo y Martha Burgos, porque gracias a su esfuerzo, sacrificio y apoyo supieron alentarme a seguir adelante y culminar este gran paso en mi vida.

A mis queridos tíos, Mamita Lolita (†) y Papito Marco (†) por haber sido mis segundos padres. Gracias por todo el cariño y amor que me entregaron, todos sus recuerdos los llevo en mi corazón con mucha nostalgia. Gracias a eso y a todas las enseñanzas que desde pequeña me han inculcado, viviré cada día siendo una mejor persona e hija para mis padres.

A todos mis familiares, compañeros y amigos, gracias por su valiosa amistad y por sus locuras. ¡Los quiero mucho!

Nunca es tarde para volver a empezar…

Estefanía S.

DEDICATORIA

El presente trabajo va dedicado a la persona que me supo acompañar durante todos estos años de estudio en la Universidad y al desarrollo de este proyecto. Nada de esto hubiera sido posible sin tu apoyo y sin aquellas palabras de aliento que me ayudaron a levantar cuando creía ver todo perdido. Por todo ello y más te agradezco infinitamente de corazón y te quedaré eternamente agradecido Carito.

A mi madre y hermanas por haber soportado todos estos momentos de ausencia en la casa. Nada en el mundo podrá compensar todo su sacrificio y la entrega que me han dedicado durante todos estos años.

Luis A

INDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

PRESENTACIÓN

Los sistemas de elevación y transporte de carga convencionales son de vital importancia en la industria pese a que emplean ciertas formas de control que impiden acelerar su funcionamiento en cualquier proceso. Para el desarrollo de este proyecto se ha optado por usar otra forma de control que se acople a la nueva tecnología y así salir de lo convencional, esto es utilizando la tecnología Bluetooth.

El presente proyecto se desarrolla en 5 capítulos resumidos a continuación:

Capítulo Uno: En este capítulo se da una breve descripción conceptual de los principales componentes eléctricos y mecánicos que forman parte del prototipo como son el microcontrolador que es la parte central del proyecto, los motores PAP y el servomotor, los drivers de control para los motores PAP, los sistemas de transmisión encargados de transmitir el movimiento, el módulo Bluetooth y la Tablet que comprenden el control inalámbrico del prototipo y desde la cual se envian las órdenes de control.

Capítulo Dos: En este capítulo, se describe brevemente la construcción de la estructura del prototipo y el montaje de los elementos en la misma. Del mismo modo se hace una descripción más específica de los elementos a ser usados en la implementación del hardware.

Capítulo Tres: En este capítulo, se detalla el desarrollo de la programación tanto para el control de la secuencia de funcionamiento del prototipo como para la interfaz gráfica de la Tablet que permite el control inalámbrico del puente grúa. Se incluye además la lógica de funcionamiento que debe cumplir en condiciones de posicionamiento manual y automático.

Capítulo Cuatro: En este capítulo se presentan las pruebas realizadas y los resultados obtenidos para comprobar su correcto funcionamiento y constatar el cumplimiento del objetivo principal de este proyecto.

Capítulo Cinco: En este capítulo se exponen las principales conclusiones del proyecto y se dan algunas recomendaciones al respecto.

RESUMEN

En varios procesos de producción que se realizan en talleres, bodegas, almacenes y demás plantas industriales; la utilización de puentes grúas (o sistemas de izaje) resultan de gran apoyo para reducir el tiempo de almacenamiento, distribución y traslación de cargas de gran peso a diferentes lugares. El funcionamiento de los puentes grúa convencionales, consta de varios procesos donde generalmente se emplea un control remoto básico que se limita a posicionar el gancho de sujeción de objetos únicamente de forma manual, impidiendo acelerar los procesos.

El presente proyecto pretende mejorar el elemento de control adaptándolo a la tecnología actual. Para tal fin se emplea un dispositivo móvil inteligente (Tablet) para la que se ha desarrollado una aplicación a modo de interfaz gráfica y desde la cual, se puede controlar el puente grúa. La comunicación entre dispositivos se establece de forma inalámbrica, empleando la tecnología Bluetooth. La característica principal de este nuevo dispositivo es que permite posicionar la pinza de sujeción de forma automática al lugar deseado.

A esto se suma la utilización de una pinza que reemplaza al gancho de sujeción de piezas convencional, la misma que; mediante un sensor infrarrojo, realiza el proceso de apertura y cierre.

CAPÍTULO 1

MARCO TEÓRICO

1.1 EL MICROCONTROLADOR

Un microcontrolador es un circuito integrado programable que contiene en su interior los componentes necesarios para controlar el funcionamiento de una tarea determinada, estos son: la unidad de procesamiento central (CPU), memoria de programa, memoria de datos y puertos de entrada y salida.

El funcionamiento de los microcontroladores está determinado por el programa almacenado en su memoria el cual puede escribirse en distintos lenguajes de programación. Una vez programado y configurado el microcontrolador, solamente sirve para controlar la tarea asignada. [1]

Sus líneas de entrada/salida soportan la conexión de conversores de analógico/ digital, temporizadores, sensores y dispositivos de control que permitan efectuar el proceso deseado.

Los componentes de los que dispone normalmente un microcontrolador son:

- Procesador o CPU (unidad central de procesamiento).
- Memoria RAM para contener los datos de propósito general.
- Memoria tipo ROM/PROM/EPROM/EEPROM destinada para el programa.
- Líneas de entrada/salida para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, puertos serie y paralelo, CAD, etc.).
- Señal de reloj que sincroniza el funcionamiento de todo el sistema. [2]

En la figura 1.1 se puede observar cómo está estructurado internamente un microcontrolador.

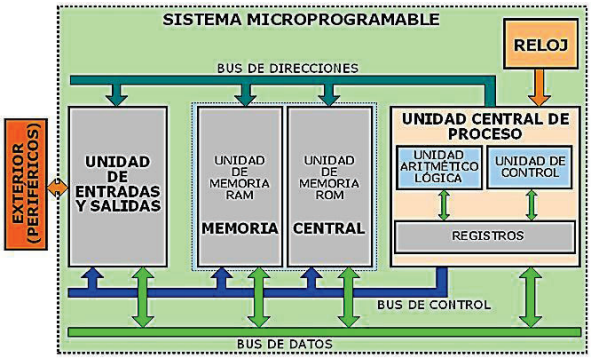


Figura 1.1 – Arquitectura interna de un microcontrolador [3]

1.1.1 EL MICROPROCESADOR

Un microprocesador es básicamente un chip que contiene la CPU (*Central Proccesing Unit*) que se encarga de controlar todo el sistema. Un sistema digital basado en un microprocesador es un sistema abierto ya que su configuración difiere según la aplicación a la que se destine. Se pueden acoplar los módulos necesarios para configurarlo con las características que se desee. Para ello, saca al exterior las líneas de sus buses de datos, direcciones y control de modo que permita su conexión con la memoria y los módulos de entrada/salida. Finalmente, resulta un sistema implementado por varios circuitos integrados dentro de una misma placa de circuito impreso.

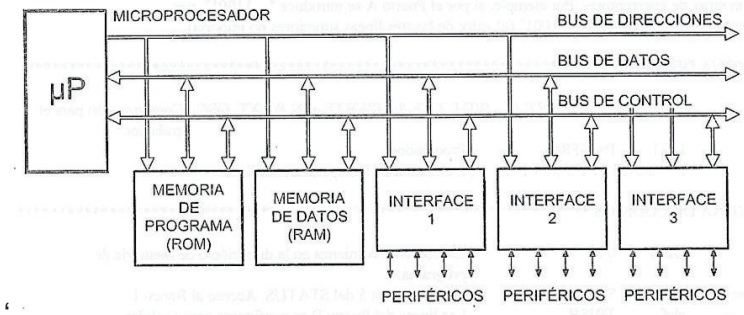


Figura 1.2 – Estructura de un sistema digital basado en microprocesador [1]

1.1.2 EL MICROCONTROLADOR VS EL MICROPROCESADOR

Un microcontrolador es un sistema cerrado, lo que quiere decir que en un solo circuto integrado se encierra un sistema digital programable completo. Este

dispositivo se destina a gobernar una sola tarea que no se puede modificar. Un microprocesador es básicamente un chip que contiene la CPU que se encarga de controlar todo el sistema.

La diferencia fundamental entre ambos es que un sistema digital basado en un microcontrolador está formado por un solo circuito integrado lo que reduce notablemente el tamaño y el coste, mientras que un sistema basado en un microprocesador, al estar compuesto por varios circuitos integrados para soportar las memorias y los módulos de entrada/salida, tiene mayor tamaño, más coste y menor fiabilidad. [1]

1.1.3 ARQUITECTURA HARVARD [1]

Los microcontroladores actuales utilizan la arquitectura Harvard que dispone de dos memorias independientes a las que se conectan mediante dos grupos de buses separados como son la memoria de datos y la memoria de programa. Ambos buses son totalmente independientes y pueden ser de distintos anchos, esto permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones, consiguiendo que las instrucciones se ejecuten en menos ciclos de reloj. En la fig. 1.3 se puede apreciar de forma esquemática cómo está constituida la arquitectura Harvard.

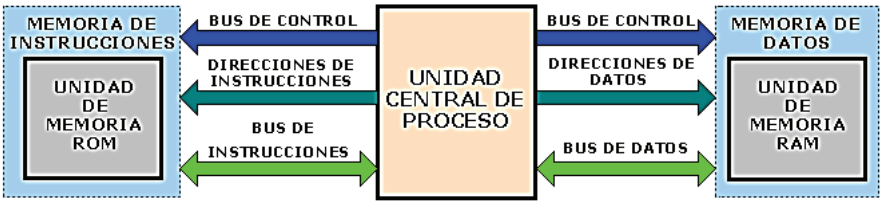


Figura 1.3 – Arquitectura Harvard [3]

La dualidad de la memoria de datos por un lado y por otro la memoria de programa, permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y los datos.

Las principales ventajas de la arquitectura Harvard son:

- El tamaño de las instrucciones no está relacionado con el de los datos y, por lo tanto, puede ser optimizado para que cualquier instrucción ocupe

una sola posición de memoria de programa. Logrando mayor velocidad y una menor longitud del programa.

- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

1.1.4 PROCESADORES RISC [1]

Un procesador RISC (Reduced Instruction Set Computer) es un microprocesador con un repertorio de instrucciones reducido. Las instrucciones son muy simples y suelen ejecutarse en un ciclo de máquina. Los procesadores RISC suelen tener una estructura Pipeline (procesador segmentado), que le permite realizar simultáneamente la ejecución de una instrucción y la búsqueda de código de la siguiente, de esta manera; posibilita el ejecutar una instrucción en un ciclo de máquina. Es decir, ejecuta casi todas las instrucciones en el mismo tiempo.
Una arquitectura Harvard acompañada de un procesador RISC produce que el set de instrucciones y el bus de la memoria de programa puedan diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar. Añadiendo a las anteriores, la arquitectura RISC también posee las siguientes características:

- Load/Store: Operaciones registro a registro, separadas de los accesos a memoria.
- Repertorio de instrucciones cuidadosamente seleccionados, optimizados e implementados en memoria
- Instrucciones en formato fijo.
- Modos de direccionamiento simple

1.1.5 MICROCONTROLADORES ARM [4] [5] [6]

ARM (Advanced RISC Machine) es una arquitectura RISC de 32 bits desarrollada por ARM Holdings. La arquitectura ARM se diseñó para permitir

implementaciones de tamaño muy reducido y de alto rendimiento. Como lo indica su nombre, se caracteriza fundamentalmente por ser una computadora de set de instrucciones reducido, y debido a su simplicidad; es ideal para aplicaciones de baja potencia. La arquitectura ARM incorporó algunas características del diseño RISC, como son: arquitectura de carga y almacenamiento (load-store), instrucciones de longitud fija de 32 bits y formatos de instrucción de 3 direcciones.
Actualmente, la arquitectura ARM se ha convertido en dominante en el mercado de la electrónica móvil e integrada, encarnada en microprocesadores y microcontroladores pequeños, de bajo consumo y relativamente bajo coste.

1.1.5.1 **Microcontrolador ARM Cortex-M4 [7]**

Es un microcontrolador que pertenece a la arquitectura ARM7 de última generación y de los más poderosos dentro de la gama de núcleos escalares con un set de instrucciones ortogonal que le proporcionan gran flexibilidad y eficiencia para ejecutar un código en C. Las prestaciones de este microcontrolador permite la ejecución de sistemas operativos como: Android, iOS, Kernel de Linux, etc.
A principios del 2012, ST-Microelectronics ha desarrollado kits de evaluación para microcontroladores ARM de 32 bits, con el fin de dar a conocer las distintas familias de microcontroladores de dicha marca. ST posee varios modelos de estas placas con distintas familias de microcontroladores ARM, que van desde los CORTEX-M0 (familia STM32F0) hasta los CORTEX-M4 (familia STM32F4). Además de ST, existen otras empresas que producen este tipo de placas, como es LPCXpresso de NXP y las lauchpad de TI entre otras.

1.2 MOTORES ELÉCTRICOS

Son máquinas eléctricas rotatorias compuestas por un estator y un rotor o a su vez es un dispositivo electromecánico que transforma la energía eléctrica en energía mecánica por medio de la acción de los campos magnéticos generados en sus bobinas.
Las aplicaciones que tienen los motores en el área de la automatización son muy amplias y van desde los juguetes hasta la robótica industrial, pasando por la

medicina, aplicaciones militares, computadoras, dispositivos de entretenimiento, simuladores, máquinas-herramientas, etc.

1.2.1 MOTOR DE CORRIENTE CONTINUA

Un motor de corriente continua o simplemente motor DC está compuesto internamente por un estator que crea un campo magnético en el cual se encuentra una bobina o electroimán arrollada en un eje giratorio (rotor). La tensión de alimentación aplicada al motor hace que se generen fuerzas de atracción y repulsión o una interacción de campos entre el estator y el rotor, lo que hace que el motor se mantenga en movimiento.

La velocidad de un motor DC se expresa en revoluciones por minuto (rpm) y es función de la tensión aplicada, corriente por el motor y carga mecánica del mismo. Un posicionamiento preciso del eje de un motor DC no es posible por métodos sencillos. [1]

1.2.2 MOTORES PASO A PASO

Los ***motores paso a paso*** o PAP (*Stepper Motor*) son muy utilizados en los dispositivos controlados por sistemas digitales, en situaciones en que se requiere un control preciso de la trayectoria a seguir por el eje del motor.

Los motores PAP proporcionan una considerable ventaja sobre los motores de corriente continua o DC. El eje de un motor PAP gira a intervalos regulares en lugar de hacerlo continuamente, como ocurre con los motores DC.

Un motor PAP gira en función de una secuencia de pulsos aplicados a sus devanados. El eje del motor gira un determinado ángulo por cada pulso de entrada. Cada pulso provoca la rotación del motor en un incremento de ángulo preciso, denominado ***paso,*** de ahí el nombre de motor “paso a paso”. El resultado de este movimiento, fijo y repetible, es un posicionamiento preciso y fiable. Los incrementos de pasos de la rotación del rotor se traducen en un alto grado de control de posicionamiento.

Los incrementos de rotación o pasos se miden en grados y es el parámetro fundamental de un motor PAP. También se puede expresar en números de pasos

por revolución de 360°. Un motor PAP puede girar un número exacto de grados en ambos sentidos.

1.2.2.1 **Principio de funcionamiento**

Aún basados en el mismo fenómeno que el principio de funcionamiento de los motores de corriente continua, los motores PAP son aún más sencillos.
La figura 1.4 ilustra el modo de funcionamiento de un motor PAP.

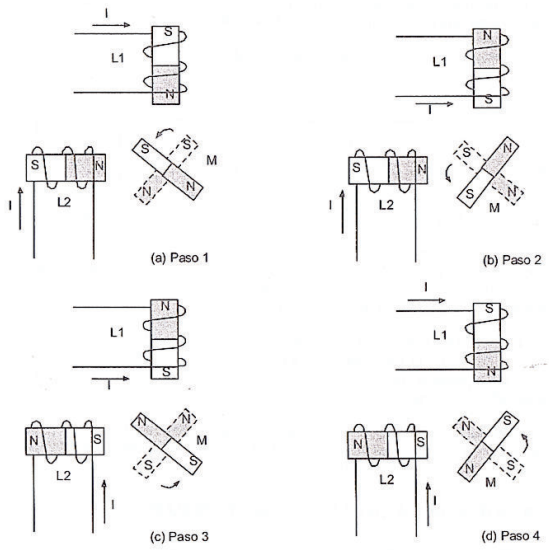


Figura 1.4 – Principio de funcionamiento de un motor paso a paso [1]

Suponemos que tanto las bobinas L1 como L2 poseen un núcleo de hierro dulce capaz de imantarse cuando dichas bobinas sean recorridas por una corriente eléctrica, esto es el denominado **estator**. Por otra parte el imán M puede girar libremente sobre el eje de sujeción central, éste es el **rotor**.
Inicialmente, sin aplicar ninguna corriente a las bobinas (que también reciben el nombre de fases) y con el imán M en una posición cualquiera, el imán permanece en reposo si no se somete a una fuerza externa.
Si se hace circular corriente por ambas fases como se muestra en la figura 1.4a, se crean dos polos magnéticos NORTE en la parte interna. Bajo esta influencia el imán M se desplaza hasta la posición indicada en la figura 1.4a.
Si invertimos la polaridad de la corriente que circula por la bobina L1, se obtiene la situación magnética indicada en la figura 1.4b, y el imán M, se desplaza hasta la nueva posición de equilibrio, es decir, gira 90° en sentido anti horario.

Invirtiendo ahora la polaridad de la corriente que atraviesa la bobina L2, se llega a la situación de la figura 1.4c, habiendo girado el imán M otros 90°. Si nuevamente invertimos el sentido de la corriente en la bobina L1, el imán M gira otros 90°, obteniéndose una revolución completa de dicho imán en cuatro pasos de 90°.

Si se mantiene la secuencia de excitación expuesta para L1 y L2 y dichas corrientes son aplicadas en forma de pulsos, el rotor avanza pasos de 90° por cada pulso aplicado.

Analizando lo expuesto se puede decir que un motor PAP, es un dispositivo electromecánico que convierte pulsos eléctricos en un movimiento rotacional, constante y finito, dependiendo de las características propias del motor.

1.2.2.2 Tipos de motores PAP

Los motores PAP se dividen en dos categorías principales:

- Motores PAP de imán permanente a veces llamados “de rotor activo”. Estos a su vez se clasifican en:
  - Motores PAP bipolares
  - Motores PAP unipolares
- Motores PAP de reluctancia variable.

También existe una combinación de ambos, a los que se les llama híbridos.

La figura 1.5 muestra una breve clasificación de los motores PAP.

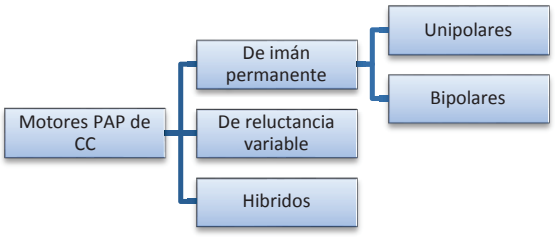


Figura 1.5 – Clasificacion de los motores PAP

De la clasificación de los motores PAP, nos concentraremos únicamente en el estudio de los motores PAP de imán permanente, pues son los más conocidos comercialmente.

1.2.2.2.1 Motores PAP bipolares

En este tipo de motores las bobinas del estator se conectan formando dos grupos, tal y como se muestra en la figura 1.6. Recibe el nombre de bipolar ya que para obtener la secuencia completa, se requiere disponer de corrientes de dos polaridades, presentando tal circunstancia un inconveniente importante a la hora de diseñar el circuito que controle el motor.

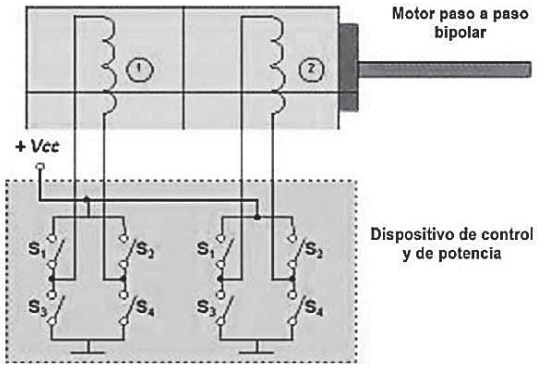


Figura 1.6 – Circuito de control de un motor bipolar [1]

De este motor, salen cuatro hilos que se conectan al circuito de control, que realiza la función de cuatro interruptores electrónicos dobles, permitiendo variar la polaridad de la alimentación de las bobinas. Con la activación y desactivación ordenada de dichos interruptores dobles, podemos obtener las secuencias adecuadas para que el motor pueda girar en un sentido o en otro.

1.2.2.2.2 Motores PAP unipolares

En los motores PAP unipolares, todas las bobinas del estator están conectadas en serie formando 4 grupos. Estos a su vez, se conectan dos a dos y se montan sobre dos estatores diferentes.

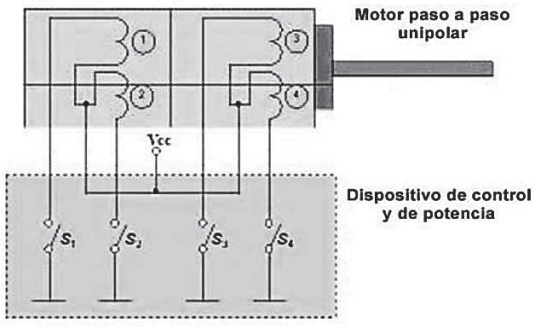


Figura 1.7 – Circuito de control de un motor unipolar [1]

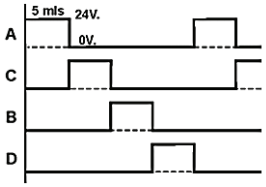
La figura 1.7 ilustra un motor PAP unipolar de cuatro fases (o bobinas), donde la corriente circula por las bobinas en un único sentido; paliando así, el inconveniente que supone la necesidad de dos polaridades de la corriente para generar la secuencia del motor.

Como se observa, del motor PAP salen dos grupos de 3 cables, uno de los cuales es común a dos bobinados. Los seis terminales que parten del motor se conectan al circuito de control, el cual, se comporta como cuatro conmutadores electrónicos que al ser activados o desactivados, producen la alimentación de los cuatro grupos de bobinas con que está formado el estator. Generando una secuencia adecuada de funcionamiento de estos interruptores, se pueden producir saltos de un paso en el número y sentido deseado.

1.2.2.3 Secuencias de manejo de un motor PAP

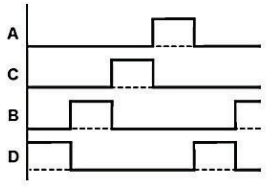
1.2.2.3.1 Manejo de un motor PAP en secuencia Wave Drive

La secuencia Wave Drive o secuencia por ola, consiste en energizar una sola bobina a la vez tal como se indica en la figura 1.8. Esto brinda un funcionamiento más suave en algunos motores. La desventaja es que al estar solo una bobina activada, el torque de paso y retención es menor



Bobina	P 1	P 2	P 3	P 4
A	1	0	0	0
C	0	1	0	0
B	0	0	1	0
D	0	0	0	1

a) Secuenca de giro de sentido antihorario



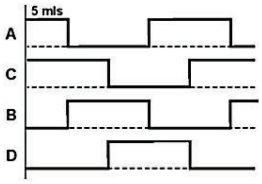
Bobina	P 1	P 2	P 3	P 4
A	0	0	0	1
C	0	0	1	0
B	0	1	0	0
D	1	0	0	0

b) Secuencia de giro de sentido horario

Figura 1.8 – Secuencia de energizado de bobinas en secuencia por ola [1]

1.2.2.3.2 Manejo de un motor PAP en secuencia Full Step

En el modo full step o secuencia por paso completo, el rotor del motor PAP avanza un paso por cada pulso de excitación obteniéndose un alto torque de paso y de retención. Pese a que consume un 40% más de corriente, que el caso anterior, es la forma de control que recomienda el fabricante. La figura 1.9 indica la secuencia en que se debe energizar las bobinas para poner en marcha el motor

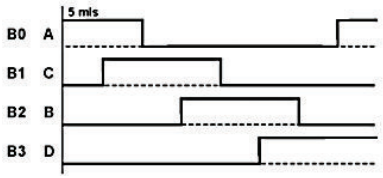


Bobina	P 1	P 2	P 3	P 4
A	1	0	0	1
C	1	1	0	0
B	0	1	1	0
D	0	0	1	1

Figura 1.9 – Secuencia de energizado de bobinas en secuencia full step [1]

1.2.2.3.3 Manejo de un motor PAP en secuencia Half Step

También conocida como secuencia a medio paso, es una combinación de las 2 secuencias anteriores. En este modo, el rotor avanza medio paso por cada pulso de excitación. Presenta como principal ventaja una mayor resolución de paso, ya que disminuye el avance angular (la mitad que en el modo de paso completo). Para conseguir tal cometido, el modo de excitación consiste en hacerlo alternativamente sobre 2 bobinas y sobre una sola de ellas, según se muestra en la figura 1.10. Por consiguiente, para girar una vuelta completa utilizando esta forma de control, se necesita el doble de pasos que en los casos anteriores.



Bobina	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8
A	1	1	0	0	0	0	0	1
C	0	1	1	1	0	0	0	0
B	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

Figura 1.10 – Secuencia de energizado de bobinas en secuencia half step [1]

1.2.2.4 Parámetros principales de los motores PAP

Desde el punto de vista mecánico y eléctrico, es conveniente conocer el significado de ciertas características y parámetros que definen un motor PAP:

- ***Par dinámico de trabajo*** (*Working Torque*). Depende de sus características dinámicas y es el momento máximo que el motor es capaz de desarrollar sin perder paso, es decir, sin dejar de responder a algún pulso de

excitación del estator y dependiendo, evidentemente, de la carga. Hay que tener en cuenta que, cuando la velocidad de giro del motor aumenta, se produce un aumento de la f.c.e.m (*fuerza contra-electromotriz*) en él generada y, por lo tanto, una disminución de la corriente absorbida por los bobinados del estator. Como consecuencia de ello, disminuye el par motor.

- ***Ángulo de paso*** (*Step Angle*). Se define como el avance angular que se produce en el motor por cada pulso de excitación. Se mide en grados, siendo los pasos estándar más importantes, los mostramos en la tabla 1.1.

Tabla 1.1 – Ángulos de pasos más comunes en los motores PAP

Grados por impulso de excitación	Nº de pasos por vuelta
0,72º	500
1,8º	200
3,75º	96
7,5º	48
15º	24

- ***Número de pasos por vuelta***. Es la cantidad de pasos que ha de efectuar el rotor para realizar una revolución completa. Su ecuación está dada por:

$$NP = \frac{360}{\alpha}$$

$NP = Número\ de\ pasos$

$\alpha = Resolución$

- ***Par de detención*** *(Detention Torque).* Es un par de freno, siendo propio de los motores de imán permanente. Este efecto se debe a la acción del rotor, cuando los devanados del estator están desactivados.

1.2.2.5 Control de los motores PAP

Para realizar el control de los motores PAP, es necesario como hemos visto, generar una secuencia determinada de pulsos. Además, es necesario que estos pulsos sean capaces de entregar la corriente necesaria para que las bobinas del motor se exciten.

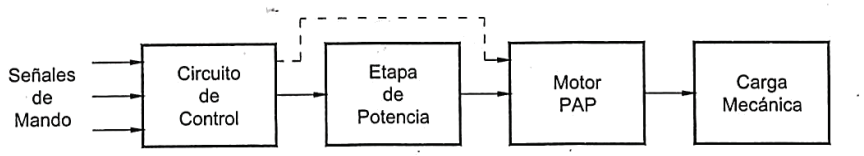


Figura 1.11 – Diagrama de bloques de un sistema de control para motor PAP [1]

La figura 1.11 muestra el diagrama de bloques general de un sistema de control para un motor PAP. Como el microcontrolador no es capaz de generar la suficiente corriente para excitar las bobinas del motor PAP es necesario hacer uso de drivers especializados para tal fin o a su vez acceder al uso de circuitos integrados que en su interior contienen los drivers necesarios para controlar el motor PAP. [8]

1.2.3 SERVOMOTORES

Un **servomecanismo** es un actuador mecánico, que posee los suficientes elementos de control para poder monitorear los parámetros de actuación mecánica, como posición, velocidad, torque, etc. Un **servomotor** es un motor eléctrico que puede ser controlado, tanto en velocidad como en posición.

1.2.3.1 Componentes del servomotor [9] [10]

Un servomotor está constituido por un pequeño motor DC, una tarjeta electrónica necesaria para el control y un sistema de engranajes utilizado como reductor de velocidad y que a la vez brinda una potencia considerable.

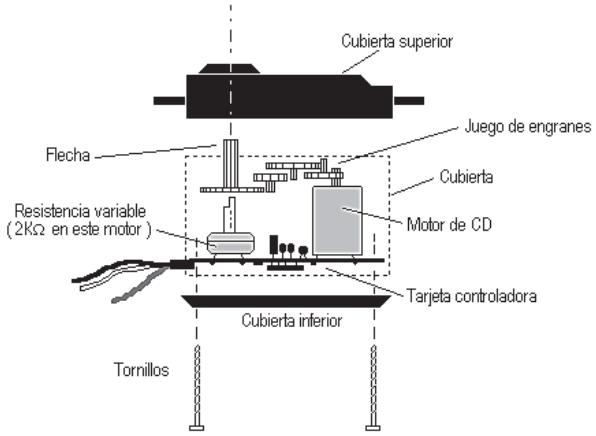


Figura 1.12 – Componentes de un servomotor [10]

En la figura 1.12, se pueden observar otros elementos que componen el servomotor como lo es el encoder (un detector que codifica la posición) conectado al eje de salida y que no es más que un potenciómetro que permite detectar y codificar la posición. Los servomotores disponen de 3 cables de conexión. Uno

es para la alimentación Vcc, otro para la conexión a tierra y un tercer cable se destina para el ingreso de la señal que permite el control del servomotor.

1.2.3.2 Funcionamiento del servomotor [1] [9]

La tensión de alimentación de los servomotores suele estar comprendida entre los 4.8V y 6V. El control de un servomotor se limita a indicar en qué posición se debe situar el eje, mediante una señal cuadrada TTL modulada en ancho de pulso PWM (*Pulse Width Modulation*), que se envía a través del cable de control del servomotor. La duración del nivel alto de la señal indica la posición donde queremos ubicar el eje del motor. El potenciómetro unido al eje indica al circuito electrónico de control interno mediante una retroalimentación, si este ha llegado a la posición deseada. La duración de los pulsos indica el ángulo de giro del motor, tal como se indica en la figura 1.13.

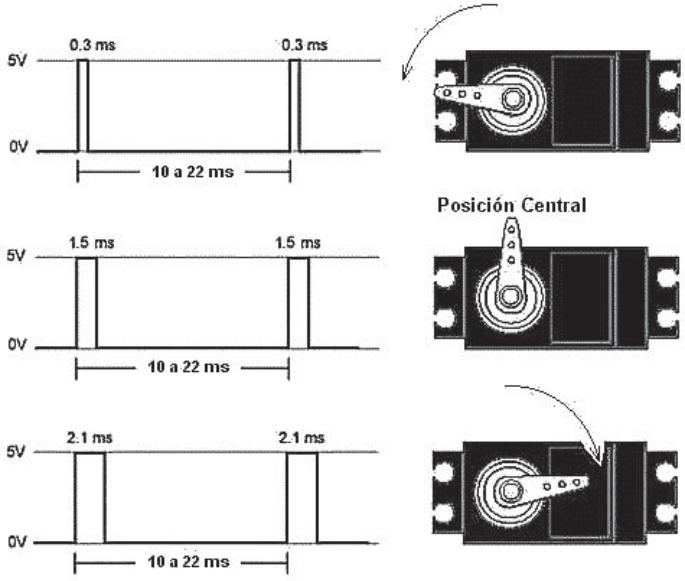


Figura 1.13 – Posición del eje del servomotor para diferentes trenes de pulsos [9]

La variación del ancho de pulso permite posicionar el servomotor en un determinado ángulo. El rango de variación del ancho de pulso está dado por el fabricante, aunque comúnmente suelen utilizarse ciertos valores estándar que facilitan el control, esto es un pulso de onda cuadrada de 1.5ms de ancho que se repite a un ritmo de entre 10ms a 22 ms. Mientras el pulso se mantenga en este ancho el servomotor se ubica en la posición central de su recorrido. Si el ancho

de pulso disminuye, el servomotor se mueve de manera proporcional hacia un lado. Si el ancho de pulso aumenta, el servomotor gira hacia el otro lado. Generalmente el rango de giro de un servomotor de éstos cubre entre 90° y 180° de la circunferencia total, o un poco más, según la marca y modelo. Si se sobrepasan los límites de movimiento del servomotor, éste comienza a vibrar o a emitir un zumbido, denunciando un cambio en el ancho del pulso.

El periodo entre pulso y pulso no es crítico. Se suelen emplear valores entre 10ms y 30ms aunque lo habitual es utilizar 20ms, que implica una frecuencia de 50Hz. Si el intervalo entre pulso y pulso es inferior al mínimo puede interferir con la temporización interna del servo causando un zumbido y la vibración del brazo de salida. Si es mayor que el máximo, entonces el servo pasa a un estado de reposo entre pulsos provocando que se mueva a pequeños intervalos.

Es importante destacar que para que un servomotor se mantenga en la misma posición, es necesario enviarle continuamente un pulso de ancho constante. De este modo si existe alguna fuerza que obligue a abandonar esta posición, intentará resistirse. Si se deja de enviar pulsos, o el intervalo entre pulsos es mayor del máximo permitido, entonces el servomotor pierde fuerza y deja de mantener su posición, de modo que cualquier fuerza externa podría desplazarlo.

1.3 LOS SENSORES

Los sensores son dispositivos capaces de convertir el valor de una magnitud física que se recibe del medio externo (temperatura, luz, sonido, etc.) en una señal eléctrica codificada de corriente o voltaje ya sea en forma analógica o digital la cual puede ser manipulada (medida, amplificada, transmitida, etc.).

1.3.1 CARACTERÍSTICAS DE LOS SENSORES [11] [12]

El sensor ideal sería aquel cuya relación entre magnitud de entrada y magnitud de salida fuese proporcional y de respuesta instantánea. Pero la realidad es otra, puesto que la respuesta real de los sensores nunca es del todo lineal, tiene un rango limitado de validez, suele estar afectada por perturbaciones del entorno

exterior y tiene un cierto retardo en la respuesta. Estas son las características estáticas y características dinámicas.

1.3.1.1 Características Estáticas

Describen la actuación del sensor en régimen permanente o con cambios muy lentos de la variable a medir. Entre las más importantes se destacan: rango de medida, resolución, precisión, sensibilidad y ruido, es decir, cualquier perturbación aleatoria del propio sistema de medida que afecta la señal que se quiere medir.

1.3.1.2 Características Dinámicas

Describen el comportamiento del sensor en régimen transitorio. Estas son: velocidad de respuesta y estabilidad.

1.3.2 SENSORES DE CONTACTO

Los sensores de contacto, también conocidos como fines de carrera son dispositivos situados al final del recorrido de un elemento móvil permitiendo detectar la posición límite de desplazamiento de un componente. Su accionamiento de tipo mecánico vincula a un conjunto de contactos en el interior del dispositivo que pueden abrirse o cerrarse dependiendo de la operación que cumplan al ser accionados. Cuando un objeto entra en contacto con el accionador, el dispositivo opera como un pulsador que dispone de un muelle de recuperación para abrir o cerrar una conexión eléctrica. La figura 1.14 ilustra la estructura interna básica de un fin de carrera.

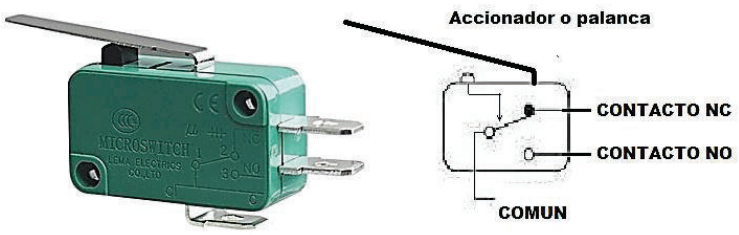


Figura 1.14 – Sensor de contacto y su constitucion interna básica

Como se puede observar, si se activa la palanca, el terminal común cambia de posición hacia el contacto normalmente abierto, abriendo de este modo el circuito. Si la palanca no se activa, el dispositivo se mantiene en el contacto normalmente cerrado, manteniendo activado cualquier circuito eléctrico al que se encuentre conectado. [13]

1.3.3 SENSOR DE INFRARROJOS

El sensor infrarrojo es un dispositivo electrónico capaz de reaccionar a la radiación electromagnética infrarroja emitida por los cuerpos en su campo de visión. Todos los cuerpos reflejan o emiten una cierta cantidad de radiación, esta resulta invisible para el ojo humano pero no para estos aparatos electrónicos, ya que se encuentran en el rango del espectro justo por debajo de la luz visible.
Este tipo de sensor puede ser del tipo pasivo o activo siendo este último el que se basa en la combinación de un emisor y un receptor próximos entre ellos, normalmente forman parte de un mismo circuito. El emisor puede ser un LEDIR (Diodo Emisor de Luz Infrarroja) y el componente receptor el fototransistor.

1.3.4 DIODOS EMISORES DE LUZ INFRARROJA (LED IR)

Ciertos diodos LED emiten fotones con longitudes de onda más larga que no forman parte del espectro visible y están localizados en la parte infrarroja (IR) del espectro. Los diodos emisores de luz elaborados en materiales semiconductores como Arsénido de Galio (GaAs), emiten radiación infrarroja, que no es visible al ojo humano. Por las características que prestan este tipo de diodos se lo denomina diodos emisores de luz infrarroja (LED IR). Su apariencia y conexión es la misma que de un LED convencional, generalmente son de color azul. Tal como se indica en la fig. 1.15 [14]

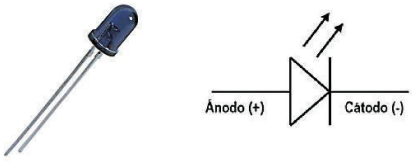


Figura 1.15 – Apariencia de un diodo emisor de luz infrarroja

1.3.5 EL FOTOTRANSISTOR

Este dispositivo se diferencia de un transistor común porque presenta una unión PN colector – base fotosensible a través de un lente en el encapsulado del transistor y por el cual se expone a la luz incidente, es decir su base ha sido sustituida por un cristal fotosensible el cual regula el flujo de corriente de colector de acuerdo a la intensidad de luz que incida sobre él. Cuando no hay luz incidente, existe una pequeña corriente de fuga colector – emisor generada térmicamente y conocida como corriente oscura la que suele estar en el rango de los nA. Si incide luz sobre el lente, entonces se genera una corriente de base que es proporcional a la cantidad de luz que incida sobre él.

Un fototransistor puede ser un dispositivo con 2 ó 3 terminales. Para la configuración de 3 puntas, el conector de la base se saca al exterior de modo que el dispositivo pueda ser usado como un TBJ convencional con o sin la característica de fotosensibilidad. En la configuración con 2 terminales se elimina la conexión eléctrica a la base de modo que el dispositivo puede ser usado únicamente con luz como entrada. Su apariencia es muy similar a la de un diodo convencional tal como se ilustra en la figura 1.16. [14]

Figura 1.16 – Apariencia y símbolo de un fototransistor

1.4 ACTUADOR FINAL [15]

En aplicaciones industriales, las capacidades de un robot básico se amplían por medio de dispositivos adicionales denominados periféricos. El actuador final representa la herramienta especial que permite al robot de uso general realizar una aplicación particular, por lo que se diseña específicamente para dicha aplicación. Los actuadores finales que generalmente se emplean, se dividen en: herramientas y pinza, haciendo un breve hincapié únicamente en este último.

1.4.1 PINZAS (GRIPPER)

Se suele denominar pinza a los elementos de sujeción que se emplean para agarrar y sostener objetos. Se utilizan para tomar un objeto y sujetarlo durante el ciclo de trabajo. De entre la gran variedad de pinzas se distinguen las que utilizan dispositivos de agarre mecánico y las que utilizan algún otro tipo de dispositivo (ventosas, pinzas magnéticas, adhesivas, ganchos, etc.). Hay una diversidad de métodos de sujeción que pueden utilizarse, además de los métodos mecánicos obvios de agarre de la pieza entre dos o más dedos. Estos métodos suplementarios incluyen el empleo de casquetes de sujeción, imanes, ganchos, entre otros. Se pueden clasificar según el sistema de sujeción empleado, como se puede apreciar en la tabla 1.2.

Tabla 1.2 – Clasificación de los sistemas de sujeción [15]

SISTEMAS DE SUJECIÓN		
TIPO	ACCIONAMIENTO	USO
PINZA DE PRESIÓN • DESP. ANGULAR • DESP. LINEAL	NEUMÁTICO O ELÉCTRICO	Transporte y manipulación de piezas sobre las que no importe presionar
PINZA DE ENGANCHE	NEUMÁTICO O ELÉCTRICO	Piezas de grandes dimensiones o sobre las que no se pueden ejercer presión
VENTOSA DE VACÍO	NEUMÁTICO	Cuerpos con superficie lisa poco porosa (cristal, plástico etc.)
ELECTROIMÁN	ELÉCTRICO	Piezas ferromagnéticas

El accionamiento neumático es el más empleado por ofrecer mayores ventajas en simplicidad, precio y fiabilidad, aunque presenta dificultades de control de posiciones intermedias. En ocasiones se utilizan accionamientos de tipo eléctrico. En la pinza se suelen situar sensores para detectar el estado de la misma (abierto o cerrado). Se pueden incorporar a la pinza otro tipo de sensores para controlar el estado de la pieza, sistemas de visión que incorporen datos geométricos de los objetos, detectores de proximidad, sensores fuerza par, etc.

Los tipos de pinzas más comunes pertenecen al tipo llamado **pivotante**. Los dedos de la pinza giran en relación con los puntos fijos del pivote. De esta manera, la pinza se abre y se cierra. Otro tipo de pinzas se denominan de

movimiento lineal. En este caso, los dedos se abren y cierran ejecutando un movimiento paralelo entre sí. Ambos casos se diferencian en la figura 1.17.

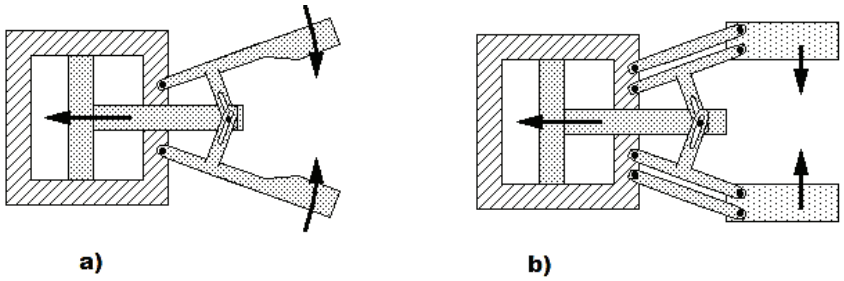


Figura 1.17 – Tipo de pinzas: a) Pinza de tipo pivotante; b) Pinza de movimiento lineal

En la elección o diseño de una pinza se debe tener en cuenta diversos factores. Entre los que afectan al tipo de objeto y de manipulación a realizar destacan el peso, la forma, el tamaño del objeto y la fuerza necesaria a ejercer y mantener para sujetarlo. Entre los parámetros de la pinza cabe destacar su peso (que afecta a la inercia del sistema), el equipo de accionamiento y la capacidad de control.

1.5 LOS MECANISMOS DE TRANSMISIÓN DE MOVIMIENTO [16]

Generalmente, cuando se habla de máquinas es necesario hacerlo también de motores, pues en la mayoría de los casos toda máquina lleva incorporada un motor que transforma, mediante un mecanismo de transmisión, la energía rotatoria de su eje en trabajo mecánico.

Un **mecanismo** es un dispositivo que transforma el movimiento producido por un elemento motriz en un movimiento similar o diferente a la salida. La trasformación de la fuerza y el movimiento producido generalmente por un motor, suele realizarse mediante cadenas cinemáticas, que son sistemas de elementos mecánicos conectados convenientemente entre sí para transmitir potencia mecánica del elemento motriz a la carga propiamente dicha.

Estos elementos mecánicos, a su vez, suelen ir montados sobre los llamados ejes de transmisión, que son piezas cilíndricas sobre las cuales se colocan los mecanismos de transmisión correspondientes y que son los encargados de transmitir el movimiento de una parte a otra del sistema.

Entre los mecanismos de transmisión más importantes empleados en la transmisión de potencia mecánica a través de cadenas cinemáticas, podemos destacar: sistemas de poleas y correas, sistemas de ruedas de fricción, sistemas de engranajes, sistemas de ruedas dentadas y cadenas, sistemas de tornillo sinfín y rueda helicoidal, sistemas de rueda dentada y cremallera también conocido como sistema de piñón y cremallera, etc.

1.5.1 SISTEMA DE TRANSMISIÓN DE POLEAS Y CORREAS

Se emplea para transmitir la potencia mecánica proporcionada por el eje del motor entre dos ejes separados entre sí por una cierta distancia.

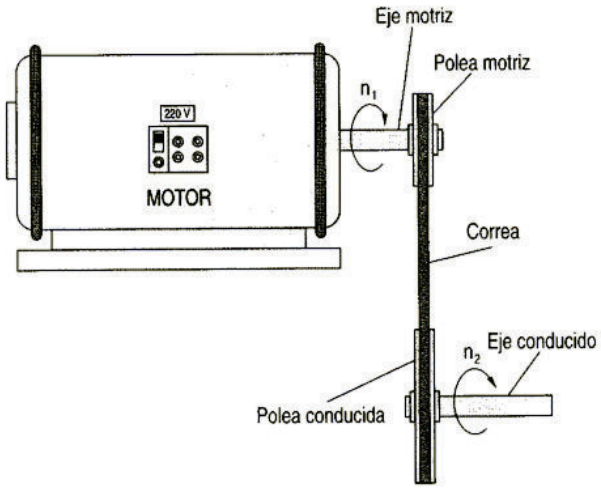


Figura 1.18 – Representación del sistema de transmisión de poleas y correas

La transmisión del movimiento por correas se debe al rozamiento de éstas sobre las poleas. El valor del rozamiento depende, sobre todo, de la tensión de la correa y de la resistencia de ésta a la tracción; es decir, del tipo de material con que está construida (cuero, fibras, etc.) así como de sus dimensiones.

Las poleas son ruedas con una o varias hendiduras, sobre las cuales se apoyan las correas.

Las correas son cintas cerradas de cuero y otros materiales que se emplean para transmitir movimiento de rotación entre dos ejes generalmente paralelos. Pueden ser de forma plana, redonda, trapezoidal o dentada.

Este sistema se emplea cuando no se quiere transmitir grandes potencias de un eje a otro. Su principal inconveniente se debe a que el resbalamiento de la correa sobre la polea produce pérdidas considerables de potencia sobre todo en el arranque. Para evitar esto, parcialmente se puede utilizar una correa dentada, que aumenta la sujeción. La figura 1.18 representa esquemáticamente el sistema.

1.5.2 SISTEMA DE TRANSMISIÓN DE CADENAS Y RUEDAS DENTADAS

Mediante este sistema se consigue transmitir potencias relativamente altas entre 2 ejes distantes entre sí, sin que exista resbalamiento o desprendimiento entre las 2 ruedas dentadas y la cadena, que es el elemento de enlace entre ambas ruedas.

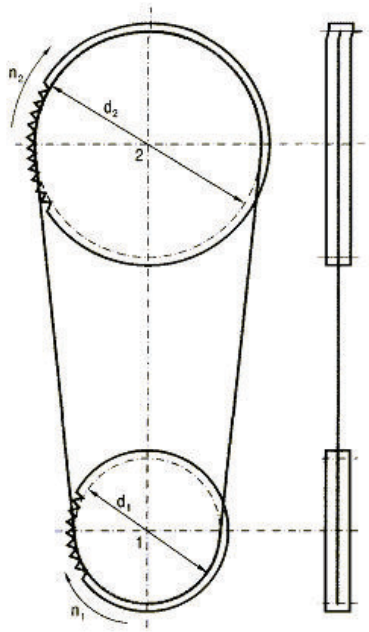


Figura 1.19 – Representación del sistema de transmisión de cadenas y ruedas dentadas

Este sistema consta de 2 ruedas dentadas montadas sobre 2 ejes paralelos sobre las cuales se adentran los eslabones flojamente articulados que componen la cadena. Si se hace girar una de las ruedas dentadas (rueda motriz) esta arrastra a la otra (rueda conducida). El movimiento rotatorio y el movimiento de torsión se transmiten entre ejes por la tracción entre la cadena y las ruedas dentadas. La figura 1.19 ilustra una representación de este tipo de sistema.

1.5.3 SISTEMA DE TRANSMISIÓN DE ENGRANAJES

Se trata de un sistema reversible capaz de transmitir potencia en ambos sentidos, en el que no son necesarios elementos intermedios como correas y cadenas para transmitir el movimiento de un eje a otro. Este sistema posee grandes ventajas con respecto a las correas y poleas como son: reducción del espacio ocupado, relación de transmisión más estable (no existe posibilidad de resbalamiento) y, sobre todo, mayor capacidad de transmisión de potencia. Sus aplicaciones son numerosas, y son de vital importancia en el mundo de la mecánica en general.
En un sistema de este tipo se suele llamar **rueda** al engranaje de mayor diámetro y **piñón** al más pequeño. Cuando el piñón mueve la rueda se tiene un sistema reductor de velocidad, mientras que cuando la rueda mueve el piñón se trata de un sistema multiplicador de velocidad. El hecho de que una rueda tenga que endentar con otra para poder transmitir potencia entre dos ejes hace que el sentido de giro de éstos sea distinto. Esto se observa en la figura 1.20.

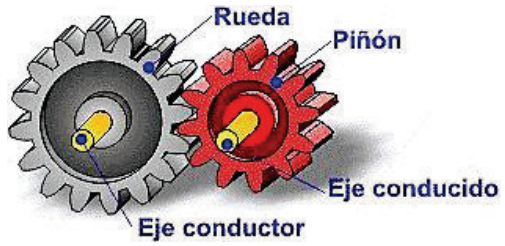


Figura 1.20 – Representación de 2 ruedas dentadas [17]

1.5.4 SISTEMA DE TRANSMISIÓN DE PIÑÓN Y CREMALLERA

Con este sistema se consigue transformar el movimiento circular que llega a la rueda dentada (**piñón**) en rectilíneo al engranar los dientes de dicha rueda con los dientes de una barra prismática (**cremallera**) que se desplaza longitudinalmente (movimiento rectilíneo). Se trata de un sistema reversible donde los dientes de la rueda dentada y de la cremallera deben tener el mismo paso a fin de que el piñón pueda deslizarse sobre la cremallera. La figura 1.21 ilustra este sistema.
En mecánica, una cremallera es un prisma rectangular con una de sus caras laterales tallada con dientes. Estos pueden ser rectos o curvados, además pueden estar dispuestos en posición transversal u oblicua. [17]

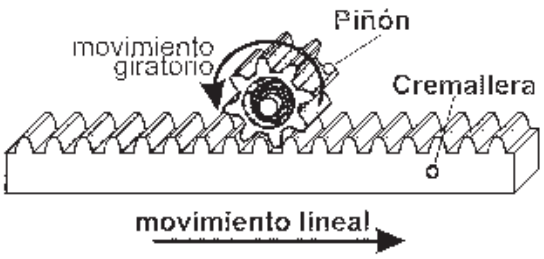


Figura 1.21 – Sistema de transmisión de piñón y cremallera [17]

1.5.5 SISTEMA DE TRANSMISIÓN DE TORNO Y POLEA

Permite convertir un movimiento giratorio en uno lineal continuo, o viceversa. Este mecanismo se emplea para la tracción o elevación de cargas por medio de una cuerda. Consiste básicamente en un cilindro horizontal (**tambor**) montado sobre dos soportes y sobre el que se enrolla (o desenrolla) una cuerda o cable cuando se aplica un movimiento giratorio a su eje a través de una manivela. Los soportes permiten mantener el eje del torno en una posición fija sobre una base; mientras que la manivela se encarga de imprimir al eje el movimiento giratorio, tal como se ilustra en la figura 1.22.

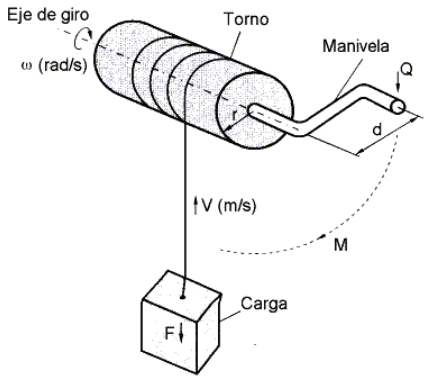


Figura 1.22 – Sistema de transmisión de torno y polea [16]

En sistemas más complejos la manivela se sustituye por un motor eléctrico con un sistema multiplicador o reductor de velocidad. [17]

Cuanto más larga es la manivela o más fino es el tambor, más peso se puede levantar. Este es un mecanismo muy frecuente en grúas y sistemas de izaje. [18]

1.6 LOS PUENTES GRÚA

La grúa es una máquina de funcionamiento discontinuo destinada a la elevación y traslado de cargas suspendidas de un gancho o de cualquier otro accesorio de prensión. Una grúa tipo puente consta de un elemento portador formado por una o dos vigas móviles, apoyadas sobre 2 carriles que se encuentran elevados por unos postes y sobre las que se desplaza el carro con los mecanismos elevadores, todo ello dispuestos sobre una estructura resistente. [19]

Se distingue 3 movimientos diferentes con los que se cubre el espacio de trabajo para el traslado de la carga:

- ***El movimiento longitudinal:*** Se lleva a cabo mediante la traslación de la viga principal o puente a través de los carriles elevados.
- ***El movimiento transversal:*** Se realiza mediante el desplazamiento de un polipasto o carro sobre uno o dos carriles dispuestos sobre la viga principal
- ***El movimiento vertical:*** Se ejecuta a través del mecanismo de elevación: polipasto o carro. [20]

En la figura 1.23 se aprecia los elementos de acción de los 3 movimientos.

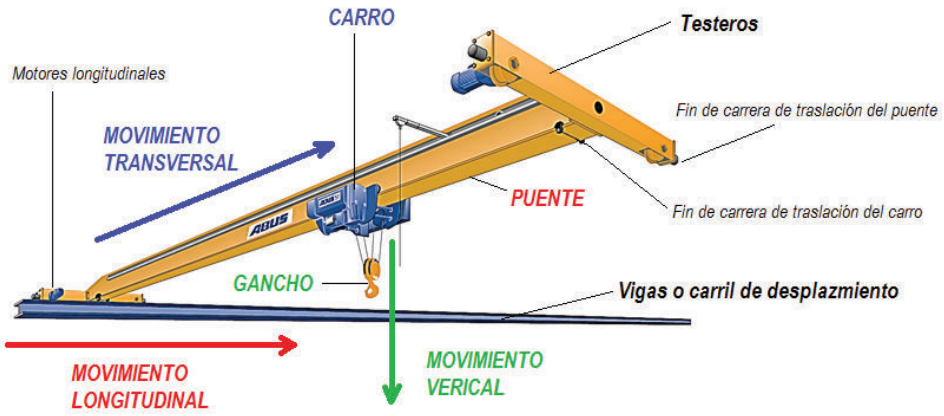


Figura 1.23 – Elementos básicos del puente grúa y sus desplazamientos

1.6.1 PARTES PRINCIPALES DEL PUENTE GRÚA [21] [22]

Viga principal: Es el miembro principal de carga, constituido por perfiles estructurales cargados transversalmente al eje de la viga generando esfuerzos de flexión. Se utiliza una viga principal si se va a elevar cargas bajas, pero si las

cargas a elevar son altas, se utilizan 2 vigas principales constituyendo un puente grúa birriel.

Testeros: Son carros de traslación que mueven la viga principal del puente a lo largo de su corredera. Están ubicados a los extremos de la viga principal.

Vigas de desplazamiento: Forman parte del puente grúa y sobre las cuales se deslizan los carros testeros apoyados sobre unas guías, transportando así la carga a lo largo del espacio de trabajo.

Columnas: Son las encargadas de soportar todo el peso del conjunto. Están distribuidas a lo largo de toda la viga de desplazamiento.

Motores de movimiento longitudinal: Como se puede ver en la figura 1.23, los motores aportan con la energía motriz a los testeros para mover el puente grúa en su movimiento longitudinal a lo largo de la corredera.

Botonera de control: Son utilizados para el manejo del puente. Están constituidos por una serie de botones los cuales permiten el desplazamiento transversal o longitudinal del puente así como la subida y bajada del gancho. Los mandos se los puede realizar desde una cabina de control, si el puente lo tuviera, o desde cualquier ubicación que pueda cubrir la botonera de control.

Polipasto: Constituye el componente que está sujeto a la viga principal del puente y no es más que un conjunto de motores acoplados a un sistema de poleas y cables destinados a variar fuerzas y velocidades que se aplican en un movimiento vertical para elevar cargas. Lleva incorporado dos o más poleas para minimizar el esfuerzo. Lo anterior se observa en la figura 1.24.

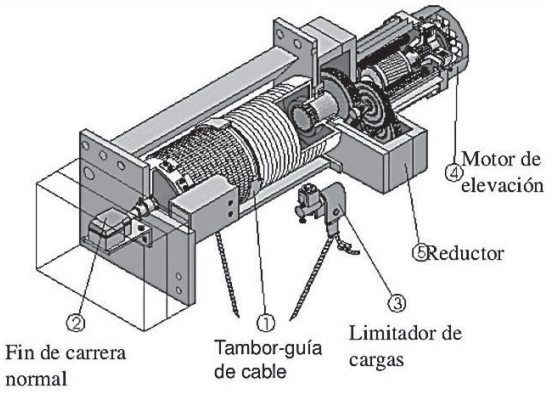


Figura 1.24 – Estructura interna y componentes básicos del polipasto

Los 3 elementos principales que se pueden diferenciar del puente grúa son:

El puente: El cual se desplaza a lo largo del área de trabajo.
El carro: Que se desplaza sobre el puente y recorre el ancho del área de trabajo.
El gancho: Va sujeto del carro mediante el cable principal, realizando los movimientos de subida y bajada de las cargas.

1.6.2 TIPOS DE PUENTES GRÚA [21]

Existe muchos tipos de puentes grúa que varían por la forma de su estructura, pero que parten del mismo principio en el que se ha de desplazar un sistema de polipasto, el cual permite el levantamiento de cargas. Entre los tipos de grúa más conocidos y empleados en la industria, podemos denotar los siguientes: del tipo monorriel, del tipo birriel y del tipo pórtico. Nos centraremos en la descripción del puente grúa del tipo monorriel.

Puente grúa monorriel: Está constituido por una sola viga y representa una solución para mover cargas cuando resulta necesario aprovechar toda la altura disponible del espacio de trabajo o cuando éste no sea muy ancho. Están equipados con polipasto y suelen disponer de doble velocidad en todos los movimientos (elevación, traslación del carro y traslación del puente). La fig. 1.25 indica la estructura de un puente grúa monorriel.

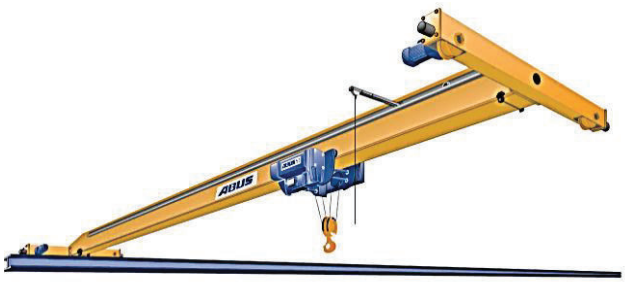


Figura 1.25 – Puente grúa monorriel

1.7 LA TABLET [23]

Una Tablet es un dispositivo electrónico similar a una computadora, pero con ciertas características físicas y funcionales que las diferencian ampliamente. Están integradas en una pantalla táctil mediante la cual se interactúa directamente

con el tacto sin la necesidad de un periférico externo conectado a éstas, como puede ser un mouse o un teclado. Este último se ve reemplazado por un teclado virtual integrado en el sistema operativo multimedia de la Tablet.

El término Tablet puede aplicarse a una variedad de formatos que difieren en el tamaño o la posición de la pantalla con respecto a un teclado, pero que tienen las mismas prestaciones.

Los usos que se les suelen dar a estos dispositivos son variados y van mucho más allá que los usos convencionales de conexión a internet y juegos. Su tamaño, varía de acuerdo a las prestaciones que brinde, desde las pequeñas de 7" hasta las más grandes y funcionales de 20", en las que se integran varios puertos para conectar algún periférico lo que hace que su tamaño aumente. Sin embargo, esto no limita a la Tablet de brindar al usuario la característica de portabilidad, pues se adapta a cualquier uso y puede además ser usado en cualquier entorno.

Otro punto importante de la Tablet es su resolución de pantalla. Desde sus inicios, los propulsores de estos aparatos compensaron su pequeño tamaño con altas resoluciones de nitidez destacables. Los fabricantes de microprocesadores y de chips gráficos han tenido mucho que ver con el crecimiento de las Tablets. El reto que éstos han vencido es ofrecer el máximo rendimiento con CPU's diminutos que no ocupan mucho espacio y que optimizan la batería.

1.7.1 El sistema operativo de la Tablet

Al igual que las computadoras, las Tablets requieren de un sistema operativo para funcionar, con la particularidad de que deben estar diseñados para dispositivos móviles. En la actualidad, los principales sistemas operativos con los que se suele equipar a las más modernas Tablets del mercado son:

- IOS del fabricante Apple.
- Android de Google, basado en Linux
- Windows RT, una síntesis de Windows 8.

1.8 EL BLUETOOTH

1.8.1 LA TECNOLOGÍA BLUETOOTH

Bluetooth es un estándar de conectividad inalámbrica que suprime el cableado entre dispositivos fijos o portátiles, manejando los servicios de comunicaciones de voz y datos. Está diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en terminales de bajo costo. La tecnología inalámbrica Bluetooth consta de hardware, software y la capacidad de inter operar entre los dispositivos. Los dispositivos que incorporan esta tecnología pueden comunicarse entre sí cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en lugares separados si la potencia de transmisión es suficiente.

Desde el punto de vista técnico, la tecnología Bluetooth posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4Ghz. Esto se debe a que uno de los requisitos que se establecieron a prioridad para el desarrollo de Bluetooth consistía en que tenía que ser operativo en cualquier lugar del planeta, y sólo la banda ISM (*Médico-Científica Internacional*) cumple este requisito. Bluetooth está basado en un sistema de saltos de frecuencia y división de tiempo dúplex, donde el canal queda dividido en *slots* de 625 µs. El canal de comunicación tiene una capacidad de 1Mbps. El alcance de su cobertura de transmisión, llega a los 10m de radio, aunque este valor puede aumentarse hasta los 100m con el uso de repetidores.

Según el estándar Bluetooth de la normativa IEE 802.15.1 se definen 3 clases de transmisores, cuyo alcance varía en función de su potencia de transmisión.

Tabla 1.3 – Clase de Transmisores

Clase	Potencia (pérdida de señal)	Alcance
I	100 mW (20 dBm)	100 metros
II	2,5 mW (4 dBm)	15-20 metros
III	1 mW (0 dBm)	10 metros

1.8.2 TIPOS DE REDES INALÁMBRICAS

- ***Redes inalámbricas de área personal WPAN (Wireless Personal Area Network).***

Son aquellas que cubren distancias cortas, inferiores a los 10 m. Debido a su poca cobertura, la finalidad de estas redes es comunicar cualquier dispositivo personal (un ordenador, Tablet, etc.) con sus periféricos (impresora inalámbrica), así como permitir una comunicación directa a corta distancia entre estos dispositivos. Su objetivo principal es eliminar el cableado en los dispositivos personales de uso más común ofreciendo así una mayor versatilidad. Estas redes se emplean dentro de lo que se denomina "espacio operativo personal", es decir, el espacio que rodea una persona. Las tecnologías que utilizan este tipo de redes son Bluetooth, DECT y los infrarrojos.

- ***Redes inalámbricas de área local WLAN (Wireless Local Area Network).***

Cubren distancias de varios centenares de metros. Estas redes están pensadas para crear un entorno de red local entre ordenadores situados en un mismo edificio. Tal es el caso de Wi-Fi, cuya tecnología emplea este tipo de red.

- ***Redes inalámbricas de área metropolitana WMAN (Wireless Metropolitan A. N.).***

Tienen una cobertura que va desde unos cientos de metros hasta varios km. El objetivo es poder cubrir el área de una ciudad o entorno metropolitano.

- ***Redes inalámbricas globales WWAN (Wireless Wide Area Network).***

Son los sistemas basados en tecnología móvil con posibilidad de cubrir toda una región (país o grupo de países).

1.9 COMUNICACIÓN SERIAL [24]

La conexión paralela entre el microcontrolador y los periféricos a través de los puertos de entrada/salida es una solución perfecta para las distancias cortas de hasta varios metros. No obstante, en otros casos cuando es necesario establecer comunicación entre dos dispositivos a largas distancias no es posible utilizar la conexión paralela. En vez de eso, se utiliza la conexión en serie.

Hoy en día, la mayoría de los microcontroladores llevan incorporados varios sistemas diferentes para la comunicación en serie, como un equipo estándar.

Definir cuál de estos sistemas se utiliza en un caso concreto depende de muchos factores, de entre los que se destacan los siguientes:

- ¿Con cuántos dispositivos el microcontrolador va a intercambiar datos?
- ¿Cuál es la velocidad del intercambio de datos obligatoria?
- ¿Cuál es la distancia entre los dispositivos?
- ¿Es necesario transmitir y recibir los datos simultáneamente?

Una de las cosas más importantes en cuanto a la comunicación en serie es el Protocolo que debe ser estrictamente observado. Es un conjunto de reglas que se aplican obligatoriamente para que los dispositivos puedan interpretar correctamente los datos que intercambian mutuamente. Afortunadamente, los microcontroladores se encargan de eso automáticamente, así que el trabajo de programador/usuario es reducido a la escritura y lectura de datos.

1.9.1 VELOCIDAD DE TRANSMISIÓN SERIAL

La velocidad de transmisión serial (Baud Rate) es el término utilizado para denotar el número de bits transmitidos por segundo (bps).

1.9.2 UART (UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER)

UART (es decir, Transmisor/Receptor Asíncrono Universal), corresponde a un tipo de conexión asíncrona, lo que significa que no se utiliza una línea especial para transmitir la señal de reloj. En algunas aplicaciones este rasgo es crucial (por ejemplo, en mandar datos a distancia por RF). Puesto que se utiliza sólo una línea de comunicación, tanto el receptor como el transmisor reciben y envían los datos a la misma velocidad que ha sido predefinida para mantener la sincronización necesaria. Esto es una manera simple de transmitir datos, puesto que básicamente, representa una conversión de datos de 8 bits de serie a paralelo, cuando se trata de datos recibidos (de entrada) y de convertir datos de paralelo a serie para transmisión (de salida). La velocidad de transmisión no es alta, es hasta 1 Mbit/seg.

1.10 MATLAB [25]

MATLAB es un entorno integrado de programación para el desarrollo de algoritmos, análisis de datos, visualización y cómputo numérico dando un lenguaje de alto nivel y desarrollo de herramientas, que permiten rápidamente desplegar y analizar sus algoritmos y aplicaciones. El lenguaje de MATLAB soporta las operaciones vectoriales y matriciales que son fundamentales tanto para problemas científicos como para ingeniería. Los archivos generados en MATLAB se guardan con extensión *.m.

Con el lenguaje de MATLAB se puede programar y desarrollar algoritmos más rápidamente que con los lenguajes de programación tradicionales porque no necesita desarrollar tareas administrativas de bajo nivel, como declarar variables, especificar tipos de datos y reservar memoria. En muchos casos, MATLAB elimina la necesidad de ciclos “for”. Al mismo tiempo, MATLAB da todas las características de un lenguaje de programación tradicional, incluyendo operadores aritméticos, control de flujo, estructura de datos, programación orientada a objetos y características de depurado. Se puede ejecutar comandos o grupos de comandos al mismo tiempo, sin compilar y enlazar, permitiendo una iteración rápida a una solución óptima.

MATLAB tiene un amplio rango de aplicaciones, incluyendo procesamiento de señales e imágenes, comunicaciones, sistemas de control, pruebas y medición, modelado y análisis financiero y biología computacional. Para un millón de ingenieros y científicos en la industria y la academia, MATLAB es el lenguaje de cómputo técnico.

1.11 SIMULINK

Es un entorno de programación gráfico interactivo para el modelado, análisis y simulación de una gran variedad de sistemas dinámicos (discretos, análogos e híbridos) mediante la utilización de diagramas de bloques. Trabaja totalmente integrado con MATLAB, dando acceso inmediato a una amplia gama de herramientas que le permiten desarrollar algoritmos, analizarlos y visualizar

simulaciones, personalizar el ambiente de modelado, definir señales, parámetros y datos de prueba. Los modelos SIMULINK se guardan en ficheros con extensión *.mdl. Tiene gran aplicación en la automatización del diseño mediante simulación de sistemas dinámicos.

1.11.1 EL ENTORNO SIMULINK

SIMULINK ofrece un editor gráfico así como bibliotecas de bloques personalizables para modelar y simular sistemas dinámicos. Se integra con MATLAB, lo que permite incorporar algoritmos de MATLAB en los modelos y exportar los resultados de la simulación a MATLAB para llevar a cabo análisis más complejos.

Como las funciones están representadas en forma gráfica mediante bloques, se debe seleccionar la función o funciones de las diferentes librerías y arrastrarlas hasta el entorno de trabajo. La interconexión entre los diferentes bloques se las realiza manteniendo presionado el mouse entre los puertos de entrada y salida de dichos bloques. La figura 1.26 muestra a la derecha el entorno de trabajo de SIMULINK, así como, la ventana de sus librerías a la izquierda.

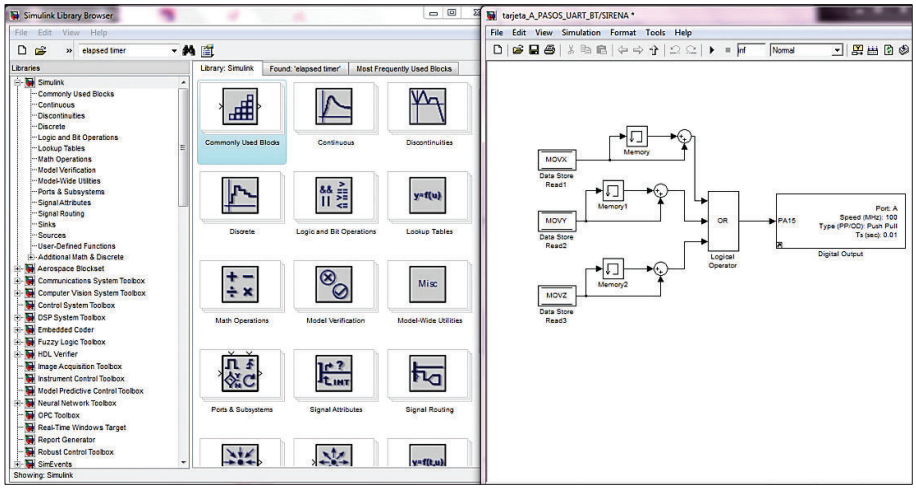


Figura 1.26 – Entorno de trabajo de MATLAB y sus librerias

1.11.2 PRINCIPALES CARACTERÍSTICAS DE SIMULINK

- Librería amplia y expandible de bloques predefinidos.

- Editor gráfico interactivo e intuitivo para ensamblar y manejar sus diagramas de bloques.
- Capacidad para manejar diseños complejos por modelos segmentados en componentes jerárquicos.
- Contiene un Explorador de modelos para navegar, crear, configurar y buscar todas las señales, parámetros, propiedades y código generado asociado con su modelo.
- Herramientas de análisis y diagnóstico de modelos para asegurar la consistencia del modelo e identificar sus errores.

1.11.3 LIBRERÍA WAIJUNG [26]

Las librerías de SIMULINK contienen los bloques operacionales agrupados de acuerdo a diferentes propósitos comunes. Dentro de estas librerías existe una en particular, se trata de la librería de bloques Waijung. Inicialmente esta librería no se encuentra disponible en el entorno de SIMULINK, y únicamente se tiene acceso a ésta después de haber descargado e instalado el software desde el enlace web: http://waijung.aimagin.com/. Una vez incorporado esta librería complementaria, podremos encontrar dentro de ésta, todas las herramientas necesarias que nos permiten interactuar con el microcontrolador. De entre los bloques operacionales más importantes se tienen:

- ***Librería IO:*** Dentro de esta se encuentran los bloques que implementan los módulos de entrada y salida digital del microcontrolador, para generar la lógica de salida o entrada digital del microcontrolador. Se pueden tener tantas entradas como salidas digitales en un puerto especificado del microcontrolador, según se haya realizado la configuración de cada una.
- ***Librería TIM:*** Aquí se pueden encontrar los diferentes bloques referentes a los timers, temporizadores así como generadores de señales PWM.
- ***Librería UART:*** Esta librería contiene los bloques referentes a la activación y configuración del módulo UART el cual es usado cuando la aplicación necesita enviar o recibir datos de un dispositivo externo mediante el protocolo UART.

1.12 ANDROID

Android es una plataforma o sistema operativo inicialmente pensado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o Tablets. El núcleo de Android está formado por el sistema operativo Linux.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado grandes expectativas, y a la vez, ha tenido gran acogida tanto por parte de los usuarios, como de la industria. En la actualidad, se está convirtiendo en la alternativa estándar frente a otras plataformas como iPhone, Windows Phone o BlackBerry. Una de las mayores fortalezas del entorno de aplicación de Android es que aprovecha el lenguaje de programación de Java.

1.12.1 Características de Android

- *Plataforma realmente abierta*. Es una plataforma de desarrollo libre basado en Linux y de código abierto. Una de las grandes ventajas es que se puede usar y customizar[1] el sistema sin pagar royalties[2].
- *Adaptable a cualquier tipo de hardware*. Android no ha sido diseñado exclusivamente para su uso en teléfonos y Tablets. Hoy en día, se puede encontrar variedad de dispositivos táctiles que incorporan esta plataforma.
- *Portabilidad asegurada*. Las últimas aplicaciones son desarrolladas en Java para asegurar su ejecución en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
- *Optimizado para baja potencia y poca memoria*. Android utiliza una máquina virtual especial (Dalvik), implementación de Google de la máquina virtual estándar de Java optimizada para cubrir las limitaciones de los dispositivos móviles donde ha de correr (poca memoria y procesador limitado).
- *Alta calidad de gráficos y sonido*. Gráficos vectoriales suavizados en 3D, animaciones inspiradas en Flash, etc.

---

[1] Se trata de una adaptación del término inglés *customize*, que refiere a modificar algo de acuerdo a las preferencias personales.

[2] Pago que se efectúa al titular de derechos de autor, patentes a cambio del derecho a usarlos

En conclusión, Android ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para diferentes tipos de dispositivos.

1.12.2 Aplicaciones

Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en desarrollar aplicaciones utilizando C/C++. Para esta opción, se puede utilizar el Android NDK (Native Development Kit).

1.12.3 Las versiones de Android y niveles de API [27]

Antes de empezar a programar en Android, se debe elegir la versión del sistema para la que se desee realizar la aplicación. Es muy importante observar que hay clases y métodos, que están disponibles a partir de una versión y que si se los va a utilizar, se debe conocer la versión mínima necesaria.
Cuando se ha lanzado una nueva plataforma siempre ha sido compatible con las versiones anteriores. Es decir, solo se añaden nuevas funcionalidades y en el caso de modificar alguna funcionalidad no se elimina sino que se etiquetan como obsoletas pero se pueden continuar utilizando.
A continuación, se enumeran las plataformas lanzadas hasta la fecha. Las plataformas se indican de tres formas alternativas: Versión, nivel de API y nombre comercial. El nivel de API corresponde a números enteros comenzando desde 1. Para los nombres comerciales, se han elegido nombres de diferentes postres en inglés. En cada versión el postre elegido empieza por una letra distinta siguiendo un orden alfabético:

- Android 1.0 Nivel de API 1 (septiembre 2008).
- Android 1.1 Nivel de API 2 (febrero 2009).
- ***Cupcake***: Android 1.5 Nivel de API 3 (abril 2009).
- ***Donut***: Android 1.6 Nivel de API 4 (septiembre 2009).
- ***Éclair:*** Android 2.0 Nivel de API 5 (octubre 2009).
    - Android 2.1 Nivel de API 7 (enero 2010).
- ***Froyo:*** Android 2.2 Nivel de API 8 (mayo 2010).

- ***Gingerbread:*** Android 2.3 Nivel de API 9 (diciembre 2010).
- ***Honeycomb:*** Android 3.0 Nivel de API 11 (febrero 2011).
    - Android 3.1 Nivel de API 12 (mayo 2011).
    - Android 3.2 Nivel de API 13 (julio 2011).
- ***Ice Cream Sandwich:*** Android 4.0 Nivel de API 14 (octubre 2011).
    - Android 4.0.3 Nivel de API 15 (diciembre 2011).
- ***Jelly Bean: Android:*** 4.1 Nivel de API 16 (julio 2012).
    - Android 4.2 Nivel de API 17 (noviembre 2012).
- ***KitKat: Android:*** 4.4 Nivel de API 19 (octubre 2013).
- ***Lolipop: Android:*** 5.0 Nivel de API 21 (noviembre 2014).

Las 2 primeras versiones que hubieran correspondido a las letras A y B, no recibieron nombre. La figura 1.27 da la representación gráfica con que se conoce comercialmente a cada versión de Android que está en constante desarrollo.

Figura 1.27 – Representación gráfica de cada versión de Android

1.12.4 Eleccion de la plataforma de desarrollo

A la hora de seleccionar la plataforma de desarrollo se debe consultar si se necesita alguna característica especial que solo esté disponible a partir de una versión. Todos los usuarios con versiones inferiores a la selección no podrán instalar la aplicación. Por lo tanto es recomendable seleccionar la menor versión posible que pueda soportar la aplicación que se va a crear. Para ayudarnos a

tomar la decisión de que plataforma utilizar resulta útil consultar los porcentajes de utilización en: http://developer.android.com/about/dashboards/index.html.
Tras estudiar la figura 1.28 podemos destacar el reducido número de usuarios que utilizan la versión 1.1 (0.4%) por lo tanto puede ser buena idea utilizar como versión mínima la 2.3 para desarrollar nuestro proyecto, dado que daríamos cobertura a más del 95% de los terminales. Las versiones 1.x han tenido muy poca difusión y presentan tendencia a disminuir es por eso que no se las ha representado en el esquema estadístico. La versiones más reciente con un 3.3%, todavía son minoritarias pero se prevé que este porcentaje vaya aumentando. No obstante, estas cifras son referenciales y cambian mes a mes por lo que se recomienda mantenerse al tanto para el desarrollo de aplicaciones.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

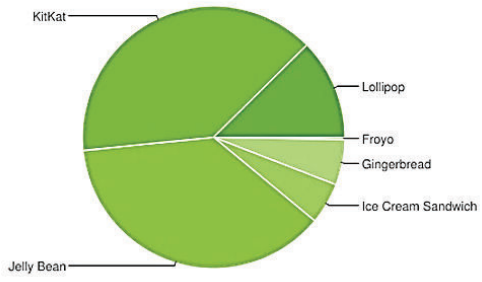


Figura 1.28 – Indicador de uso de las diferentes plataformas Android por los dispositivos.

1.12.5 Desarrollo de aplicaciones en Android Studio [28]

Android Studio proporciona un extenso marco de desarrollo que permite construir aplicaciones para dispositivos móviles en un entorno de lenguaje Java.
Las aplicaciones de Android están escritas en el lenguaje de programación Java como una combinación de componentes distintos que se puede invocar de forma individual, es decir una actividad individual proporciona una sola pantalla para una interfaz de usuario. Las herramientas del SDK de Android compilan su código junto con todos los datos y archivos en un recurso APK, un paquete de Android,

que es un archivo histórico con una extensión *.apk. Un archivo APK contiene todo el contenido de una aplicación Android y es el archivo que los dispositivos con plataforma Android utilizan para instalar la aplicación.

Una vez instalado en un dispositivo, cada aplicación Android vive en su propio entorno limitado de seguridad. De esta manera, el sistema Android implementa el principio de privilegios mínimos, es decir, cada aplicación, por defecto, sólo tiene acceso a los componentes que necesita para hacer su trabajo y nada más. Esto crea un ambiente muy seguro en el que una aplicación no puede acceder a partes del sistema a las que no se le ha dado permiso.

CAPÍTULO 2

CONSTRUCCIÓN Y MONTAJE

2.1 DISEÑO DE LA ESTRUCTURA MECÁNICA

El diseño del prototipo está basado en el modelo de puente grúa del tipo monorriel, esto por su sencillez frente a los otros modelos de puentes grúa existentes que únicamente varían por la forma de su estructura, pero que al final parten del principio de desplazar un sistema de polipasto para permitir el levantamiento de cargas.

En base al modelo de puente grúa del tipo monorriel y considerando sus partes principales, el prototipo está estructurado de la siguiente manera:

- Un armazón rectangular compuesto de tubos cuadrados de hierro de 5/8", siendo las dimensiones las siguientes:
  - 62cm de ancho, correspondiente al eje X.
  - 72cm de largo, correspondiente al eje Y.
  - 60cm de alto, correspondiente al eje Z.
- Sobre cada lado de la parte superior de la estructura se montan 2 rieles metálicas del tipo U-20, las que nos sirven para simular las guías o correderas principales del puente grúa. La característica de este tipo de riel es que nos permite montar sobre ésta un tipo de corredera especial con rodamientos, para reducir la fricción en el desplazamiento. Estos rodamientos simulan los testeros o carros de traslación que mueven la viga principal del puente a lo largo de su corredera, y van dispuestos uno a cada extremo de la viga principal.
- La viga principal está representada por una riel metálica del tipo U-100. Dicha riel cuenta con una doble guía interna que permite introducir un sistema de doble corredera, esto, con el fin de simular el sistema de carro del puente grúa.

- Sobre el sistema que simula el carro del puente grúa, se acopla un sistema de transmisión de torno y polea que representa al polipasto del puente grúa. Este sistema se complementa con un juego de poleas que descienden desde la parte superior del carro apoyadas por un cable, hasta la parte inferior donde termina en un elemento de sujeción.
- El elemento de sujeción tiene una particularidad en el prototipo, pues el convencional gancho que permite sujetar los objetos ha sido reemplazado por una pinza de sujeción con el fin de agilizar el proceso de anclaje.

En la figura 2.1 se puede apreciar el esquema general de la estructura con los elementos descritos anteriormente.

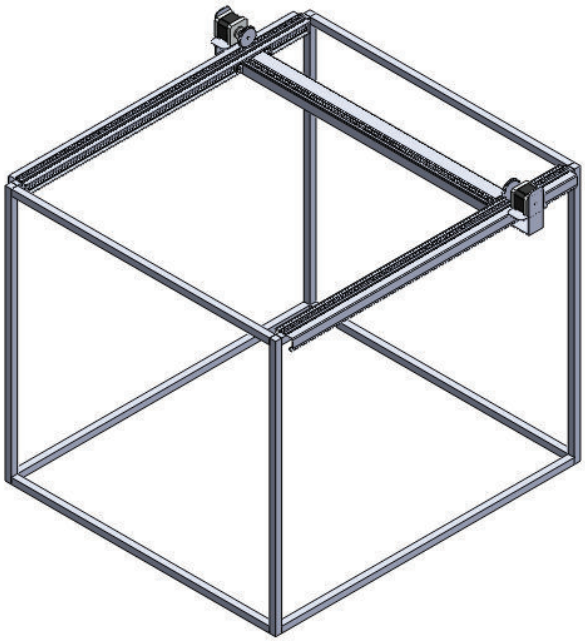


Figura 2.1 – Esquema general de la estructura

2.2 MONTAJE DE MOTORES Y SISTEMAS DE TRANSMISIÓN

La estructura descrita en la sección anterior sirve para hacer el montaje de los mecanismos de transmisión; así como, de los elementos motrices para que con todo ello en conjunto se consiga los desplazamientos X, Y, Z del prototipo.

Los motores que se emplean para conseguir el movimiento del prototipo se distribuyen de la siguiente manera:

2.2.1 PARA EL DESPLAZAMIENTO EN LOS EJES X, Y

Para el desplazamiento del puente (desplazamiento en el eje Y) se emplean 2 motores PAP bipolares de imán permanente de 4 hilos ubicados uno a cada extremo de éste. Para el desplazamiento del carro (desplazamiento en el eje X) se emplea únicamente un motor PAP bipolar de imán permanente. Este último está montado sobre el puente y recorre a lo largo de éste. El fabricante de los 3 motores es Geetech y sus características son las siguientes:

- Voltaje: 5 [V] DC.
- Corriente: 1,2 [A].
- Resolución: 1.8 grados por paso.
- Torque: 2,6 N.cm

Cada motor tiene acoplado en su eje un piñón de 39 dientes los cuales engranan con los dientes de una barra prismática o cremallera distribuida a lo largo de los ejes X, Y. Con ello, se logra transformar el movimiento circular en rectilíneo.

2.2.2 PARA EL DESPLAZAMIENTO EN EL EJE Z

Para el funcionamiento del sistema de polipasto (desplazamiento en el eje Z) se emplea un motor PAP unipolar de imán permanente de 6 hilos marca OKI Stepping Motor. Sus características son las siguientes:

- Voltaje: 12 [V] DC.
- Corriente: 0,6 [A].
- Resolución: 7.5 grados por paso.
- Torque: 2.94N.cm

Al eje de este motor se acopla un pequeño piñón de 24 dientes que engranan con los dientes de una correa dentada, para transmitir el movimiento del eje del motor hacia otra rueda dentada, que en definitiva es parte del sistema de torno y polea; el cual permite, el ascenso y descenso de la pinza para la sujeción y posterior izaje de los objetos.

2.2.3 PARA EL ELEMENTO DE SUJECIÓN

El elemento de sujeción a utilizar es una pinza de movimiento lineal ya que los dedos se abren y cierran ejecutando un movimiento paralelo entre sí. Esta condición es indispensable pues posteriormente se adapta un sensor infrarrojo a los dedos de la pinza, en el cual, el haz de luz entre el emisor y receptor debe permanecer lineal hasta la detección del objeto.

Finalmente, la apertura y cierre de la pinza, como elemento de sujeción, está gobernada por el funcionamiento de un servomotor, cuyo eje se acopla al eje del pivote de la pinza.

2.3 ESQUEMA DE CONTROL

La figura 2.2 muestra el diagrama en bloques de la funcionalidad del prototipo y en base al cual se d una explicación detallada de cada uno de los bloques que componen este diagrama.

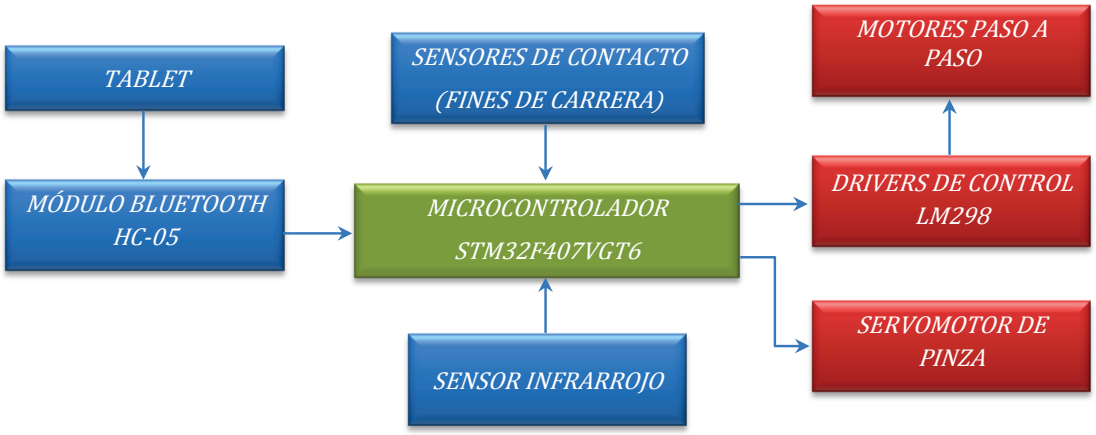


Figura 2.2 – Esquema de control del prototipo

Como se aprecia en la figura 2.2, el sistema de control para el prototipo está compuesto tanto de bloques de entrada como de salida, siendo estos últimos los que representan el objetivo de todo el proyecto. La respuesta que se tenga en los bloques de salida depende de los valores que se hayan enviado a los bloques de entrada. Todo ello en conjunto, está ligado a un bloque central, quien gobierna las funciones de recepción de datos del exterior para ser procesados y de esta

manera, tener el control sobre los bloques de salida. Cabe destacar que el correcto funcionamiento de todo el sistema de control, no depende de un único elemento, pues todos los bloques trabajan en conjunto; sin embargo, la finalidad del proyecto es poder controlar tanto los motores paso a paso a través de los drivers, así como tener el control de la pinza para la sujeción de objetos.

2.3.1 BLOQUE MICROCONTROLADOR

Representa el bloque más importante ya que gobierna a los bloques de salida dependiendo de los valores que se tengan en los bloques de entrada que están ligados a él. Está compuesto por un microcontrolador STM32F407VGT6.

2.3.1.1 Microcontrolador STM32F407VGT6

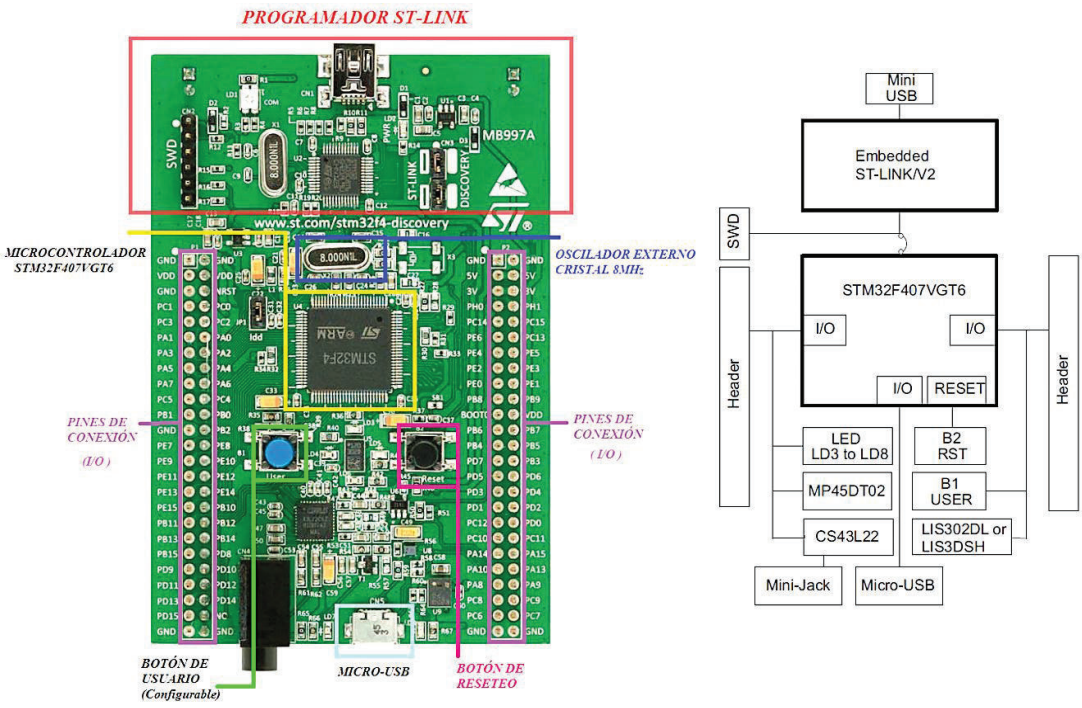


Figura 2.3 – Vista global de la tarjeta STM32F4 DISCOVERY

El microcontrolador STM32F407VGT6 está integrado en una Placa STM32F4 DISCOVERY, la que a su vez, incluye en un único elemento los periféricos necesarios para el desarrollo de aplicaciones con el microcontrolador. Este microcontrolador viene en un encapsulado cuadrado plano de bajo perfil de 100

pines LQFP (*Low-Profile Quad Flat Package*), el cual, es un encapsulado de circuito integrado para montaje superficial, cuyos pines de conexión se extienden por los cuatro lados en grupos simétricos de 25 pines por cada lado para este micro. [29]
Las características más importantes tomadas en cuenta para el desarrollo del proyecto se resumen en la tabla 2.1.

Tabla 2.1 – Características básicas del microcontolador STM32F407VGT6

Memoria Flash	1 Mbyte
Memoria RAM	192 KBytes
Frecuencia de operación	168 MHz
Puertos de entrada y salida	A, B, C, D, E, H
Puerto serial asincrónico	UART

La placa STM32F4 DISCOVERY, tiene incorporado su propio programador, el ST-LINK. Se trata de una herramienta externa desarrollada por ST Microelectronics para la programación y depuración de sistemas micro controlados, embebidos en un único conjunto. La programación y depuración, se efectúa a través de un cable USB del tipo A a mini-B, especial para programación y depuración, entre el ordenador y la placa STM32F4 DISCOVERY.
Otras de las características que ofrece la placa STM32F4 DISCOVERY son:

- La fuente de alimentación, que se proporciona ya sea por el PC a través del cable USB o mediante una fuente de alimentación externa de 5V.
- Fuente de voltaje para aplicaciones externas: 3V y 5V.
- Pese a que el microcontrolador STM32F407VGT6 tiene dos osciladores internos RC, utiliza un oscilador externo que es un cristal de 8MHz para el oscilador principal, el cual se encuentra visible en la tarjeta. [30]

Se pueden apreciar otros elementos distribuidos en toda la tarjeta STM32F4 DISCOVERY, pero que no aportan ninguna utilidad para el desarrollo de nuestro proyecto, por lo que no se da detalle sobre ellos. Si se requiere información adicional acerca de este microcontrolador, así como de los otros periféricos que conforman la tarjeta, puede hacerse uso de la hoja de datos del mismo. La figura 2.3, ilustra de manera general la conexión entre el microcontrolador STM32F407VGT6 y sus periféricos principales.

2.3.1.2 Arquitectura interna del microcontrolador STM32F407VGT6

Como se había dicho anteriormente, el corazón de esta tarjeta es el microcontrolador STM32F407VGT6. La figura 2.4 muestra el diagrama en bloques de la arquitectura interna de este microcontrolador, en la que se puede identificar como está conectada la memoria flash, memoria RAM, los puertos, etc.

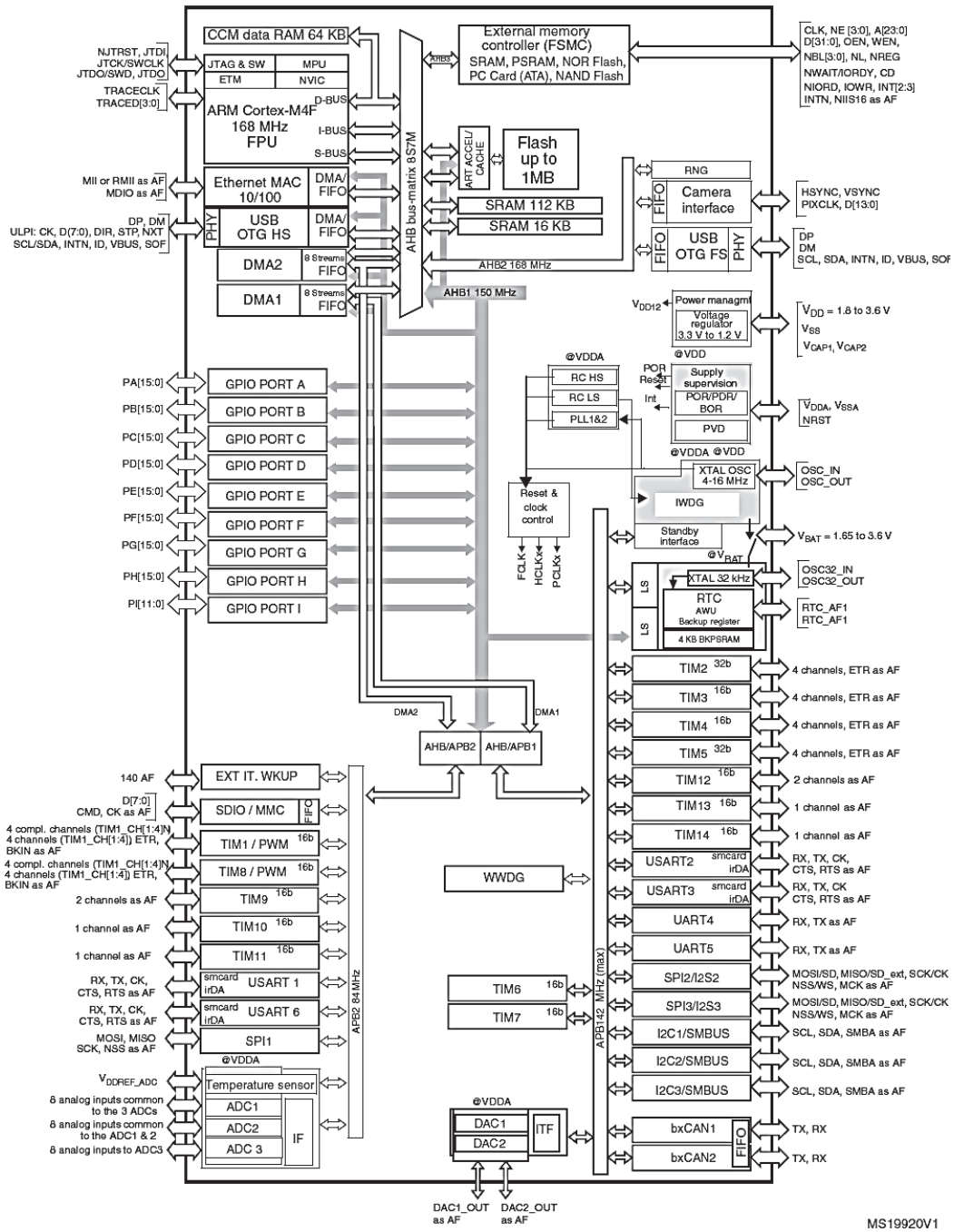


Figura 2.4 – Diagrama en bloques del microcontrolador STM32F407VGT6 [30]

2.3.1.3 Configuración de pines de la tarjeta STM32F4 DISCOVERY

Todos los pines I / O del microcontrolador STM32F407YGT6 sobresalen de la tarjeta STM32F4 DISCOVERY, a través de conectores macho que se agrupan en 2 secciones diferentes (P1, P2) distribuidas, una a cada lado de la placa a lo largo de su longitud. Cada grupo consta de 2 columnas de 25 pines, es decir 50 pines por cada lado, esto sumado con los 50 pines del otro extremo que se distribuyen de la misma forma, dan un total de 100 pines. Los nombres de los pines son serigrafiados junto a cada conector macho. Los cabezales de pin sobresalen de la parte superior de la tarjeta, para permitir la conexión de un conector hembra.
Debido al extenso número de pines que posee este microcontrolador, nos limitamos a detallar únicamente la función de los pines que fueron utilizados para el proyecto. Sin embargo, la figura 2.5a muestra un diagrama del encapsulado del microcontrolador con la numeración de cada pin. A su vez, la figura 2.5b muestra la distribución de pines en la placa STM32F4 DISCOVERY. Si se requiere información adicional acerca de la función que cumple cada pin, se puede recurrir a la hoja de datos del microcontrolador.

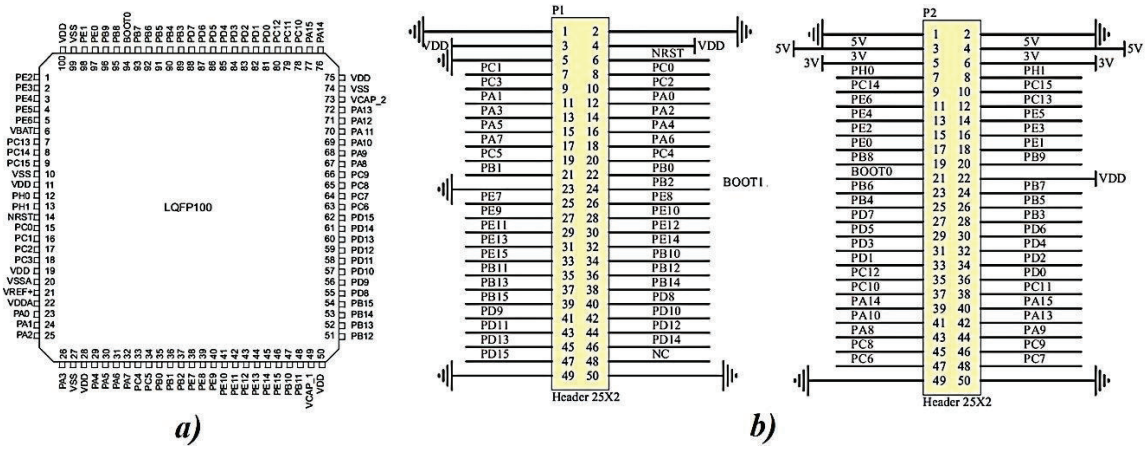


Figura 2.5 – Distribución de pines: a) En el microcontrolador; b) En la tarjeta

2.3.1.4 Distribución de pines del Microcontrolador STM32F407VGT6

Los pines de entrada/salida del microcontrolador, están agrupados en 6 puertos: Puertos A, B, C, D y E con 15 líneas y el pórtico H únicamente con 2 líneas. Los pórticos F y G no están disponibles para este microcontrolador, pues se los encuentra en otros con mayor número de pines. No todas las líneas de los

pórticos están disponibles en los pines de la tarjeta, pues fueron empleados para la interconexión de sus periféricos.

Tabla 2.2 – Lista de pines y variables de entrada utilizadas en el proyecto

ENTRADAS				
VARIABLE		PIN		FUNCIÓN
FX+		8 – P1	PC0	Fin de carrera para el desplazamiento en el eje X en avance
FX-		7 – P1	PC1	Fin de carrera para el desplazamiento en el eje X en retorno
FY+		10 – P1	PC2	Fin de carrera para el desplazamiento en el eje Y en avance
FY-		9 – P1	PC3	Fin de carrera para el desplazamiento en el eje Y en retorno
FZ-		20 – P1	PC4	Fin de carrera para el desplazamiento en el eje Z en retorno
SENS	SIND	NC	-	Emite la señal de luz infrarroja, al conectarse a una fuente de 3V.
	SINT	19 – P1	PC5	Receptor del sensor de luz infrarroja
BLUET	RX	23 – P1	PB6	Recepta los datos provenientes de la salida Tx del módulo Bluetooth
	TX	24 – P2	PB7	-

Tabla 2.3 – Lista de pines y variables de salida utilizadas en el proyecto

SALIDAS					
VARIABLE			PIN		FUNCIÓN
PLACA 1	MOTOR X	IN4	31 – P2	PD3	El resultado de pulsos generados en este orden, es la secuencia de pasos que se envían al driver del motor PAP para ponerlo en movimiento a lo largo del eje X.
		IN3	34 – P2	PD2	
		IN2	33 – P2	PD1	
		IN1	36 – P2	PD0	
PLACA 2	MOTOR Y1-	IN4	43 – P1	PD11	Corresponden a la secuencia de pasos que debe enviar el microcontrolador al driver para conseguir el giro y desplazamiento del motor a lo largo del eje Y.
		IN3	42 – P1	PD10	
		IN2	41 – P1	PD9	
		IN1	40 – P1	PD8	
PLACA 3	MOTOR Y2+	IN4	40 – P1	PD8	Corresponden a la misma secuencia de pasos que en el caso anterior, pero en lectura inversa enviados al driver, con el fin de tener en el motor un sentido de giro y desplazamiento relativo respecto del otro motor a lo largo del eje Y.
		IN3	41 – P1	PD9	
		IN2	42 – P1	PD10	
		IN1	43 – P1	PD11	
PLACA 4	MOTOR Z	IN4	27 – P2	PD7	Corresponde a la secuencia de pasos que debe enviar el microcontrolador al driver para conseguir el desplazamiento a lo largo del eje Z.
		IN3	30 – P2	PD6	
		IN2	29 – P2	PD5	
		IN1	32 – P2	PD4	
SIRENA			40 – P2	PA15	Permite la conexión de una sirena para tener una señal auditiva cuando el puente está en funcionamiento.
SEÑAL PWM			27 – P1	PE9	La señal generada se envía al cable receptor de señal del servomotor, para la apertura y cierre de la pinza.

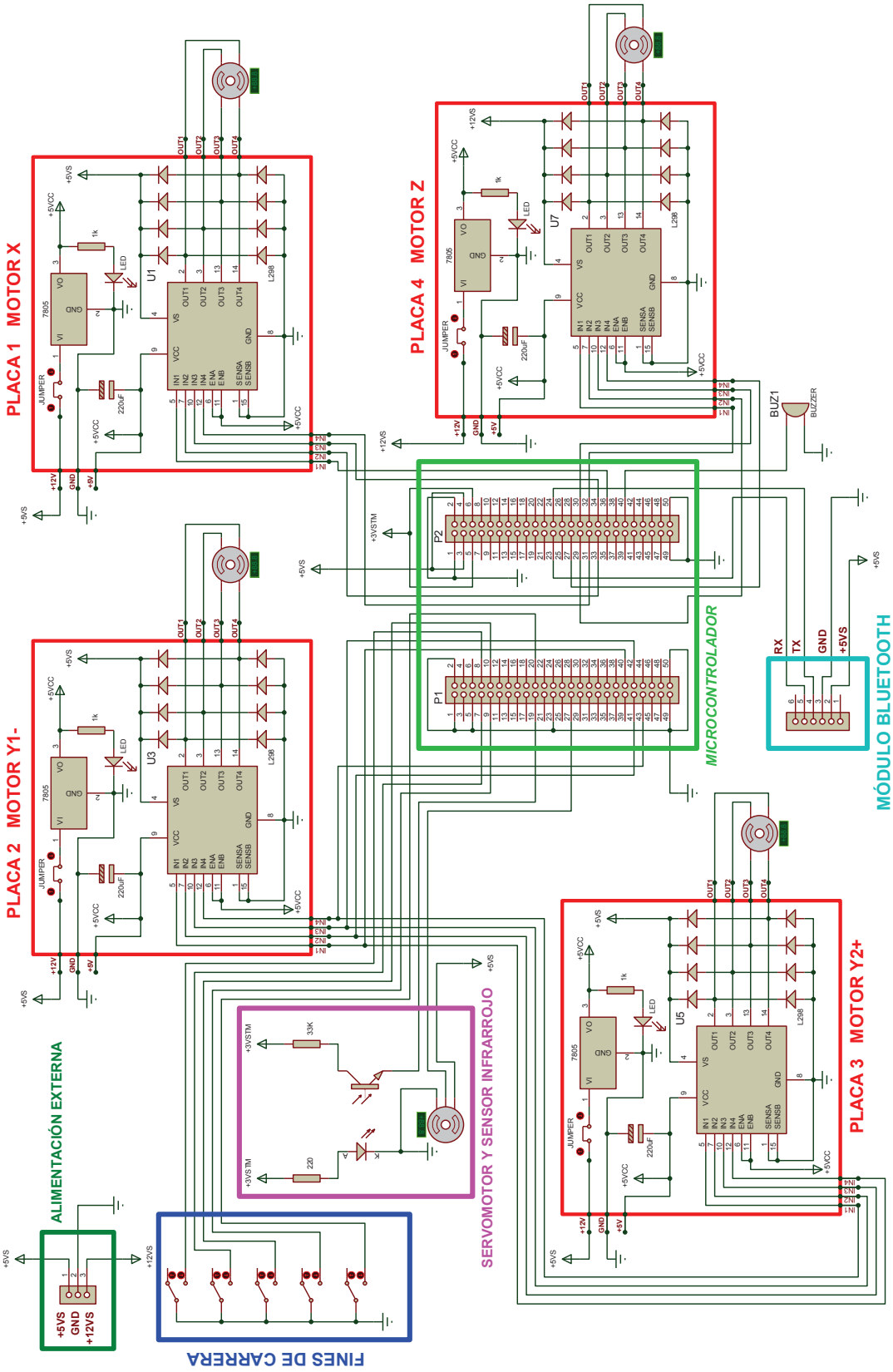


Figura 2.6 – Circuito de conexión de los diferentes bloques al microcontrolador

Las tablas 2.2 y 2.3 muestran la lista de variables de entrada y salida respectivamente así como la lista de pines empleados para la realización del proyecto. La numeración de cada pin hace referencia a los pines de la placa STM32F4 DISCOVERY.

Las funciones de las variables tanto de entrada como de salida, para la lógica de funcionamiento del prototipo, se detallan posteriormente en el desarrollo de la programación del microcontrolador (Capítulo 3). Sin embargo, para las variables de los motores que permiten el desplazamiento a lo largo del eje Y, se debe destacar que los 2 drivers (uno para cada motor) se conectan a un mismo grupo de 4 pines del microcontrolador, con la finalidad de que los 2 motores estén sincronizados y tener el funcionamiento simultáneo de ambos. Lo que difiere entre ambos, es el orden de conexión de los pines para conseguir que uno de los 2 motores gire en sentido contrario respecto del otro, esto se consigue invirtiendo el orden de la secuencia de pasos de uno de los 2 motores.

Para el caso del sensor infrarrojo, la única señal de entrada hacia el microcontrolador es la proveniente del fototransistor, pues da una señal digital en 1 o 0 según la incidencia de luz proveniente del diodo emisor de luz infrarroja.

La figura 2.6, da una vista general de la disposición de los diferentes bloques en la placa STM32F4 DISCOVERY; así como, la conexión de los elementos que conforman cada uno de los bloques.

2.3.2 BLOQUE DE SENSORES DE CONTACTO

El prototipo consta de varios elementos móviles que se desplazan a lo largo de los ejes X, Y, Z. Los motores, al ser los encargados de transmitir el movimiento a dichos elementos, necesitan de un dispositivo situado al final del recorrido para detectar la posición límite de desplazamiento. Una vez detectada tal posición, él o los motores deben dejar de actuar inmediatamente. Esto se consigue con la ayuda de los sensores de contacto o también conocidos como fines de carrera.

2.3.2.1 El sensor de contacto de tipo palanca con muelle (final de carrera)

Existe una amplia gama de sensores de contacto que se distinguen por el elemento móvil que genera la señal eléctrica de salida. Se tienen, por ejemplo,

los de palanca metálica con muelle, palanca con rodillo, varilla, de pulsador, etc. La construcción y forma de trabajo de la máquina a ser controlada, determina el tipo de actuación del sensor de contacto a ser usado. Para el desarrollo de este proyecto, basta únicamente el sensor de contacto de tipo palanca con muelle. Este es un dispositivo de control que convierte un movimiento mecánico en una señal eléctrica de control. Tiene como función principal limitar el desplazamiento de un elemento en movimiento. Su accionamiento es de tipo mecánico y está vinculado a un conjunto de contactos en el interior del dispositivo que pueden abrirse o cerrarse dependiendo de la operación que cumplan al ser accionados.
Los sensores de contacto de este tipo pueden estar, o no, dotados de un amortiguador en el punto cero. Con dicha amortiguación, se evita que la palanca sobrepase el punto cero al retornar libremente desde el ángulo hasta el cual fue accionado y, por tanto se evita la emisión de órdenes falsas. [31]

2.3.2.2 Funcionamiento de los fines de carrera en el prototipo

El funcionamiento del prototipo comprende el desplazamiento de los siguientes elementos:

- Un puente que se desplaza a lo largo del eje Y
- Un carro que se desplaza a lo largo del eje X
- Un sistema de polipasto que permite el ascenso y descenso de objetos.

Cuando uno de estos 3 elementos ejecuta su máximo desplazamiento a lo largo de su respectivo eje, entran en contacto con el accionador del fin de carrera que para este caso es una palanca, el dispositivo opera un pulsador que dispone de un muelle de recuperación para abrir o cerrar un circuito. Dicha acción se traduce en el envío de 2 señales digitales al microcontrolador, un 1L o un 0L, dependiendo de la acción que se tenga sobre el fin de carrera. Para esto, el dispositivo dispone de 3 terminales que permiten la configuración de su comportamiento inicial, es decir, se puede configurar como normalmente abierto o normalmente cerrado. Para nuestro caso, todos los fines de carrera están configurados como normalmente abiertos, por lo tanto; se envía un 0 lógico cuando el fin de carrera no se haya accionado. Si se acciona el fin de carrera se envía un 1 lógico al microcontrolador. Estas señales digitales permiten la detención o la puesta en

marcha de los motores según sea la configuración que se haya dado en la programación.

2.3.2.3 Disposición de los fines de carrera

Se dispone de un total de 5 fines de carrera distribuidos de la siguiente manera:

- Dos fines de carrera por cada eje de desplazamiento X, Y, dispuestos uno a cada extremo del eje, esto es para indicar su posición máxima y mínima.
- Un único fin de carrera en el desplazamiento Z. Éste permite limitar el ascenso del sistema a lo largo del eje Z. La limitación del descenso está dada por el microcontrolador.

La figura 2.7 muestra el tipo de fin de carrera empleado y su forma de conexión.

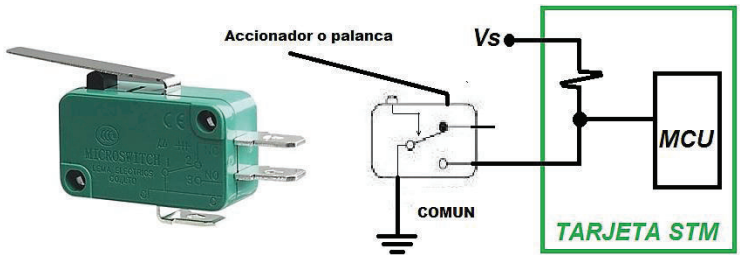


Figura 2.7 – Fin de carrera de tipo palanca con muelle, configurado como N.O.

El terminal COM (común) de cada fin de carrera va conectado a tierra, en tanto que, el segundo terminal seleccionado para configurar su comportamiento inicial, se conecta a cada uno de los respectivos pines del microcontrolador que anteriormente fueron designados como entradas para estos elementos. Se tiene entonces, un total de 5 pines del microcontrolador que son usados para la conexión del terminal N.O. de cada fin de carrera.

2.3.3 BLOQUE DE SENSOR INFRARROJO [32] [33]

El bloque de sensor infrarrojo comprende dos circuitos que son un emisor y un receptor de luz infrarroja. Estos 2 circuitos están eléctricamente aislados y dispuestos en los dedos de la pinza de tal manera que la única comunicación entre ambos sea la luz infrarroja emitida por el LED. Por tanto, el elemento emisor debe estar alineado con el elemento receptor de luz infrarroja. Se describe

a continuación cada uno de los 2 elementos que conforman tanto el circuito emisor como el circuito receptor y la forma de conexión de ambos.

2.3.3.1 El Diodo Emisor de Luz Infrarroja (IRLED)

A diferencia de los diodos convencionales, el Diodo Emisor de Luz Infrarroja, IRLED (*lnfraRed Light Emitting Diode*), por sus siglas en inglés, emite un tipo de radiación electromagnética llamada infrarroja, que es invisible para el ojo humano porque su longitud de onda es mayor a la del espectro visible. Su apariencia es muy similar a la de un diodo convencional. Viene en un encapsulado azul transparente de 3mm de diámetro, ideal para adaptarlo a uno de los orificios de los dedos de la pinza. Tiene una caída de tensión típica de 1.2V en polarización directa cuando está en conducción y un máximo de 1.5V.

2.3.3.2 Conexión del IRLED

Tal y como se observa en la figura 2.6, el ánodo del diodo IRLED va conectado a una resistencia en serie que limita el valor de la corriente a un valor adecuado para iluminar el LED. El valor de esta resistencia está comprendido entre 220Ω y 330Ω para garantizar la circulación de la corriente que especifica el fabricante. Para nuestro caso, se ha elegido 220Ω que limita la corriente a un valor de 10mA, valor especificado por el fabricante y que proporciona una luminosidad suficiente para su aplicación en el prototipo. Si se desea reducir la intensidad de luz infrarroja, se debe aumentar el valor la resistencia a 330Ω, consiguiendo que pase menos corriente y por ende disminuya la intensidad de luz emitida.
El diodo IRLED junto con la resistencia, constituyen el circuito emisor encargado de enviar la señal luminosa al fototransistor. El circuito está alimentado por una fuente de 3 [V] que proporciona el regulador de la placa STM32F4 DISCOVERY.

2.3.3.3 El Fototransistor [14]

El circuito emisor se complementa con el circuito receptor, que consta de un fototransistor y una resistencia que regula el paso de la corriente de colector.
El elemento fototransistor a utilizarse es del tipo NPN. Éste se diferencia de un transistor convencional, porque su base ha sido sustituida por una fina capa de

silicio fotosensible para permitir que la corriente de base sea producida y controlada por la cantidad de luz que incida sobre ésta. La unión que se expone a la luz incidente es la unión *PN colector-base* o unión colectora. El fototransistor es un dispositivo comúnmente sensible a la luz que se encuentra dentro de la parte roja e infrarroja del espectro visible, como lo es, la luz infrarroja. Este dispositivo puede tener una configuración de 2 o 3 terminales de conexión. Para el prototipo, es suficiente el de dos terminales. Aunque su apariencia puede ser igual a la de un diodo, difiere de éste por el color del encapsulado que es negro.

2.3.3.4 Conexión del Fototransistor

Cuando el IRLED emite radiación, empieza a circular corriente por el colector del fototransistor. Para tener una señal de 3V a la salida del emisor, se requiere de una resistencia de 33KΩ conectada en el colector y alimentado por un voltaje de 3V que provee la misma placa STM32F4 DISCOVERY en uno de sus pines marcados con este valor. La señal de 3V que se tiene en el emisor, se conecta a uno de los pines disponibles de la placa, destinándose un único pin.

2.3.3.5 Funcionamiento del bloque sensor infrarrojo

El circuito emisor está alimentado por una fuente de 3[V] provenientes de uno de los pines de la placa microcontroladora. Esto hace que por el IRLED, circule una corriente de unos 10mA que lo hacen funcionar. La luz emitida por el LED ilumina la unión colectora del transistor, generándose una corriente en la base que es directamente proporcional a la intensidad de luz. Esta acción, produce una corriente en el colector, que varía proporcionalmente con el aumento de la corriente en la base, es decir, con un aumento de la incidencia de luz sobre la unión colectora. Cuando el fototransistor entra en conducción, este se comporta como un interruptor cerrado, permitiendo el envío de una señal continua de 3[V] hacia uno de los pines que sobresalen de la placa STM32F4 DISCOVERY. La señal recibida permite controlar el cierre de una pinza a través de un servomotor acoplado al eje del pivote de la pinza.

2.3.4 BLOQUE DEL SERVOMOTOR Y LA PINZA

La aplicación que se da al sensor infrarrojo es, detectar la presencia de un objeto entre los dedos de la pinza que es donde se encuentran alineados el emisor por un lado y el receptor por el otro. La luz emitida por el IRLED mantiene al fototransistor en conducción hasta que el haz de luz infrarroja sea interrumpido por la presencia de un objeto que bloquea el paso de luz hacia el fototransistor. Cuando el microcontrolador recibe esta señal de interrupción, inmediatamente da la orden al servomotor para que ejecute la acción de cierre de la pinza a través del envío de una serie de pulsos PWM, consiguiendo así, sujetar el objeto que detectó entre los dedos. Para lograr ese cometido, es necesario que tanto la pinza como el servomotor, cumplan ciertos parámetros como son: el desplazamiento lineal de los dedos para la pinza y el ángulo que cubre el servomotor en el desplazamiento de los mismos.

2.3.4.1 Selección de la pinza

La pinza constituye el elemento que reemplaza al gancho de sujeción. Se suele denominar pinza a los elementos de sujeción que se emplean para agarrar y sostener objetos. Se utilizan para tomar un objeto y sujetarlo durante el ciclo de trabajo. De entre la gran variedad de pinzas se distinguen las que utilizan dispositivos de agarre mecánico y las que utilizan algún otro tipo de dispositivo. Para el desarrollo del proyecto, utilizaremos una pinza que permita un desplazamiento lineal y paralelo de los dedos, entre sí, para la apertura y cierre. Esta característica la cumple la pinza de tipo lineal, aunque existe otra de tipo pivotante, donde los dedos de la pinza giran en relación con los puntos fijos del pivote. La característica mecánica de ésta, nos imposibilita adaptar el sensor infrarrojo entre los dedos de la pinza. Por tal razón se utiliza la pinza de tipo lineal.

2.3.4.1.1 Adaptación del sensor infrarrojo

Los 2 elementos que componen el sensor infrarrojo, IRLED y fototransistor, se adaptan uno a cada ranura del dedo de la pinza. La separación entre dedos es de 5cm, por lo que nuestro sensor está calibrado para cubrir esta distancia. Los elementos deben estar totalmente alineados para permitir la correcta emisión de

luz del IRLED al fototransistor. El terminal del colector del fototransistor se conecta al pin del microcontrolador, destinándose un solo pin para este bloque.

2.3.4.2 Selección del servomotor

El elemento motriz de la pinza es un pequeño servomotor, comúnmente utilizado en radiocontrol. Se trata de un motor D.C. con un circuito electrónico y cuyo eje se acopla un sistema de engranes para reducir la velocidad y aumentar el torque. Este servomotor cumple las características que lo hacen idóneo para adaptarlo a la pinza de nuestro prototipo, como son: buen par de salida, potencia suficiente para trasladar objetos, baja inercia. Permite una fácil fijación a una estructura plana al ir dentro de una carcasa rectangular de plástico con soportes para fijar los tornillos. El eje del motor encaja perfectamente en el eje del pivote de la pinza, permitiendo la apertura y cierre cuando el motor entre en funcionamiento. Otra de las características de este motor, es que no gira los 360°, pues su giro está limitado internamente por un tope mecánico, esto hace que el eje del motor se limite a girar hasta 90°, 180° o el número de grados que especifique el fabricante. Para este prototipo, el servomotor a utilizar es el TowerPro MG995. Las especificaciones más relevantes dadas por el fabricante para este motor son:

- Peso: 55 g
- Dimensión: 40.7 x 19.7 x 42.9 mm.
- Torque: 8.5 kgf x cm (4.8 V), 10 kgf x cm (6 V)
- Velocidad de operación: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Voltaje de operación: 4.8 V a 7.2 V

La unidad viene con 30cm de cable trifilar y conector hembra tipo 'S' de 3 pines que se ajusta a la mayoría de receptores. Este servomotor puede girar aproximadamente 120° (60° en cada dirección).

2.3.4.2.1 Conexión del servomotor

La unidad servomotora dispone de 3 terminales de diferente color:

- Rojo: Corresponde al terminal de alimentación positiva, 5V valor que está dentro del rango establecido por el fabricante.
- Café: Corresponde al terminal de conexión a tierra.

- Naranja: Va conectado a uno de los pines de la tarjeta microcontroladora, corresponde al terminal por donde se envía la señal de pulsos PWM desde el microcontrolador hacia el servomotor para hacerlo funcionar.

2.3.5 BLOQUE DE DRIVER´S PARA CONTROL DE MOTORES PAP

La forma más sencilla de controlar un motor PAP es mediante la conmutación electrónica de unos circuitos realizados básicamente con transistores que reciben el nombre de Puente en H. El problema de este tipo de circuitos, es la caída de tensión real que hay en los transistores y que se debe compensar con la tensión de alimentación. Para evitar estos problemas se hace uso de un circuito integrado que incluye en su interior los drivers para el control. [1]
A la vez, para realizar el control de los motores paso a paso es necesario generar una secuencia determinada de pulsos que deben ser capaces de entregar la corriente necesaria para que las bobinas del motor se exciten. Vamos a centrarnos en el control de los motores PAP utilizando nuestro microcontrolador STM32F407VGT6. Como el microcontrolador no es capaz de generar la corriente suficiente para excitar las bobinas del motor PAP, utilizaremos los drivers contenidos en el integrado L298N, ya que la corriente máxima que puede proporcionar cualquier línea de salida de nuestro microcontrolador está limitada a 25mA como máximo. Esta corriente es demasiado pobre para alimentar a los motores PAP directamente, por ello se hace necesaria la utilización de driver´s de control.

2.3.5.1 El driver L298 [34]

Es un driver de puente dual de 4 canales, capaz de proporcionar una corriente de salida de hasta 2A por canal. Cada canal está diseñado para ser controlado por señales o niveles lógicos de entrada TTL estándar. Para cada pareja de canales se proporcionan 2 entradas de habilitación que activan o desactivan las salidas de los mismos independientemente de las señales de entrada. Este driver está diseñado especialmente para manejar cargas inductivas. La figura 2.8 muestra el diagrama de bloques y la numeración que se da a cada patilla del L298N en el encapsulado de 15 pines. La tabla 2.4 describe la función de cada patilla.

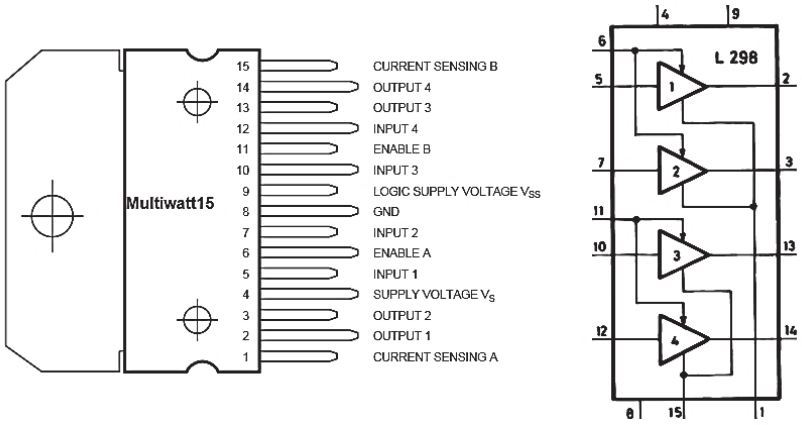


Figura 2.8 – Numeración de pines del L298N

Tabla 2.4 – Descripción de pines del L298N

PIN	NOMBRE	DESCRIPCIÓN
1;15	Sense A; Sense B	Entre estos pines y tierra se conecta la resistencia de detección para controlar la corriente de la carga.
2;3	Out 1; Out 2	Salidas del Puente A. La corriente que fluye a través de la carga conectada entre estos dos pines se monitorea en el pin 1.
4	Vs	Tensión de alimentación para las etapas finales de potencia (Alimentación de las cargas). Un condensador de 100nF no inductivo debe ser conectado entre este pin y tierra.
5;7	Input 1; Input 2	Entradas TTL del puente A
6;11	Enable A; Enable B	Entradas de habilitación TTL: Habilita o deshabilita el puente A o el puente B.
8	GND	Conexión a tierra
9	Vss	Tensión de alimentación para los bloques lógicos. Un condensador de 100nF debe ser conectado entre este pin y tierra.
10;12	Input 3; Input 4	Entradas TTL del puente B
13;14	Out 3; Out 4	Salidas del Puente B. La corriente que fluye a través de la carga conectada entre estos dos pines se supervisa en el pin 15.

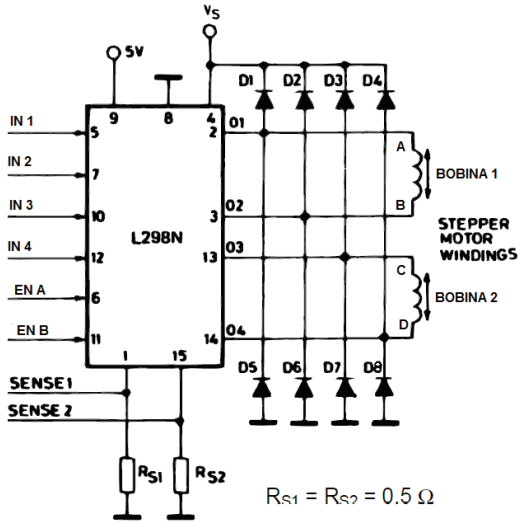


Figura 2.9 – Conexión de los diodos en paralelo con las bobinas del motor PAP

El driver dispone de un pin (Vs) para alimentar a la carga que se está controlando de modo que dicha alimentación es independiente de la lógica de control.
Es indispensable conectar un diodo en paralelo con cada devanado del motor, tal como lo muestra la figura 2.9. Esto, como medida de protección frente a los picos de la fuerza contra-electromotriz producidos por la carga inductiva de la bobina en el momento de la conmutación.

2.3.5.2 Módulo L298N [35] [36]

Este módulo está basado en el integrado L298N y permite controlar dos motores de corriente continua o un motor PAP bipolar de hasta 2A. La figura 2.10, muestra en detalle el módulo L298N y sus partes principales

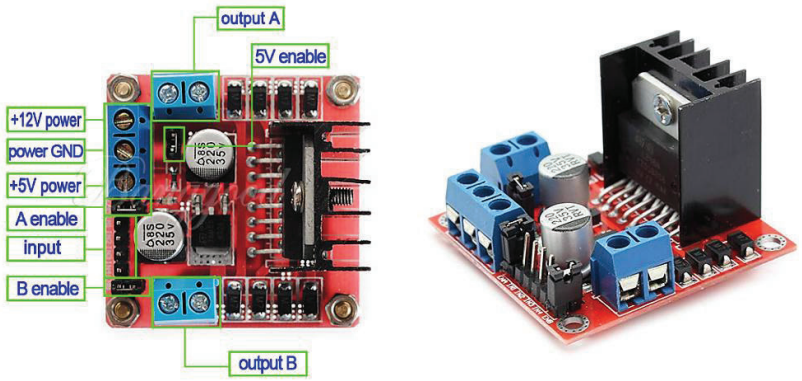


Figura 2.10 – Vista detallada del módulo L298N

El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales, entre ellos, diodos de protección y un regulador LM7805 que suministra 5V a la parte lógica del integrado L298N. Cuenta con jumpers de selección para habilitar cada una de las salidas A y B del módulo mediante los pines de habilitación ENA y ENB respectivamente. La salida **A**, está conformada por OUT1 y OUT2 y la salida **B**, por OUT3 y OUT4. En la parte inferior se encuentran los pines de control del módulo, marcados como IN1, IN2, IN3 e IN4. La figura 2.11 muestra esta nomenclatura marcada en el módulo.

2.3.5.3 Configuración y alimentación del módulo L298N

El módulo puede ser alimentado de 2 maneras gracias al regulador integrado LM7805. Cuando el jumper de selección de 5V se encuentra activo, el módulo

permite una alimentación de entre 5V a 12V DC por la bornera marcada como +12. Como el regulador se encuentra activo, la bornera de salida marcada como +5V tiene un voltaje de 5V DC. Este voltaje puede ser usado para alimentar al microcontrolador, siempre que el consumo no supere los 500mA.

Cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC por la bornera marcada como +12. Como el regulador no está funcionando, se debe conectar la bornera de +5V a una tensión de 5V para alimentar la parte lógica del L298N. En la figura 2.11 se aprecia la disposición de los jumpers que permiten el funcionamiento del módulo.

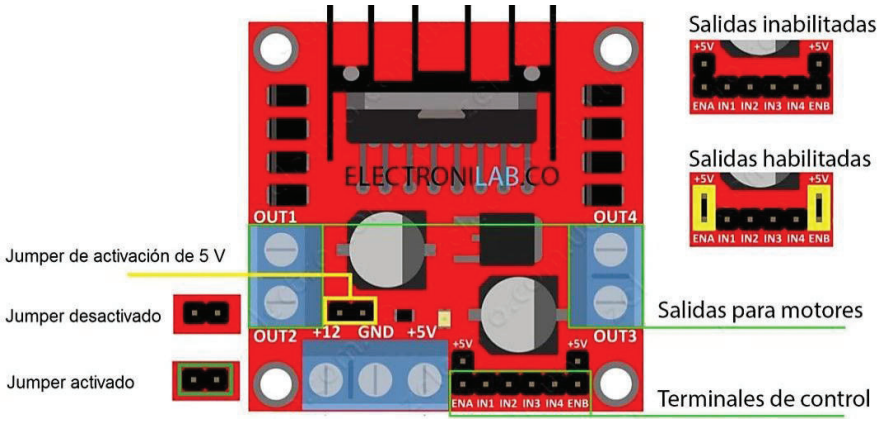


Figura 2.11 – Disposicion de jumpers en el módulo L298N [36]

En la misma figura 2.11 se muestra la disposición que deben tener los puentes para habilitar los dos grupos de salidas A y B. Los pines del módulo marcados como IN1, IN2, IN3 e IN4 se conectan al microcontrolador.

Para el desarrollo del proyecto se utilizan 4 módulos L298N, uno para cada motor PAP. Nótese que cada módulo dispone de 4 borneras de salida dispuestas en 2 grupos. Cada grupo corresponde a los terminales de 1 bobina de cada motor. Nuestros motores PAP disponen de 2 bobinas, es decir 4 terminales por lo que se habilitan los dos grupos de salida mediante los respectivos jumpers. Para todos los módulos se habilita el jumper de activación de 5V, pues los 4 motores trabajan en el rango de 12V que admite esta configuración. No obstante, cada módulo se alimenta con el voltaje que admite el motor que controla. Ahora bien, lo único que nos queda es conectar los terminales de cada motor a las borneras de salida de su respectivo módulo. El único inconveniente es identificar los terminales de cada bobina. La solución se da en el siguiente apartado.

2.3.5.4 Identificación de terminales de los motores

La primera dificultad que se presenta es poder identificar las bobinas internas con los terminales del motor. Para ello se debe tener en cuenta el número de hilos que dispone el motor PAP, esto nos ayuda a identificar en primer lugar el tipo de motor. Para nuestro caso se dispone de dos tipos de motores que se diferencian por el número de hilos que salen de su estructura: Un motor de tipo unipolar (6 hilos) y 3 de tipo bipolar (4 hilos).

Luego de haber identificado el tipo de motor, el segundo paso es identificar la disposición de las bobinas. Para ello haremos uso del multímetro y procedemos a medir continuidad entre los terminales. Para nuestro motor bipolar que tiene 4 hilos, tendremos dos grupos de 2 hilos que indiquen continuidad. Cada par de hilos pertenece a los terminales de una bobina. Se puede identificar a cada lado de bobina como: A-B para la bobina 1; C-D para la bobina 2.

El motor unipolar de 6 hilos, está provisto de un doble arrollamiento con toma central por lo que se tiene 2 grupos de 3 hilos que indiquen continuidad. Se identifica primero la toma central de cada grupo, tomando arbitrariamente de un único grupo un terminal como común y en base a éste, se mide la resistencia que hay entre los 2 terminales restantes. Se habrá identificado la toma central cuando la resistencia entre éste y los otros 2 terminales sea igual, por lo que se debe conmutar varias veces a otro punto común hasta obtener este resultado. Los 2 terminales que difieren del punto común, corresponden a los terminales de la bobina 1 y la resistencia entre éstos, será aproximadamente el doble de la resistencia entre cualquiera de los 2 terminales a la toma central. El mismo proceso se lo realiza con el otro grupo para identificar los terminales de la bobina 2 y su respectiva toma central. Las tomas centrales de cada grupo se descartan en la conexión pues se configura al motor unipolar para que trabaje como bipolar sin que ello afecte su característica. Al igual que el caso anterior, se identifica a los terminales de las bobinas como: A-B para la bobina 1; C-D para la bobina 2. Nótese que en ninguno de los 2 casos se toma en cuenta el orden de los terminales de cada bobina para el etiquetado, pues si el motor no gira o se manifiesta con un leve zumbido, solo es necesario invertir la conexión de los terminales de uno de los 2 grupos.

2.3.5.5 Conexión de los motores al módulo L298N

Una vez identificado los bornes de cada motor, solamente nos queda conectar cada terminal a su respectiva bornera. Para ello basta seguir el diagrama de conexiones de la figura 2.6. Se debe tomar en cuenta que no todos los motores PAP en condiciones iniciales giran en el mismo sentido, pues como se había indicado en apartados anteriores, para el desplazamiento del puente (eje Y) se utiliza dos motores PAP, dispuestos uno a cada extremo de la viga. Para conseguir el desplazamiento sincronizado del puente, uno de los 2 motores debe girar en sentido contrario respecto del otro. La inversión de giro se consigue intercalando el orden de conexión de los terminales de control que van de la placa STM32F4 DISCOVERY al módulo L298N.

2.3.6 BLOQUE DEL MÓDULO BLUETOOTH

El módulo Bluetooth es el encargado de mantener la comunicación entre la Tablet y el prototipo, a través del microcontrolador, permitiendo la recepción de información proveniente de la Tablet, desde donde se envía la señal de control del prototipo. La información se procesa y se envía al microcontrolador.

2.3.6.1 Módulos Bluetooth HC-06 y HC-05 [37] [38]

Comercialmente existen 2 tipos de módulos Bluetooth, el HC-05 y el HC-06. Son dispositivos relativamente económicos comúnmente usados para aplicaciones con microcontroladores. Ambos módulos disponen de pines que permite insertarlos en un protoboard y cablearlo directamente a cualquier microcontrolador, incluso sin realizar soldaduras. Aunque aparentemente iguales, ambos módulos difieren por el número de pines que salen de su hardware. El módulo HC-05 dispone de 6 pines (TX, RX, VCC, GND, EN, STATE), 2 pines más que el módulo HC-06, en el que solo encontramos 4 pines (TX, RX, VCC, GND). Otra de las diferencias entre estos 2 módulos es la forma en que se identifican con otros dispositivos Bluetooth: El HC-05 se identifica como “HC-05″, mientras que el HC-06 se identifica como “Linvor” o “HC-06″. La figura 2.12 ilustra la diferencia física entre los 2 módulos.

Una de las diferencias más importante a tomar en cuenta, es que el módulo HC-06, trabaja únicamente en modo esclavo, esto quiere decir que el módulo siempre queda a la espera de vincularse con algún equipo o dispositivo cercano.

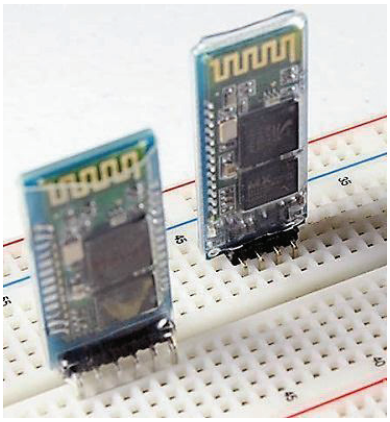


Figura 2.12 – Comparación entre el módulo HC-05 (izqu.) y HC-06 (der) [39]

2.3.6.2 Módulo Bluetooth HC-05 [40]

El módulo Bluetooth HC-05 ofrece una mejor relación de precio y características, ya que se lo puede configurar como **Maestro** o como **Esclavo**, lo que significa que además de recibir conexiones desde una PC o Tablet, también es capaz de generar conexiones hacia otros dispositivos Bluetooth. Esto nos permite por ejemplo, conectar dos módulos Bluetooth y formar una conexión punto a punto para transmitir datos entre dos microcontroladores o dispositivos.

El módulo Bluetooth HC-05, utiliza el protocolo UART RS 232 serial siendo ideal para aplicaciones inalámbricas y de fácil implementación con microcontroladores.

Viene incorporado con 6 pines (TX, RX, VCC, GND, EN, STATE), de fácil acceso para uso en protoboard. Posee un regulador interno que permite su alimentación de 3.6 a 6V. Otras características relevantes dadas por el fabricante son:

- Voltaje de alimentación: 3.6VDC – 6VDC.
- Voltaje de operación: 3.3VDC.
- Baud rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Baud rate por defecto: 9600
- Tamaño: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)

- Corriente en modo de espera (*antes de enlazarse a un dispositivo*): 30 - 40mA
- Corriente de operación (*cuando está enlazado a un dispositivo*): 8mA

La figura 2.13 muestra el aspecto externo del módulo HC-05.

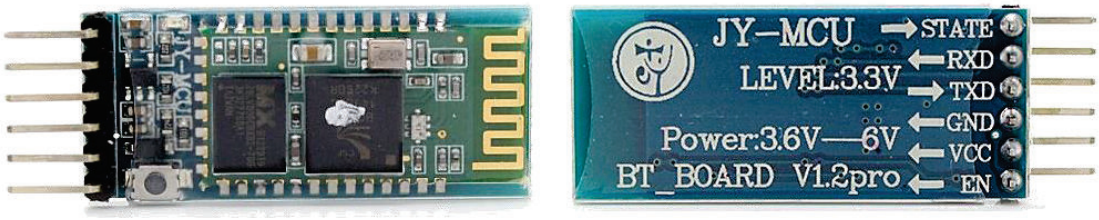


Figura 2.13 – Aspecto externo y disposición de pines del módulo Bluetooth

La comunicación entre dos dispositivos configurados como maestro - maestro o esclavo - esclavo no es posible. La PC, Tablet o cualquier otro dispositivo móvil están siempre configurados como maestros por lo que es necesario que el módulo Bluetooth sea configurado como esclavo para establecer un vínculo de comunicación con una Tablet que es el dispositivo móvil a usar en el proyecto.
El módulo HC-05 viene configurado de fábrica como esclavo, aunque como ya se mencionó se puede cambiar esta configuración a modo maestro accediendo a su firmware (programación interna del módulo) en la que también, se pueden modificar otros parámetros como: nombre del dispositivo, password, velocidad de transmisión que por default es 9600 baudios, etc. Vamos a aprovechar los parámetros de configuración inicial que tiene el dispositivo por default, es decir, su configuración en modo esclavo y la velocidad de transmisión de datos a 9600 baudios, que son suficientes para el desarrollo de nuestro proyecto, esto implica que no tendremos que acceder al firmware del módulo para configurarlo.

2.3.6.3 Conexión del módulo Bluetooth HC-05

La alimentación del módulo se la realiza con un voltaje de 5VDC, luego el módulo se encarga de regular este voltaje a 3.3VDC. Para la conexión del módulo Bluetooth al microcontrolador, se debe tener muy en cuenta la disposición de pines de cada uno. La conexión de los pines Tx y Rx del módulo Bluetooth va en conexión cruzada con respecto a los pines Tx y Rx del microcontrolador. Esto es: el pin Tx del módulo Bluetooth con el pin Rx del micro y el pin Rx del módulo

Bluetooth con el pin Tx del micro. Este es un detalle que no se especifica en la hoja de datos del módulo Bluetooth. Entonces lo que para el fabricante es Tx, para nosotros es la salida de datos recibidos por parte del módulo, es decir, lo que el módulo recibe, de lo que otro dispositivo le transmite. En consecuencia, el pin Rx es la salida de datos que transmitiremos y que el otro dispositivo recibe. Esto es lo más importante a tener en cuenta.

2.3.6.4 Operación del módulo Bluetooth HC-05

Una vez alimentado el módulo y conectado al microcontrolador, inmediatamente el diodo LED que viene integrado al módulo empieza a parpadear, anunciando estar listo y a la espera de vincularse a algún dispositivo cercano. En esta situación, el módulo HC-05 posee su más alto consumo energético que oscila entre 30 y 40mA. Luego de vincularse a cualquier dispositivo, la corriente de operación desciende aproximadamente a 8mA. Por lo general, al momento de establecer un enlace por primera vez, con un ordenador o con un dispositivo móvil (teléfono, Tablet), tenemos la posibilidad de observar y anotar la dirección MAC del módulo. El módulo Bluetooth indicó la dirección 98:D3:31:90:09:AC, para luego pasar a "auto-nombrarse" HC-05. Este es el momento indicado para saber con qué módulo se está trabajando, aunque también existen programas y aplicaciones que ayudan a identificar estos parámetros de cualquier módulo como lo es el programa AMARINO.

El diodo LED indicador de estado del módulo posee una característica particular. Cuando el módulo está energizado pero no se ha conectado a algún dispositivo, el diodo LED se enciende intermitentemente a una frecuencia de 1Hz.

Cuando el módulo se ha vinculado a algún dispositivo, el LED parpadea 2 veces por segundo y se mantiene apagado 3s. Este es un indicador de que la comunicación se ha establecido con éxito entre el módulo HC-05 y la Tablet. Esta última acción del LED se repite de manera continua mientras dure la conexión. Si se pierde la conexión, el diodo LED vuelve a comportarse como en el primer caso. [41]

2.4 PLACA DE CIRCUITO IMPRESO DEL PROTOTIPO

Los bloques detallados en las secciones anteriores se integran en conjunto en una sola placa de circuito impreso. En ésta, se encuentran disponibles todas las borneras tanto de entrada como de salida, así como los zócalos necesarios donde se introducen los pines de la tarjeta STM32F4 DISCOVERY.

En la figura 2.14, se puede observar que se utilizan elementos como pines macho de conexión, borneras y zócalos. Los pines macho de conexión se emplean para conectar las entradas de los elementos hacia la tarjeta STM32F4 DISCOVERY con conectores hembra. Se dispone de 3 borneras. Una bornera marcada como 5VDC sirve para alimentar con 5VDC a la placa STM32F4 DISCOVERY y a 3 motores PAP (2 en Y, 1 en X), otra bornera marcada con 12VDC permite alimentar con este voltaje al motor PAP dispuesto para el desplazamiento en el eje Z. En la misma figura, se observa la disposición del Buzzer ya soldado en la placa.

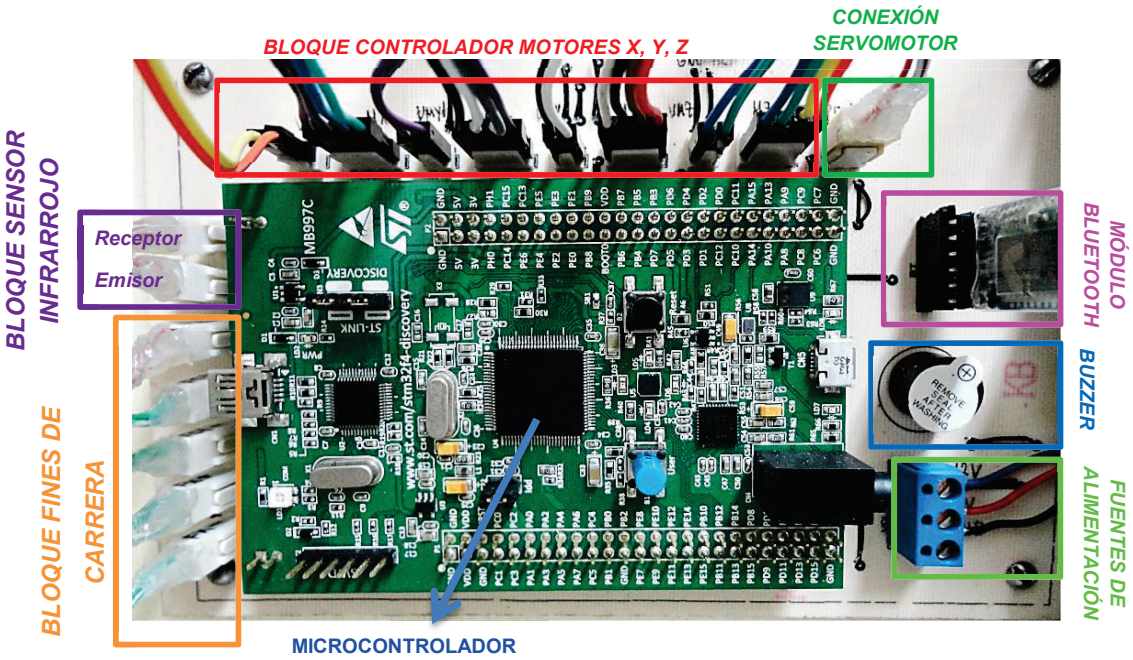


Figura 2.14 – Disposición de elementos en la placa de circuito impreso

CAPÍTULO 3

DISEÑO DEL SOFTWARE Y PROGRAMACIÓN

3.1 ANTECEDENTES

Dentro del diseño del software y la lógica de control en la que se ha basado la programación para este proyecto, se ha utilizado la herramienta SIMULINK de MATLAB; y para el caso del desarrollo de la interfaz gráfica de la aplicación para el control inalámbrico mediante la Tablet, se ha empleado el software ANDROID STUDIO, que nos permite diseñar y desarrollar dicha interfaz desde la computadora. Las herramientas empleadas en la programación que han sido mencionadas anteriormente se detallan en este capítulo.

3.2 SOFTWARE DE PROGRAMACIÓN EN SIMULINK

La forma de programación empleando SIMULINK, se realiza básicamente mediante un diagrama de bloques, subsistemas, variables y funciones, las cuales se relacionan directa e indirectamente en la programación. Debido a la extensión del programa, se ha dividido al software de programación en bloques más pequeños, que nos permitan una breve comprensión de los procesos que se han de realizar en casos de ciclos de lazo cerrado y secuencias, con el fin de cumplir la lógica de control del prototipo esperada. El software de programación completo para el prototipo se lo observa en el ANEXO A1.

En el desarrollo de la programación con SIMULINK, es posible realizar operaciones lógicas y matemáticas, con el fin de dar solución a ciertas funciones e implementaciones que se requieren para llegar a la lógica de control esperada en el prototipo.

3.3 LÓGICA DE CONTROL Y FUNCIONAMIENTO

Para fines didácticos, el control de posicionamiento para el prototipo se lo realiza de dos formas: posicionamiento manual y posicionamiento automático. Se han determinado ciertas condiciones que el puente grúa debe cumplir con el único propósito de mantener la seguridad en los procesos. Se pueden nombrar algunos de estos procesos, como son: la elevación de la carga, la apertura de la pinza, el transporte de la carga y demás condiciones, que generalmente se cumplen en sistemas de puentes grúa empleados en la industria.

Adicionalmente, se determina mediante programación un espacio de seguridad, denominado HOME, que representa el punto inicial y/o punto de referencia desde el cual, el puente grúa empieza a moverse. Se considera también a dicho punto, como el sitio de seguridad donde permanece suspendida la pinza cuando no se la utiliza.

Las señales que proporcionan los fines de carrera, sirven para detener el movimiento de los motores en la línea de desplazamiento (X, Y, Z). Mientras que la señal del sensor infrarrojo nos permite detectar la presencia de un objeto para que la pinza se cierre y tome el objeto. Otra señal importante, es la del Buzzer, el cual alerta a los usuarios que el puente está en movimiento.

A continuación, se detallan las condiciones que el puente grúa debe cumplir tanto para el modo de posicionamiento manual y automático.

3.3.1 CONTROL DE POSICIONAMIENTO EN MODO MANUAL

El posicionamiento del puente grúa en modo manual cumple las siguientes condiciones:

- Todas las líneas de desplazamiento (ejes X, Y, Z) funcionan únicamente al mantenerse presionado el botón respectivo. Esto es, subir y bajar la pinza, desplazar el carro de izquierda a derecha y, viceversa; mover el puente hacia adelante y hacia atrás.
- Se pueden ejecutar varias acciones de desplazamiento a la vez, siempre y cuando cada acción corresponda a distintas líneas de desplazamiento, es

decir, se puede desplazar el puente hacia adelante y a la vez, mover el carro a la izquierda.

- No se permite el desplazamiento de dos acciones que correspondan a la misma línea de desplazamiento.
- La apertura de la pinza es únicamente de forma manual, y la acción de apertura se lleva a cabo 4 segundos después de mantener presionado el botón de apertura.
- Se dispone de un botón HOME dispuesto en el control y que permite el posicionamiento automático de la pinza al espacio de seguridad.
- Si la pinza permanece por 2 minutos sin recibir alguna instrucción de movimiento, retorna automáticamente desde la última posición en la que se encuentra a su espacio de seguridad (HOME).
- Cuando la pinza se desplace sin carga ésta debe desplazarse siempre suspendida y cerrada.

3.3.2 CONTROL DE POSICIONAMIENTO EN MODO AUTOMÁTICO

El posicionamiento del puente grúa en modo automático cumple las siguientes condiciones:

- En el espacio determinado como de seguridad (HOME), la pinza se mantiene suspendida y cerrada.
- Cuando la pinza se encuentre en el punto HOME, se ingresan coordenadas del punto donde está el objeto y la pinza al estar sin carga, se traslada a dicho punto a mayor velocidad.
- La pinza debe desplazarse suspendida y cerrada cuando haya recorrido las posiciones (X, Y), y se abre al llegar a éste punto antes de descender por la línea Z para detectar el objeto y recogerlo.
- Un sensor infrarrojo ubicado entre los dedos de la pinza, permite realizar el cierre automático de la misma, cuando haya detectado la presencia de un objeto.
- Cuando la pinza se encuentre trasladando un objeto de un lugar a otro, su recorrido debe realizarse a menor velocidad, por motivos de seguridad.

- Con la pinza en cualquier posición (después de haber trasladado un objeto) existe la posibilidad de dar una nueva posición a la cual va a dirigirse desde ese punto.
- Después de que la pinza haya realizado su trabajo y se encuentre suspendida, debe regresar al espacio considerado como de seguridad para lo cual se tiene dos opciones:
    - Si la pinza permanece por 2 minutos sin utilizarse automáticamente se cierra y retorna desde la última posición en la que se encuentra a su espacio de seguridad (HOME).
    - Al oprimir el botón HOME que se encuentra en el control remoto. Esta acción comprende la suspensión automática de la pinza. Y sólo cuando esté totalmente suspendida, ésta se cierra para continuar con su posición de retorno a HOME.

3.4 DESARROLLO DE LA PROGRAMACIÓN

El diagrama de bloques realizado mediante el software de programación SIMULINK, está conformado por varios subsistemas, funciones, compuertas lógicas, contadores de tiempo, memorias, variables y máquinas de estado. Todos estos elementos deben mantener la lógica de control para cumplir el funcionamiento según las condiciones anteriormente explicadas para el prototipo. Mediante un diagrama de Stateflow (Diagrama de estado) propia del software, es posible desarrollar una interfaz gráfica y tabular para modelar la lógica de un sistema mediante máquinas de estado. En una máquina de estado, se modelan los modos de funcionamiento del sistema a modo de estados y se representa la lógica para cambiar de un modo a otro mediante transiciones y uniones. [42]

En la programación, las expresiones booleanas, se introducen entre corchetes [ ], mientras que las acciones a realizar durante una transición, si las hubiera, se indican entre llaves { }, como se muestra en la figura 3.1 [43]

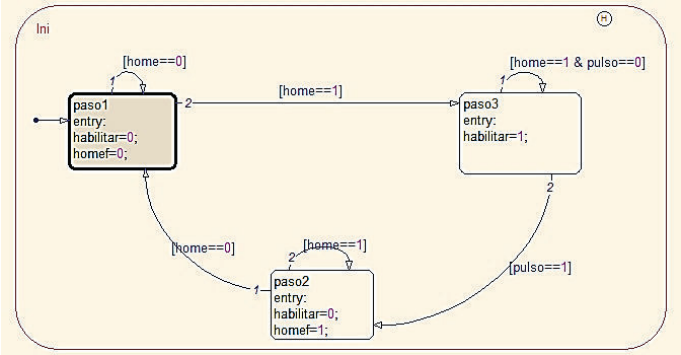


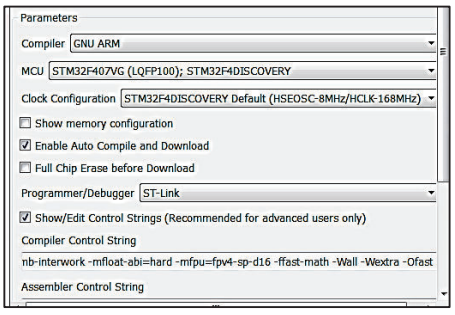
Figura 3.1 – Esquema ejemplar de una máquina de estados

3.4.1 INICIALIZACIÓN DEL PROGRAMA [26]

En la inicialización del programa, se tiene una descripción general de las variables globales, los subsistemas, las funciones y las características principales de la tarjeta STM32F4DISCOVERY.

3.4.1.1 Bloque de Caracterización

El bloque de caracterización permite configurar los parámetros y el modelo de tarjeta que se utiliza en el entorno de SIMULINK para este proyecto. Al dar clic sobre el bloque de caracterización, se obtiene un cuadro de opciones que permite modificar los parámetros de utilización de la tarjeta STM32F4 DISCOVERY, tal como se indica en la figura 3.2b



a) Bloque de Caracterización b) Vista interna

Figura 3.2 – Esquema del bloque de caracterización y parámetros internos

Los parámetros más relevantes para este bloque se han descrito en la tabla 3.1, donde se define también otros parámetros como la librería **Waijung14.05a.**

Tabla 3.1 – Parámetros del bloque Caracterización de la Tarjeta

PARÁMETRO	OPCIÓN ELEGIDA	FUNCIÓN
Compiler *(Compilador)*	GNU ARM	GNU ARM como compilador permite mantener una cadena de herramientas GNU GCC (Colección de Compiladores GNU) para la arquitectura de procesadores ARM que forman parte de la tarjeta STM32F4 DISCOVERY.
MCU *(Microcontrolador)*	STM32F407VG(LQFP100): STM32F4DISCOVERY)	Nos permite determinar el microcontrolador que se va a emplear en la programación del proyecto. Para nuestro caso, se ha escogido STM32F407 que pertenece al modelo del microcontrolador de la tarjeta electrónica a utilizarse.
Clock Configuration *(Configuración* de reloj)	STM32F4DISCOVERY Default (HSE0SC-8MHz/HCLK-168MHz)	Los parámetros de frecuencia del cristal externo de 8MHz (HSEOSC) y la velocidad de reloj de 168MHz son los más adecuados para el uso de la tarjeta STM32F4 DISCOVERY, ambos valores se los utiliza por defecto.
Programmer/ Debugger *(Programador/ Depurador)*	ST- Link	Se refiere a la utilidad para descargar archivos ejecutables desde el ordenador hacia la memoria flash del microcontrolador STM32F407.

La *librería Waijung*, es un conjunto de bloques a los cuales se pueden acceder en el desarrollo de la programación y que proporcionan la lógica de control necesaria para el prototipo. Tal librería junto al generador de código de SIMULINK, RTW (Real–Time Workshop), permiten la obtención de un código de fácil comprensión que puede ser codificado, compilado y descargado automáticamente hacia el microcontrolador.

3.4.1.1.1 Instalación del software en el entorno de MATLAB

La utilización de la tarjeta STM32F4DISCOVERY, requiere que se incluyan todas las características necesarias en la programación para su posterior compilación hacia el microcontrolador. Una de ellas, es el software ST-Link que permite la compilación, depuración y transferencia del programa desde el entorno de trabajo

en la computadora hacia el microcontrolador de la tarjeta. El archivo ejecutable para instalarlo se descarga siguiendo el enlace web: http://waijung.aimagin.com/. Al ingresar a la página web, en el panel de contenidos se debe dirigir a la pestaña *Get Started* y posteriormente a la pestaña *Software Installation*.

Aquí, se encuentran los links de descarga para *ST-Link Utility* y *ST-Link/V2 USB driver*; así como, para *Waijung Blockset*. Una vez descargado todos los anteriores, se indican los pasos a seguirse:

- Instalar el archivo ejecutable a ST-Link Utility correspondiente: STM32 ST-LINK_Utility_v3.3.0.exe
- Una vez instalado el anterior, se procede a instalar: st-link_v2_usbdriver.
- Proseguir con la instalación de la librería Waijung en el entorno de Matlab. Para ello, se debe copiar la carpeta waijung14_05a (ya descomprimida) en la carpeta *documentos* de la carpeta de Matlab instalada.
- Al finalizar la copia de los archivos, continuamos y abrimos el software MATLAB. Al estar cargado completamente, en la sección Current Folder, cambiamos el directorio: documentos/matlab/waijung14_05.
- Finalmente, se procede a ejecutar el archivo: install_waijung.m dentro del software Matlab. Con esto, la librería Waijung necesaria para el desarrollo lógico de este proyecto queda instalada en el software.

3.4.1.2 Variables Globales

En el entorno del software de programación, se han declarado espacios de memoria para cada una de las variables que se van a utilizar para el control durante todo el ciclo de funcionamiento. Al ser declaradas como espacios de memoria en el microcontrolador STM32F407, se las emplea para leer y escribir datos, valores y señales en un determinado ciclo o a su vez, durante un tiempo específico, dependiendo del proceso a realizarse en la lógica de control. Son denominadas *variables globales*, debido a su relación directa con bloques y funciones generales que son empleadas para la realización de procesos.

En el ANEXO A-1, se indican todas las variables que han sido empleadas para la programación. Como se observa, en el software del programa se ha definido un nombre para cada variable, según la función que van a realizar para el control del prototipo. La tabla 3.2 detalla la función que realiza cada variable global.

Tabla 3.2 – Variables globales y su función en el programa

VARIABLE	SIGNIFICADO	FUNCIÓN
VEL	Velocidad	Determina la velocidad medida por frecuencia de pulsos para el movimiento del motor PAP.
POS X **POS Y** **POS Z**	Posición en los ejes X, Y, Z.	Corresponde a los valores de coordenadas X,Y,Z que el usuario ingresa por la interfaz de la Tablet
PASOS X **PASOS Y** **PASOS Z**	Pasos en los ejes X, Y, Z.	Representa la distancia de desplazamiento que debe cubrir el motor a lo largo de sus ejes para llegar a la posición indicada. Se expresa en pasos.
HOME	Proceso HOME (Espacio de Seguridad)	Representa una bandera que envía una señal para iniciar el proceso HOME en todos los ejes.
HOME X **HOME Y** **HOME Z**	HOME en c/u de los ejes X, Y, Z.	Indica que se ha iniciado el proceso HOME, lo cual significa que cada uno de los ejes deberá realizar un determinado proceso para finalmente llegar al espacio de seguridad (HOME).
HOME P	Home de la Pinza	Indica que el proceso que debe cumplir la pinza para ir al espacio de seguridad (HOME) se ha iniciado
HOME XF **HOME YF** **HOME ZF**	Home final en los ejes X, Y, Z.	Indica que el proceso que debe cumplir cada uno de los ejes para llegar a HOME ya ha concluido.
HOME PF	Home final de la Pinza	Indica que el proceso que debe cumplir la pinza para ir al espacio de seguridad (HOME) ha finalizado.
PIEZAREC	Pieza recoger objeto	Representa una bandera y cuya señal indica que la pinza ha tomado un objeto para proseguir a realizar la suspensión del mismo.
ANG	Ángulo de apertura de la pinza	Indica el ángulo de apertura de la pinza.
MMANU	Modo Manual/Automático	Indica la activación del modo manual o automático para el funcionamiento del puente
MOV X **MOV Y** **MOV Z**	Movimiento en ejes X, Y, Z.	Representa una bandera e indica con una señal auditiva que el puente grúa está en movimiento.
NANG	No ángulo de apertura	Resetea el conteo de los 4 segundos para abrir la pinza, si éste tiempo no se ha cumplido totalmente.
INICIAR	Inicio del proceso automático	Representa una bandera y envía una señal que permite al puente grúa dirigirse a la posición ingresada desde la Tablet en el modo automático.

3.4.1.3 Bloque de Transmisión (Comunicación UART)

El bloque *Comunicación UART* permite configurar los parámetros del módulo *UART* utilizado en el proyecto. Los comandos de funciones que son empleados para el funcionamiento del prototipo se los envían mediante un control inalámbrico implementado en la interfaz gráfica de la Tablet, hacia el módulo Bluetooth dispuesto en uno de los pines del microcontrolador STM32F407 de la tarjeta STM32F4 DISCOVERY, lo cual implica la configuración de un protocolo *UART,* que nos permita transmitir de forma serial e inalámbricamente, los datos y mandos necesarios para el movimiento y el desarrollo de la lógica de control que debe cumplir el puente grúa en nuestro proyecto.

Figura 3.3 – Bloque de Transmisión

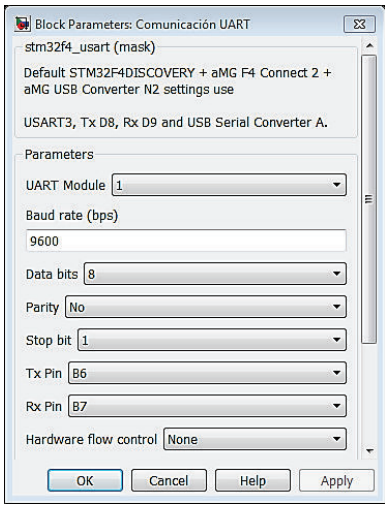


Figura 3.4 - Parámetros de configuración Bloque de Transmisión

La figura 3.3 indica el bloque de transmisión, mientras que en la figura 3.4 se observan los parámetros de configuración y que se detallan a continuación.

- ***UART Module (Módulo UART/USART):*** Permite seleccionar el número de módulos que van a emplearse en la programación

- ***Baud Rate (Velocidad de Transmisión):*** Se establece el valor de 9600 bps (baudios por segundo) acorde a las características iniciales de nuestro módulo Bluetooth HC-05.
- ***Data bits (Bit de Datos):*** Determina el número de bits que se pueden enviar de forma serial. La versión más actual de la librería Waijung en SIMULINK, permite que el bloque de comunicación UART (UART Setup) envíe únicamente 8 bits de datos.
- ***Tx (Transmisor):*** Determina el pin del microcontrolador STM32F407 que se emplea para conectar el pin de transmisión del módulo Bluetooth.
- ***Rx (Receptor):*** Determina el pin del microcontrolador STM32F407 que se emplea para conectar el pin de recepción de datos del Módulo Bluetooth.

En el Capítulo 2, se definió al pin B6 y B7 del microcontrolador como Transmisor (Tx) y Receptor (Rx) respectivamente, lo cual ha sido configurado en los parámetros del bloque de transmisión.

3.4.2 BLOQUE DE DESPLAZAMIENTO

El bloque de desplazamiento, desarrolla el movimiento de los motores PAP que van a producir el recorrido en los ejes X, Y, Z del prototipo.

Como se observa en la figura 3.5, el bloque de desplazamiento está compuesto por otros subsistemas denominados: Eje X, Eje Y, Eje Z. Cada uno de estos subsistemas, posee entradas digitales (*Digital Inputs*) y salidas digitales (*Digital Outputs*). Las *entradas digitales* (*Digital Inputs*) permiten recibir cualquier tipo de datos que son nativos del software de programación, para que dichos datos sean empleados por otros subsistemas del programa. Mientras que, las *salidas digitales (Digital Outputs),* permiten generar el lenguaje de salida digital desde el microcontrolador hacia otros dispositivos periféricos empleados en la programación.

Cada subsistema que proporciona el control del desplazamiento en los ejes (X, Y, Z) está conformado internamente por otros subsistemas y demás funciones. Las mismas que controlan, por medios de seguridad y una lógica de control, el funcionamiento en cuanto se refiere al desplazamiento del puente en todo su recorrido.

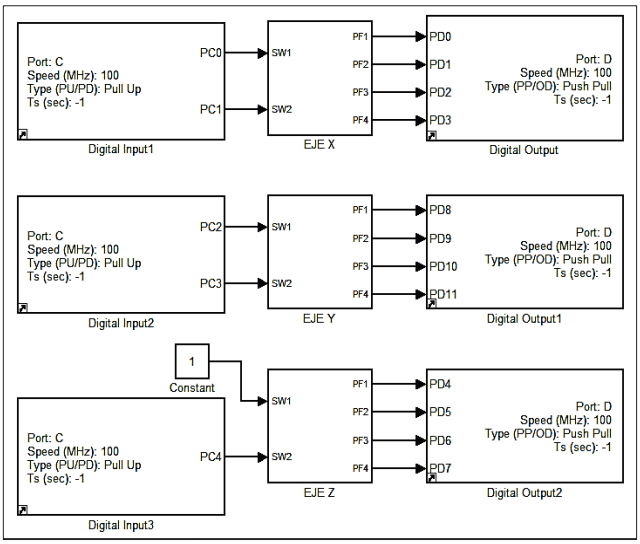


Figura 3.5 – Bloque de desplazamiento (Vista general)

Como parte fundamental para iniciar la programación para el desplazamiento del puente grúa en nuestro proyecto, se ha tomado como referencia el espacio HOME y se ha definido la disposición de los ejes como se observa en la figura 3.6.

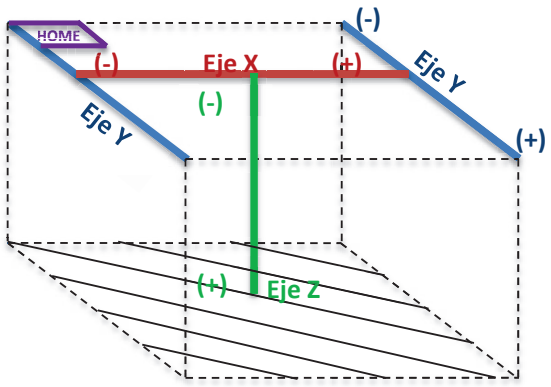
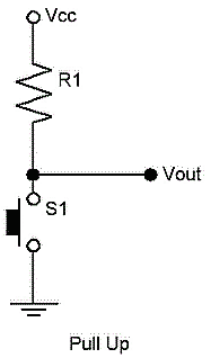


Figura 3.6 – Distribución de los Ejes X, Y, Z en la estructura

El recorrido mínimo y máximo de cada uno de los ejes, está limitado por los fines de carrera (SW1 y SW2). Para nuestro caso, todos los fines de carrera cumplen la función de un contacto normalmente abierto, sin embargo, la configuración de sus entradas al microcontrolador se comportan de la siguiente manera:



- El valor lógico del fin de carrera es 1, cuando no se ha accionado (pulsador abierto).
- El valor lógico de fin de carrera es 0, cuando se ha accionado (pulsador cerrado).

Figura 3.7 – Configuracion Pull Up

La figura 3.7 da una configuración "*Pull Up*" definida en la entrada y/o salida *digital*, y que corresponde al esquema de una resistencia conectada a VCC, un pulsador y tierra, donde se obtienen tales resultados.

El desplazamiento del motor, mantiene una relación directa con los pasos que el motor PAP debe realizar para recorrer cierta distancia ya sea ingresando los datos mediante la Tablet o a su vez, manteniendo pulsado cualquiera de los botones de control para desplazarse en cualquier sentido. El dato ingresado a través de la interfaz de la Tablet, se expresa en milímetros; pero el desplazamiento del motor, se lo realiza mediante pasos que el motor debe cumplir para llegar a dicha posición de forma precisa. Por esta razón, se hace necesaria la conversión de milímetros a pasos.

3.3.2.1 Conversión de pasos del motor PAP a milímetros

Para los ejes X, Y se han empleado 3 motores PAP con las mismas características de paso por revolución. En tanto que para el caso del eje Z, se ha empleado un motor PAP con diferentes características de paso por revolución, razón por la cual, esta sección se subdivide en dos partes.

3.4.2.1.1 Para los Ejes X, Y

La precisión de la posición en el desplazamiento del motor a lo largo del recorrido para los ejes X, Y, comprende definir una expresión que relacione los pasos del motor PAP con la distancia que abarca el diente de un piñón expresada en mm.

Para llegar a tal expresión, primero se necesita determinar el número de pasos que el motor PAP recorre para mover un diente del piñón. Para ello, se ha establecido una relación en función de los pasos que el motor PAP produce en

una revolución completa y el número de dientes en total que posee el piñón acoplado a su eje. El motor PAP empleado recorre 200 pasos, según lo que se indica en su hoja de datos. La resolución del desplazamiento angular en el número de pasos que el motor da en una revolución, se define al dividir los 360° (una revolución completa) para el número de pasos que recorre el motor en dicha revolución. Lo cual, se expresa en la fórmula 3.1.

$$\frac{360°}{200 \text{ pasos}} = 1.8 \text{ °/paso} \qquad (3.1)$$

Este valor, indica que el motor posee una resolución de 1.8°/paso, lo que quiere decir, que cada paso del motor corresponde a 1.8° de desplazamiento angular. De lo anterior, tomaremos el valor de los 200 pasos en una revolución del motor y el número de dientes que posee el piñón en toda su extensión, este valor es de 39 dientes. Dichos valores son reemplazados en la fórmula 3.2.

$$\frac{\mathbf{NPM}}{\mathbf{NDP}} = \frac{200 \text{ pasos motor}}{39 \text{ dientes del piñón}} = 5{,}128 \approx 5{,}13 \qquad (3.2)$$

Como se observa, se recorre aproximadamente 5,13 pasos para mover tan sólo un diente del piñón. Este dato nos ayuda a determinar la distancia expresada en milímetros que el motor recorre, según la relación entre la cremallera y los dientes del piñón. Esto, con el fin de obtener una conversión entre los milímetros ingresados por el usuario en la Tablet y los pasos que el motor PAP debe recorrer para alcanzar y detenerse en la posición ingresada. A continuación, la relación 3.3 determina la conversión de pasos a milímetros, donde se emplea el resultado de la expresión 3.2, y el valor de 3mm, correspondiente al espacio entre dientes del piñón o también, al espacio entre dientes de la cremallera.

$$\frac{1 \text{ diente del piñón}}{5{,}13 \text{ pasos motor}} * \frac{3\text{mm}}{1 \text{ diente piñón}} = 0{,}55 \text{ mm/paso motor} \qquad (3.3)$$

El valor obtenido, indica que el motor PAP recorre 0,55mm por cada paso, y es el referente para la conversión del valor ingresado en milímetros por la interfaz de la Tablet a pasos que debe girar el motor PAP.

3.4.2.1.2 Para el Eje Z

El motor PAP para el movimiento del eje Z, dispone de un pequeño piñón de 24 dientes. La relación necesaria para la conversión de milímetros a pasos se calcula de la misma forma que en el caso anterior, dando el siguiente valor de relación: 0,023mm/paso motor.

3.3.2.2 Sub-bloque EJE X

El sub-bloque denominado "EJE X", que se observa en la figura 3.6, permite controlar, habilitar y deshabilitar el movimiento del motor en el eje X.

En la figura 3.8 se observa el interior del sub-bloque de desplazamiento para el eje X. A continuación, se describe la función que cumple cada una de las partes constitutivas que proporcionan la lógica de control para este sub-bloque.

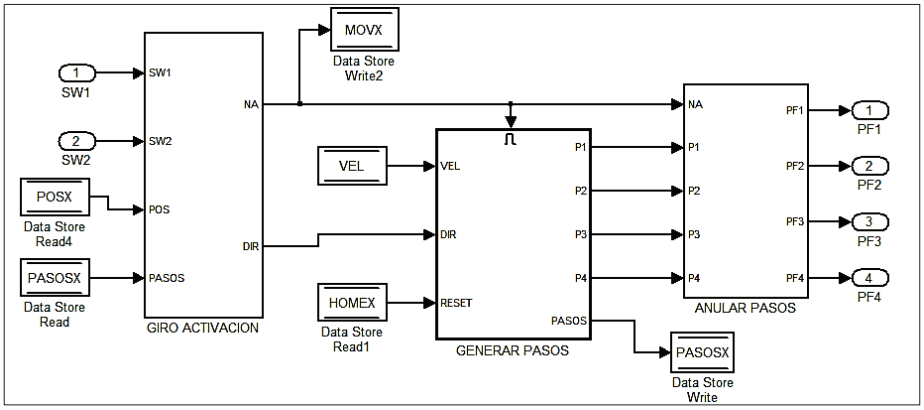


Figura 3.8 – Esquema interno del desplazamiento en el Eje X

3.3.2.2.1 Subsistema Generar Pasos (Enable Subsystem)

Este subsistema posee 3 entradas, que son: *VEL (Velocidad), DIR (Dirección)* y *HOMEX (Inicialización del proceso HOME en el eje X)*. Las variables *VEL* y *HOMEX* para este bloque se expresan como variables de lectura, es decir, que van a receptar un dato enviado desde otra parte del programa. Dichos valores son enviados al interior del bloque para realizar otras funciones. Mientras que *DIR*, es una variable que proporciona la dirección del movimiento del motor y depende del valor de la señal obtenida desde el bloque *GIRO ACTIVACIÓN.* Como salidas, tenemos a *P1*, *P2*, *P3* y *P4* cuyo valor depende de la secuencia de pasos que se ha de ir proporcionando al motor para producir su movimiento.

Adicionalmente, la salida *PASOS* se encarga de enviar información para el avance, retroceso o suspensión del movimiento del motor. Este subsistema se diferencia de otros, debido a que éste posee un símbolo en la parte superior del bloque, conocido como "*habilitador*". Por tal razón, a este subsistema se le denomina también "*Enable Subsystem"*. Esto significa que, se realiza todo el proceso que contiene internamente, siempre y cuando exista el valor 1L en la entrada del habilitador. La señal del habilitador para este subsistema, depende del valor en la salida *NA* que se obtiene del bloque GIRO ACTIVACIÓN, como se observa en la figura 3.8, mientras que el subsistema interno *GENERAR PASOS* se lo puede observar en la figura 3.9.

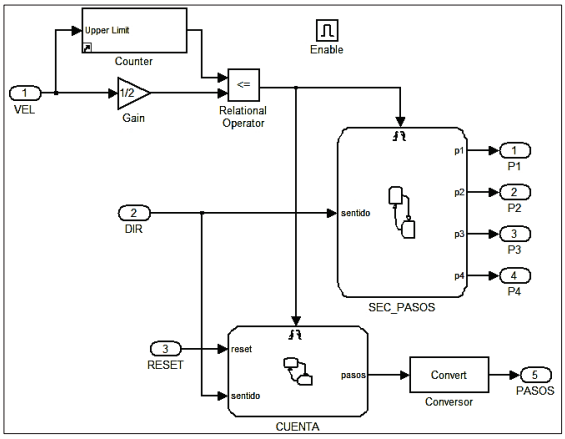


Figura 3.9 – Subsistema GENERAR PASOS

El motor PAP disponible para el eje X, es del tipo bipolar el cual es controlado en secuencia *Full Step.* Esta misma configuración se emplea en el resto de motores PAP para los ejes restantes. La figura 3.10 muestra el esquema de la disposición de las bobinas para un motor PAP bipolar, junto a la secuencia de pasos en la tabla 3.3 que se debe cumplir para la magnetización de las bobinas.

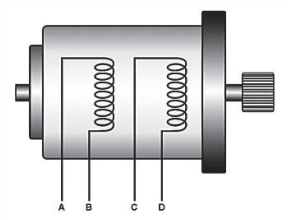


Figura 3.10 – Motor PAP

Tabla 3.3 – Secuencia de pasos

Bobina	***P1***	***P2***	***P3***	***P4***
A	1	0	1	0
C	1	0	0	1
B	0	1	0	1
D	0	1	1	0

La secuencia de la tabla anterior, se emplea en la máquina de estados *SEC_PASOS* tal como lo muestra la figura 3.11.

Para *SEC_PASOS*, tenemos 1 entrada y 4 salidas. La entrada denominada *sentido*, corresponde a la dirección (*DIR*) del sentido de giro del motor, que puede ser horario o anti horario. Como salidas tenemos a *p1*, *p2*, *p3* y *p4* que corresponden de manera homóloga a las salidas *P1*, *P2*, *P3* y *P4* del subsistema *GENERAR PASOS*. Los valores que se envían a través de ellos, son la secuencia de pasos que se van a enviar después de pasar por el subsistema *ANULAR PASOS* hacia el motor.

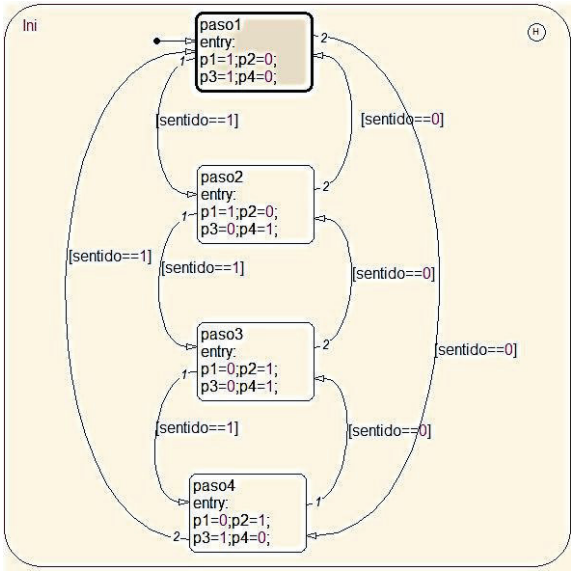


Figura 3.11 – Máquina de estados SEC_PASOS

La máquina de estados *SEC_PASOS, debe cumplir las siguientes condiciones* para que el motor actúe en sentido horario o anti horario:

- Si sentido == 0, el motor en el eje X avanza (aumenta el número de pasos)
- Si sentido == 1, el motor en el eje X retrocede (disminuye el número de pasos hasta llegar a cero).

El sentido de giro de los motores es relativo al punto de vista que lo tome el usuario. Para el prototipo, el punto de referencia corresponde al punto HOME, desde donde se registra el avance o retroceso de los componentes del puente. La figura 3.12 muestra la secuencia de pasos a cumplirse en la máquina de estados bajo estas condiciones.

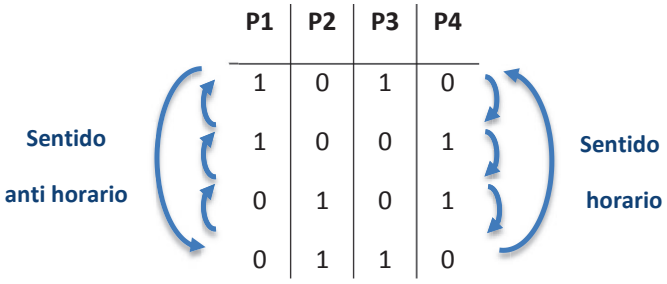


Figura 3.12 - Secuencia de pasos (sentido horario/anti horario)

En la máquina de estados, se cumple que si *sentido == 0*, los valores que van a enviarse al motor son 0110, y prosiguiendo si se mantiene la condición de *sentido== 0*, se envía 0101 y el proceso continua (siempre y cuando *sentido ==0*) hasta alcanzar el valor de 1010. Al ser un ciclo cerrado, mantiene la secuencia hasta llegar a un punto en el que *sentido==1*, que significa que el motor gira en sentido contrario. Para éste caso, se cumple todo el proceso en sentido contrario empezando desde el valor inferior de la secuencia, es decir 0110.

El avance, como el retroceso del motor PAP, se realiza de acuerdo a la frecuencia del período de la señal de reloj que se recibe en la parte superior de la máquina de estados desde el sub-bloque *Upper Limit* y la ganancia *Gain*, que pasan por un *Relational Operator (Operador Relacional)*. La combinación de estos elementos, producen una señal cuadrada que tiene pulsos altos de 10ms y pulsos bajos de 5ms. Esto, con el fin de producir una señal de reloj a una frecuencia tal que permita el movimiento del motor en el eje X. Esta señal se envía a la máquina de estados para *SEC_PASOS* y también, a la máquina de estados para *CUENTA*.

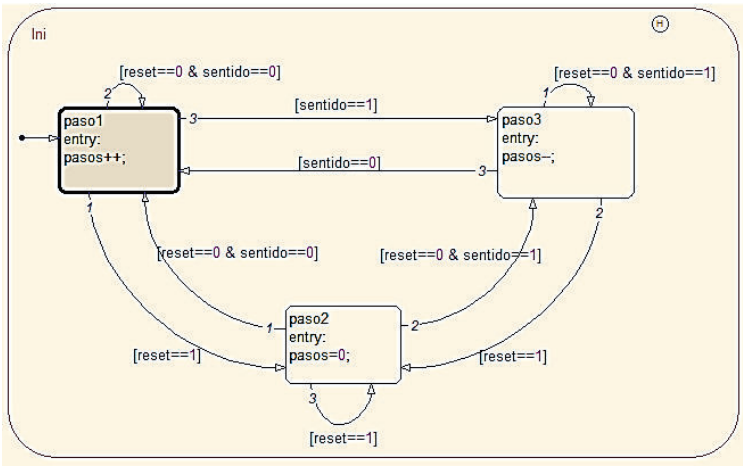


Figura 3.13 – Máquina de estados CUENTA (Eje X)

En la figura 3.9 se tiene la máquina de estados *CUENTA* la misma que posee dos entradas, que son: *reset* y *sentido*. La entrada *reset*, corresponde a la señal de *HOMEX*, la misma que se activa cuando se le envíe la señal para comenzar a realizar el proceso *HOME*. Mientras que la entrada *sentido*, es la misma entrada que posee la máquina de estados *SEC_PASOS*.

La máquina de estados para el proceso *CUENTA*, se observa en la figura 3.13.

El proceso de funcionamiento empieza desde el paso1, de modo que:

- Sí $reset == 0 \,\&\, sentido == 0, entonces\ pasos++$. Lo que significa, debido a que no se manda a realizar el proceso *HOME* en el funcionamiento, no se activa *reset*, y por lo tanto, el motor puede moverse. Y si *sentido==0*, se considera que el motor va a avanzar, según las condiciones anteriormente indicadas para el proceso *SEC_PASOS*. Si se cumplen tales condiciones, el motor empieza a moverse por el eje X.
- Ahora, si se sigue cumpliendo la condición anterior, pero si *reset==1*, significa que se ha mandado al puente a realizarse el proceso HOME. Y por lo tanto, se necesita que el motor se detenga. Es decir, se cumple que *pasos=0,* y mantiene esa posición mientras *reset==1*.
- Si se cumple $reset == 0 \,\&\, sentido == 1$, se entiende que el motor va a cambiar su sentido de giro, es decir, se retroceden pasos (obteniéndose, $pasos--$). Se mantiene en tal estado hasta que *reset* se active (*reset==1*)

Del cuadro perteneciente al paso 3, retorna al cuadro del paso 1 y viceversa, cuando se mantenga reset==0 y sentido pueda ser 1 o 0. Entonces, el proceso de reducir o retroceder pasos se mantiene, mientras se cumpla esta condición.

Los datos como: *pasos--, pasos++* y *pasos=0*, se envían al sub-bloque *Conversor* (*Convert*), con el fin de convertir el tipo de dato a la entrada en un tipo de dato ***double*** que pueda ser reconocido y almacenado en un espacio de memoria definido en la programación, pues que el software SIMULINK trabaja únicamente con datos del tipo ***double*** *(32 bits)* para finalmente enviarlos a la salida *PASOSX*.

3.3.2.2.2 Bloque Giro Activación

La figura 3.8 muestra el *bloque GIRO ACTIVACIÓN*, cuya función es deshabilitar el movimiento a lo largo del eje X, y para lo cual, emplea la señal obtenida por los fines de carrera dispuestos a los extremos de todo su recorrido. Este bloque tiene

como entradas a *SW1*, *SW2*, *POSX* y *PASOSX*; y como salidas, a *NA* y *DIR*. Las entradas *SW1* y *SW2*, responden a las señales del contacto que se produce cuando el puente grúa ha alcanzado a llegar al final de todo su recorrido en ambos sentidos. *POSX*, es la posición que el usuario ingresa por la Tablet y se entiende como la posición expresada en milímetros que se escribe en la coordenada correspondiente al eje X, a la misma que el usuario quiere que el puente grúa se traslade. Mientras que *PASOSX*, va a recibir y enviar la información que se obtiene anteriormente de la salida *PASOS* del Subsistema *GENERAR PASOS*, y que se relacionan con el movimiento del motor descrito en el bloque *GIRO ACTIVACIÓN*. La salida *NA* en este bloque, es el habilitador de la secuencia para permitir o no, la realización del proceso que se cumple internamente en el subsistema *GENERAR PASOS*; y la salida *DIR*, sirve para determinar la dirección del motor PAP. En el ANEXO A-2, se indica internamente el esquema completo del bloque *GIRO ACTIVACIÓN*. Como se observa en el esquema, se tienen tres máquinas de estado, las cuales son: *DESHABILITAR MOV*, *DESHABILITAR MOV1* y *HABILITAR HOME*. Las dos primeras se encargan de deshabilitar el movimiento del motor PAP en el instante en el que éste hace contacto con los fines de carrera *SW1* y *SW2*, respectivamente.

Mientras que el proceso *HABILITAR HOME*, se encarga de permitir o a su vez, inicializar el proceso *HOME* correspondiente al eje X. En este bloque además, se ha determinado una relación que se debe cumplir para limitar los pasos en el avance o retroceso del motor. Dicha relación, va a estar determinada por una diferencia entre dos valores de entrada, que en este caso son *PASOS* y *POS*; y un operador lógico que relaciona a los valores de las variables *MMANU* e *INICIAR*, en la figura 3.14 se observa el esquema de tal relación.

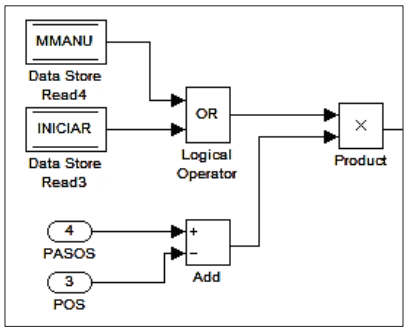


Figura 3.14 – Esquema de relación entre POS y PASOS (Eje X)

La entrada *PASOS* corresponde a *PASOSX*, mientras que la entrada *POS* corresponde a *POSX*, visto en la figura 3.8 del bloque *GIRO ACTIVACIÓN*.

El operador *Add*, cumple la función de comparar las entradas *PASOS* y *POS,* con el fin de determinar cuál de estos dos valores es el mayor con respecto a la posición que ya se ha definido por el ingreso del dato desde la Tablet. Obteniéndose así, dos posibles situaciones:

- Si $pasosx - posx \geq 0$. Esto quiere decir, que si los pasos en el eje X son $mayores$ a la posición ingresada en la coordenada X, los pasos van a disminuir. Es decir, van a ir decreciendo hasta alcanzar a la posición determinada. Si se cumple esta condición, se manda el valor lógico 1 a una de las entradas del operador lógico *Product*.
- Si $pasosx - posx \leq 0$. De igual forma, esto nos indica que si los pasos en el eje X son $menores$ a la posición ingresada en la coordenada X, los pasos van a aumentar con el fin de alcanzar a la posición determinada. Si se cumple esta condición, se envía el valor lógico 0 a una de las entradas del operador lógico *Product*.

Cuando *PASOSX* alcanza a *POSX* el motor se detiene. Mientras que el valor lógico obtenido de esta relación, con el resultado del operador lógico *OR* entre *MMANU* e *INICIAR*, proporciona la señal lógica necesaria que busca comparar el resultado final de todo el esquema para determinar la condición a cumplirse en la habilitación de la salida *DIR*.

En la figura 3.15, se observa el esquema relacional para la salida *DIR* en el bloque *GIRO ACTIVACIÓN.*

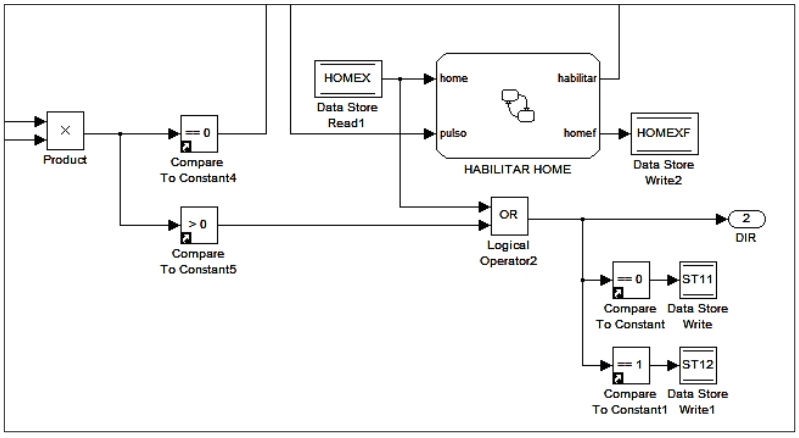


Figura 3.15 – Esquema relacional para DIR (Giro Activación)

Las señales lógicas de *MMANU* e *INICIAR* van a depender de la activación o desactivación de los botones, desde la interfaz de la Tablet. El operador lógico *Product* multiplica los valores lógicos obtenidos y en su salida, dicho valor pasa a compararse con estos dos casos (==0) o (>0).

Como se observa en la lógica de habilitación para *DIR*, se toma en cuenta el segundo caso y si se cumple dicha condición, envía 1L a la entrada del operador lógico *OR*, con lo cual se puede habilitar a *DIR*. Y a la vez, la señal receptada desde la variable *HOMEX*, que se envía a una de las entradas de la máquina de estados de *HABILITAR HOME*, podría hacerlo.

Es importante denotar que *HOMEX* tiene mayor prioridad respecto a la lógica de habilitación que comienza desde la relación de *PASOSX* y *POSX* hasta terminar en la comparación lógica (>0). Esto, debido a que el valor de *HOMEX* depende fundamentalmente de la inicialización para la activación del proceso *HOME*, función que se envía desde el botón *HOME* ubicado en la interfaz de la Tablet.

El valor enviado a *DIR* podría ser 1, o bien 0. Por esta razón, dicho valor se lo ha tomado en consideración para lograr deshabilitar el movimiento del motor. En la figura 3.16, se observa el esquema comparador.

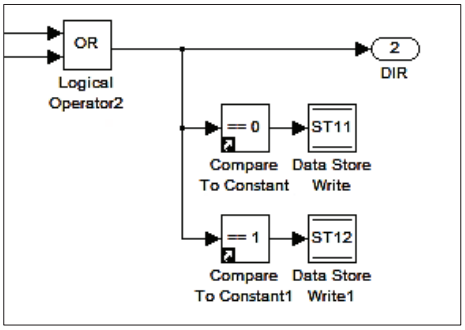


Figura 3.16 – Comparador para DESHABILITAR MOV y MOV1

En el esquema, se tienen dos variables (*ST11* y *ST12*) y cuyos valores van a depender de la resultante en la comparación de (==0) y (==1), respectivamente.

La señal que toma *ST11* es empleada en la entrada de la máquina de estados para *DESHABILITAR MOV*, la cual pasa a ser *sentido*. Mientras que, *SW1* corresponde al valor que se tiene del contacto del motor al pulsar el fin de carrera (es decir 0). De la misma manera, *ST12* corresponde a la entrada para *DESHABILITAR MOV1* e internamente pasa a ser también *sentido* para esta

máquina de estados, a la vez, se emplea la señal del fin de carrera *SW2*. Como se observa en la figura 3.17, tanto *SW1* como *SW2* pasan por un operador lógico *NOT* con el fin de obtener 1L en la entrada *SW* en ambos casos, pues como se había visto a principios de esta sección se había empleado una configuración *Pull Up* para las entradas digitales de los fines de carrera.

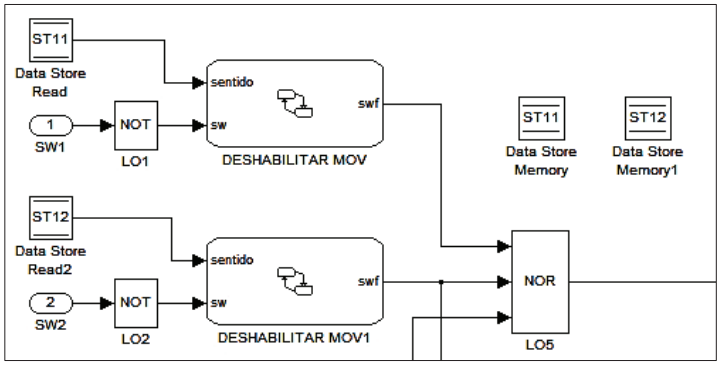


Figura 3.17 – Esquema DESHABILITAR MOV y MOV1

En la parte interna de este bloque, se han declarado las variables globales *ST11* y *ST12*, que son utilizadas para escribir y enviar datos internamente hacia otros procesos. Para este caso, se tiene la máquina de estados *DESHABILITAR MOV* y cuyo interior se muestra en la figura 3.18.

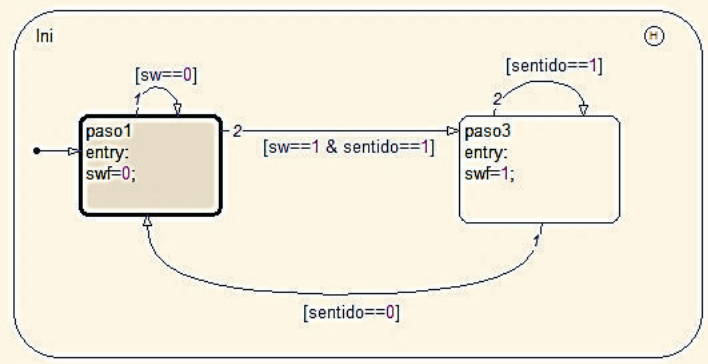


Figura 3.18 – Máquina de estados DESHABILITAR MOV

A la salida de la máquina de estados tenemos la variable *swf* cuya función es de interrumpir el giro del motor PAP. En la entrada de la máquina de estados, SW1 pasa a ser *SW* y ST11 pasa a ser *sentido*, cumpliéndose así las siguientes condiciones:

- Mientras $sw == 0 \rightarrow swf = 0$, indica que mientras no actúe el fin de carrera su valor lógico es 0, lo que implica que el motor pueda seguir girando.
- Permanece en ese bucle hasta que $sw == 1 \,\&\, sentido == 1 \rightarrow swf = 1$, esto indica que el motor moviéndose en tal dirección, ya ha alcanzado a hacer contacto con el fin de carrera, y lo mantiene presionado. Cumpliéndose estas dos condiciones, *swf* pasa a ser 1L. Esta señal, envía a desactivar inmediatamente el movimiento del motor a pesar de que el valor lógico de sentido siga siendo 1. Con lo cual, se podría considerar que el motor continua girando hacia esa dirección, pero para evitar que esto ocurra, y además, para evitar el efecto rebote propio de los fines de carrera, se previene mandar a apagar al motor directamente.
- Al mantener esa posición, en algún instante del funcionamiento el usuario ordena mediante el control de la Tablet, que el motor cambie su sentido de giro y es entonces cuando $sentido == 0$, e inmediatamente esta condición desactiva a *swf*. De esta manera, el motor continúa moviéndose pero en sentido contrario.

El ciclo de funcionamiento descrito anteriormente que pertenece a la máquina de estados, vuelve a realizarse cuando se active nuevamente la señal SW proveniente del fin de carrera. Para el caso de la máquina de estado *DESHABILITAR MOV1*, que se observa en la figura 3.19, se cumple el mismo procedimiento que la máquina de estado explicada anteriormente y únicamente se diferencia con la primera, ya que posee diferentes entradas. Siendo así que, *ST12* pasa a ser *sentido* y también, *SW2* es *SW* como se pudo observar anteriormente en la figura 3.17.

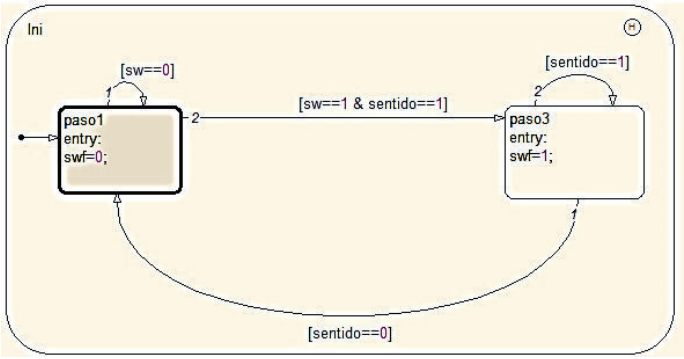


Figura 3.19 – Máquina de estados para DESHABILITAR MOV1

Los valores lógicos obtenidos en las salidas *swf* de ambas máquinas de estado, se suman a la señal obtenida en el comparador (==0) para que al pasar por un operador lógico *NOR* se obtenga el valor resultante. Este a su vez, al relacionarse con la salida de la máquina de estado *HABILITAR HOME* proporcione la activación (es decir, 1L necesario) para activar a *NA* y finalmente, habilitar la secuencia. Este dato como se explicó anteriormente, es tomado por la máquina de estado *SEC_PASOS* para permitir en éste su ciclo de funcionamiento en la secuencia de los pasos del motor. La figura 3.20 muestra el esquema de habilitación de la salida *NA*, para este bloque.

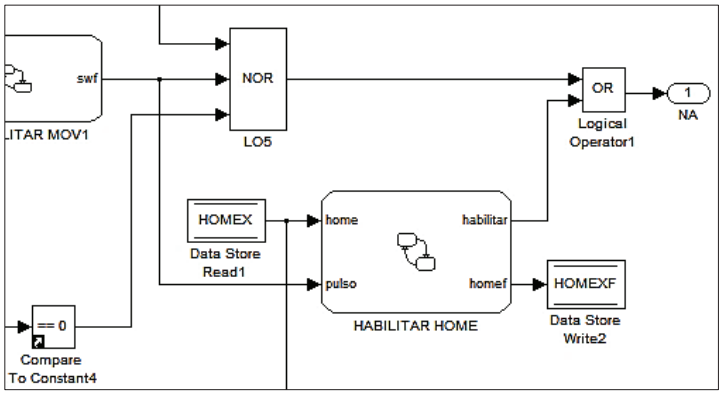


Figura 3.20 – Esquema de habilitación para NA

El esquema de habilitación también depende del proceso a realizarse en la máquina de estado *HABILITAR HOME*. Se ha desarrollado este proceso con la finalidad de obtener una señal disponible de activación que permita al proceso *HOME* desarrollarse. Esta acción básicamente ocurre cuando el puente, después de ser utilizado, se mantiene estacionario en cualquier posición, lo cual implicaría que la señal para *NA* es 0 y si al oprimir el botón *HOME* desde el control, no se obtendría el funcionamiento para *HOME* ya que el motor permanecería detenido. Por ello, necesariamente *NA* debería ser 1 para que el motor manteniéndose en ese estado al recibir la señal empiece a moverse.

En dicha máquina de estado, la señal de la variable *HOMEX*, es una bandera cuya función es dar una señal de activación a la entrada *home* para iniciar el ciclo de funcionamiento interno de la máquina de estado. Mientras que *pulso*, tiene la misma señal que se obtiene de la salida *swf* obtenida de la máquina de estado *DESHABILITAR MOV1*. Esta señal permite determinar en el funcionamiento,

cuando el motor actúa en el fin de carrera para apagarlo (donde*, swf==1*). Por otro lado, cuando el motor se encuentra en movimiento (se tiene, *swf==0*). Dichas señales permiten controlar la secuencia del proceso *HOME* para el prototipo.

La salida *habilitar* brinda un valor lógico que permite el movimiento del motor, mientras que *homef* proporciona la señal para indicar que la secuencia en el eje X correspondiente al proceso *HOME* se ha realizado y de tal forma, el proceso respectivo para los otros ejes continúe realizándose hasta finalizarlo por completo. En la figura 3.21, se puede observar el ciclo de funcionamiento del proceso *HABILITAR HOME*, que se realiza internamente en la máquina de estado.

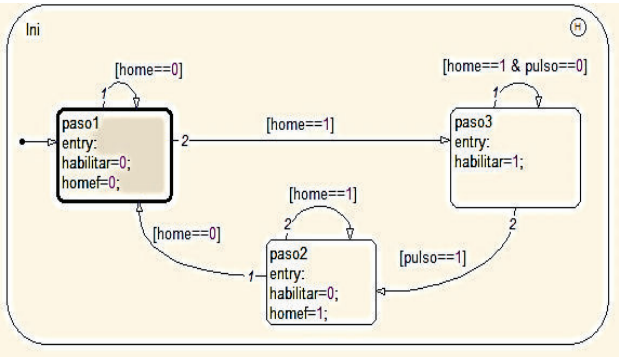


Figura 3.21 – Máquina de estado HABILITAR HOME

El proceso *HABILITAR HOME* cumple el siguiente ciclo de funcionamiento:

- Sí $home == 0 \rightarrow (habilitar = 0 \,\&\, homef = 0)$, esto significa que el usuario no ha mandado una señal al sistema para realizar el proceso HOME desde el control remoto. Lo cual, implica que no se activa la salida *habilitar* y mucho menos, la salida *homef* porque aún no se ha realizado el proceso de la máquina de estado.
- Al encontrarse activado *home*, se cumple que $(home == 1 \,\&\, pulso == 0) \rightarrow habilitar = 1$, al habilitarse *home* y permanecer en ese estado, junto a la señal de *pulso* que nos indica que el motor aún no ha hecho contacto con el fin de carrera, necesariamente la salida *habilitar* debe activarse. Con lo cual, el motor empieza a dirigirse hacia el espacio de seguridad HOME pero cumpliendo sólo el recorrido determinado para el eje X.
- Para salir de dicho estado, se necesita que la señal de *pulso* se active, lo cual indica que el motor ya finalizó su recorrido al hacer contacto con el fin

de carrera. A la vez, se cumple que $home == 1 \rightarrow (habilitar = 0 \,\&\, homef = 1)$, esto quiere decir que *home* permanece activado durante todo el proceso hasta que el proceso *HOME* finalice por completo. Debido a que falta realizarse el proceso HOME de los ejes (Y, Z), se requiere desactivar la señal para habilitar el movimiento del motor en este eje y también enviar a activarse a la salida *homef*, ya que el proceso *HOME* en este instante para este eje ya ha terminado.

- Pasa al cuadro de paso1, cuando *home* se desactive. Esto ocurre, cuando en el proceso de funcionamiento no se cumple *HOMEX*. De tal forma, con tan sólo habilitar *home,* se inicia nuevamente este ciclo cerrado.

Este proceso junto al valor obtenido como resultado al final en la salida *habilitar*, también es otra forma para activar a *NA* del bloque *GIRO ACTIVACIÓN*.

3.4.2.3.3 Sub-bloque Anular Pasos

Continuando con el esquema de desplazamiento del motor en el eje X, se tiene el sub-bloque interno *ANULAR PASOS,* cuya función en la lógica de programación para el desplazamiento del motor PAP, consiste en anular los datos anteriormente enviados al ciclo de funcionamiento del motor. Esto, con el fin de evitar que el motor PAP quede enclavado (energizado) al llegar a una posición específica a la que se le ha ordenado dirigirse. Con el desarrollo de este sub-bloque en la programación, se busca reducir los daños que podrían originarse en los bobinados del motor PAP a causa de mantener al motor energizado durante períodos de tiempo muy largos. La figura 3.22 muestra el interior de este bloque.

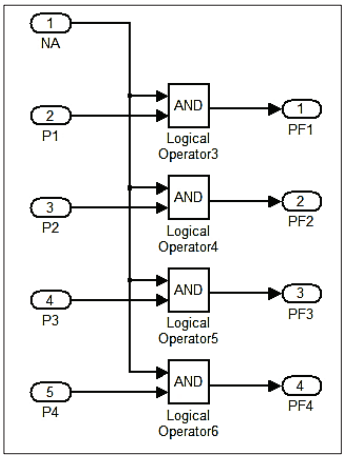


Figura 3.22 – Sub-bloque interno ANULAR PASOS

Como se observa, tenemos como entradas a *NA, P1, P2, P3* y *P4*. Como ya hemos visto *NA* es el habilitador de la secuencia, mientras que *P1*, *P2*, *P3* y *P4* son las salidas provenientes del subsistema *GENERAR PASOS* y corresponden a las entradas del motor. En la salida tenemos *PF1*, *PF2*, *PF3* y *PF4*, que representan a las salidas principales del esquema de desplazamiento para el eje X, y que van a conectarse directamente a las 4 terminales del driver L298 del motor que controla dicho eje.

El sub-bloque *ANULAR PASOS*, cumple las siguientes condiciones:

- Cuando $NA == 1$, habilita la generación de pasos y permite que la secuencia pase a los terminales del motor (*PF1, PF2, PF3, PF4*).
- Cuando $NA == 0$, deshabilita la generación de pasos y pone el valor lógico de cero (0L) a todo el sistema.

Los operadores lógicos AND comparan cada uno de los valores de las entradas del motor (*P1*, *P2*, *P3* y *P4*) con *NA*. Para que finalmente, según el valor del habilitador de *NA* sea posible, o no, enviar la secuencia al motor y generar su movimiento.

3.4.2.4 Desplazamiento del motor en el eje Y

El proceso de control determinado para el funcionamiento y la programación realizada en SIMULINK que cumple dicho proceso, corresponde a la misma estructura funcional y lógica que ha sido anteriormente explicada para el caso del desplazamiento en el eje X. La figura 3.23 representa el esquema empleado para el eje Y.

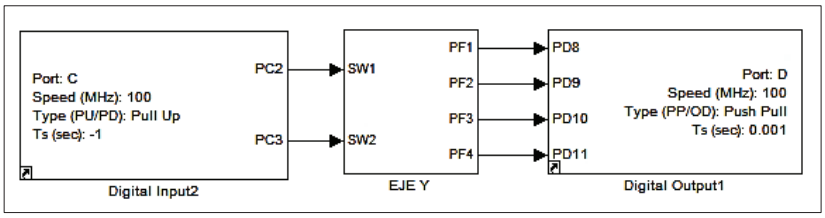


Figura 3.23 – Esquema desplazamiento del motor en el eje Y

De tal forma, se tiene el mismo esquema de desplazamiento anteriormente detallado para el caso del eje X, lo cual implica a la vez, la misma estructura de programación, ver figura 3.24

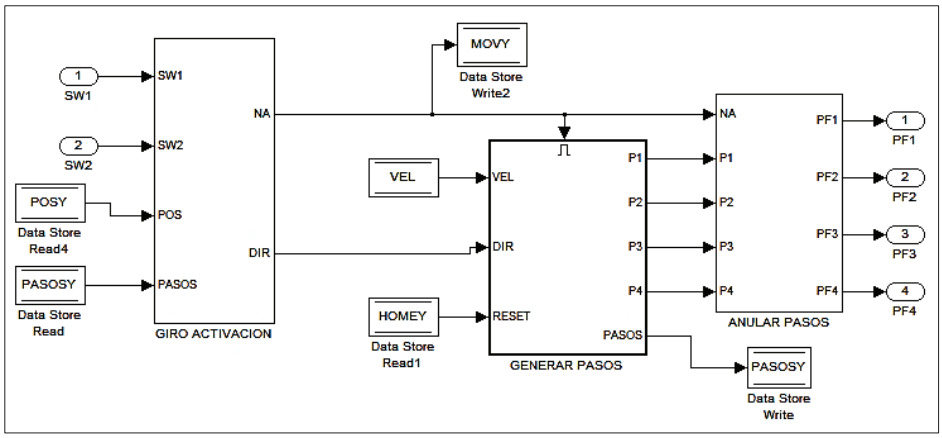


Figura 3.24 – Esquema interno del bloque de desplazamiento

El desarrollo en procesos cíclicos, tanto en máquinas de estado, como en funciones y operadores lógicos para el desplazamiento del eje X, son los mismos que para el eje Y, pues cumplen a cabalidad todas las características y funciones que se deben cumplir para el desplazamiento del motor PAP en este eje, puesto que, ambos dependen directamente de la señal enviada al hacer contacto con el fin de carrera (que le corresponde a cada uno de ellos) con el fin de detener el movimiento, o su a vez, para que sea posible la generación de pasos y permitir el desplazamiento del motor a lo largo de su recorrido. Adicionalmente, en el ANEXO A-3 se indica el esquema completo empleado para la programación en el entorno SIMULINK para el desplazamiento del motor en el eje Y.

3.4.2.5 Desplazamiento del motor en el eje Z

En el caso del desplazamiento del motor en el eje Z, existen ciertas modificaciones y adiciones en su respectivo control por motivos de construcción. Sin embargo, se mantiene el mismo definido al inicio de la programación, para el desarrollo del desplazamiento de éste eje. Esto se lo ve en la figura 3.25.

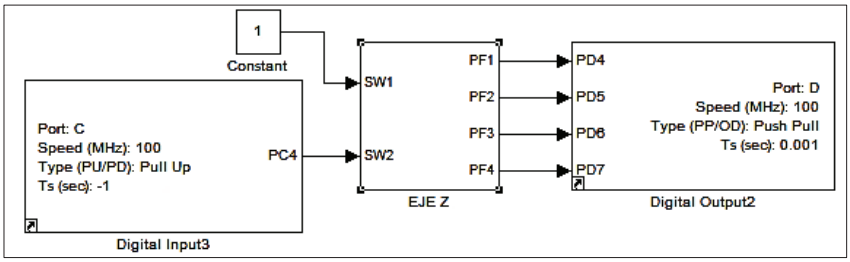


Figura 3.25 – Esquema interno del bloque desplazamiento eje Z

Como se observa, posee una cierta diferencia en comparación con los dos esquemas anteriormente analizados. La constante con valor 1 que se conecta a la entrada de *SW1*, simula la presencia de un fin de carrera presente en tal entrada, con un estado normalmente abierto. Esto, debido a que en el recorrido del motor PAP en el eje Z, sólo se dispone de un fin de carrera que limita el movimiento en el ascenso, cuya señal corresponde a *SW2*. Debido a que no existe un fin de carrera que sirva para limitar el recorrido de la pinza en su descenso, se ha determinado un valor de pasos que el motor PAP va a desarrollar para llegar al fin del recorrido admisible en el eje Z, tal valor es de 1500 pasos. Dicho valor es de gran importancia en el caso de que se quiera modificar la distancia de dicho recorrido o reemplazar el motor paso a paso junto a otro sistema de izaje.

En la figura 3.26, se observa el esquema interno del bloque de desplazamiento para el eje Z. El cual, internamente está compuesto por los mismos subsistemas, bloques y sub-bloques anteriormente descritos para los ejes X, Y. A la vez, cada uno de ellos, produce y mantiene la lógica de funcionamiento para el prototipo. Sin embargo, existen ciertos cambios realizados al bloque *GIRO ACTIVACIÓN*, los cuales se detallan a continuación.

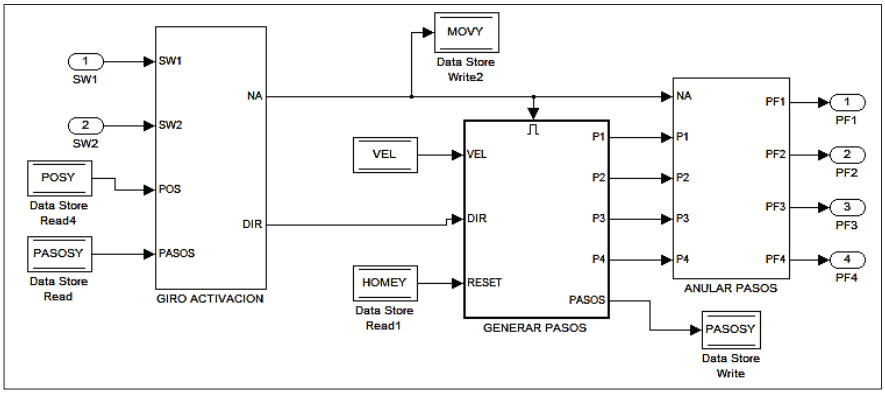


Figura 3.26 – Esquema interno del bloque desplazamiento eje Z

3.4.2.5.1 Bloque Giro Activación para el eje Z

En el ANEXO A-4 se puede observar las partes modificadas y/o añadidas al proceso. Las máquinas de estado *DESHABILITAR MOV*, *DESHABILITAR MOV1 y HABILITAR HOME* que corresponden a este bloque, mantienen su ciclo de funcionamiento al igual que sus entradas y/o salidas, según lo anteriormente explicado en el desplazamiento del motor en el eje X.

Sin embargo, una de las modificaciones para el programa es la adición de las variables en una relación de comparación que junto a la variable *MMANU* van a determinar la dirección del motor (DIR), las cuales son: *PASOSX, POSX*, *PASOSY, POSY*, que van a compararse con el operador de relación (==). Como se observa en la figura 3.27, esto quiere decir, que la variable PASOSX se va a relacionar con *POSX* y cuando se cumple que *PASOSX* == *POSX* se envía 1L a la salida; tal relación, también se va a desarrollar para el eje Y.

Adicionalmente, se tiene la variable *INICIAR* que esta vez se va a comparar con una constante (==1) y de la misma manera si se cumple que *INICIAR ==1*, se envía 1L a la salida. Para después, llegar hasta un operador lógico *AND*, que se conecta a la entrada del operador lógico *OR* y que a la vez, también se conecta a un segundo operador lógico *AND*. El cual, depende de una variable denominada *PIEZAREC*, cuya función es enviar una señal que indica que la pinza ha tomado o no un objeto.

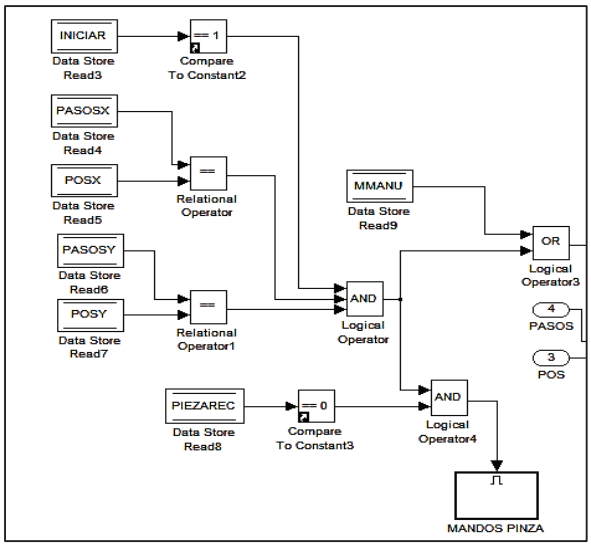


Figura 3.27 – Relación de comparación del bloque GIRO ACTIVACIÓN

Para este caso, *PIEZA REC==0* y esto significa que la pinza aún no ha recogido un objeto, por lo cual se envía 1L al operador lógico *AND*. Dicha señal, más la señal obtenida de la comparación anterior entre las variables añadidas, envía un resultado a la salida y si es 1L, se activa un *Enable Subsystem*, llamado *MANDOS PINZA* y en cuyo interior se observa el esquema de la figura 3.28. Con dicha señal de activación, la pinza se abre.

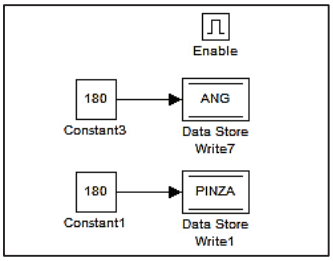


Figura 3.28 – Esquema interno de MANDOS PINZA (Enable Subsystem)

En dicho esquema, se tienen dos constantes de valor 180 que llegan a las variables *ANG* y *PINZA*. Cuando se envía una constante con valor 180 a la variable *ANG*, ésta envía la señal para que empiece el conteo de los 4 segundos para la apertura de la pinza. Esta condición, ya ha sido definida a principios de este capítulo en la lógica de control, ya sea en modo de posicionamiento manual o automático del prototipo y que ha sido determinada por motivos de seguridad al momento de abrir la pinza para soltar el objeto cuando se coloca en otra posición.

Sin embargo, dicha condición no es empleada en esta parte de la programación debido a que en el desplazamiento de modo automático, cuando el puente grúa se sitúe en una posición definida en las coordenadas (X, Y), primero debe abrir completamente la pinza para descender hasta la posición que ha sido determinada por el usuario en el eje Z.

Si se activa solamente la variable *ANG*, la pinza en esa posición se abriría después de 4 segundos y para evitar esto, el mismo valor de la constante (180) se envía a la variable *PINZA*, que se encarga de realizar la operación de apertura de la pinza. Con esto, si la constante *ANG* después de 4 segundos manda la señal a *PINZA* para realizar el proceso de apertura no afecta en nada, ya que dicha variable anteriormente ya había sido activada.

De lo anterior, en lo que se refiere a cumplir la condición de realizar primero el desplazamiento de los motores de los ejes (X, Y) para luego continuar con la apertura de la pinza y el desplazamiento del motor en el eje Z, significa que, estando en modo de desplazamiento automático, el desplazamiento de los motores correspondientes a los dos ejes debe realizarse de manera simultánea e inmediata cuando se ordene al puente grúa dirigirse a cierta posición mediante la Tablet.

Prosiguiendo con las modificaciones en la programación, en la figura 3.29, se observa la adición de una cierta relación que comprende a las variables: *POSZ* y *PASOSZ*; que forma parte del esquema de habilitación para la salida *NA* del bloque *GIRO ACTIVACIÓN*, para el eje Z.

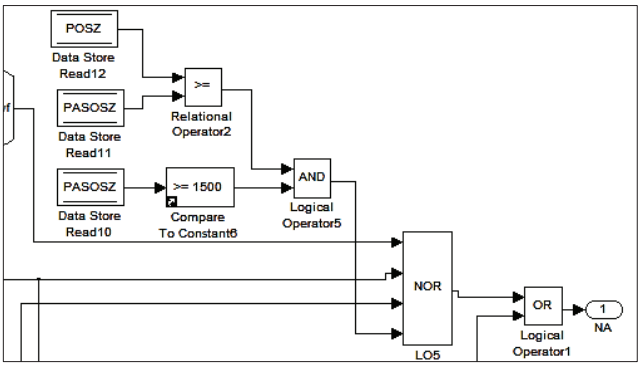


Figura 3.29 – Esquema de habilitación del bloque GIRO ACTIVACION

Esta relación modificada y añadida al bloque *GIRO ACTIVACIÓN*, comprende una comparación entre *POSZ* y *PASOSX* empleando el operador relacional (>=). Puesto que, la dirección de movimiento en el eje Z es hacia abajo, entonces $POSZ >= PASOSZ$. Al cumplirse esta condición, se envía un 1L a la entrada de la compuerta lógica AND, con lo cual la pinza comienza a descender, ya que no se ha habilitado NA para detener el movimiento del motor. De modo que, cuando se cumple que $PASOSZ >= 1500$, implica que el motor a pasos ha terminado de recorrer los 1500 pasos definidos para su desplazamiento en el eje Z, para lo cual debe detenerse, activando finalmente a *NA*.

Caso contrario, cuando se ordene a la pinza empezar a ascender nuevamente, no se cumple la condición $POSZ >= PASOSZ$, con lo cual se deshabilita *NA* y la pinza puede subir libremente hasta que el fin de carrera ubicado en la parte superior de ésta, termine haciendo contacto para detener el movimiento del motor.

3.4.3 BLOQUE CONTROLADOR DE LA PINZA [26]

Para el control de la pinza y en cuanto a la programación, se ha definido el bloque que se puede observar en la figura 3.30.

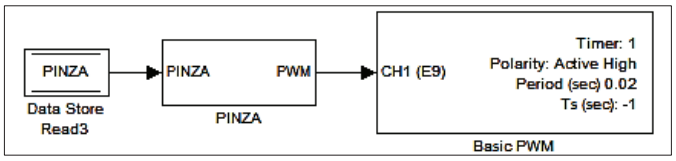


Figura 3.30 – Bloque controlador de la PINZA

Dicho bloque, tiene como entrada a la variable *PINZA* y como salida a PWM. La variable *PINZA* recibe la información de apertura o cierre de la pinza, siempre y cuando haya sido comandado anteriormente por la señal de la variable *ANG*, que representa el valor de apertura o cierre de la pinza. En la salida, tenemos un sub-bloque llamado *Basic PWM,* cuya función es generar un PWM (Modulación de Ancho de Pulso) que va a controlar al servomotor para el ascenso y descenso de la pinza. Los parámetros determinados para la programación en este sub-bloque se los presenta en la figura 3.31.

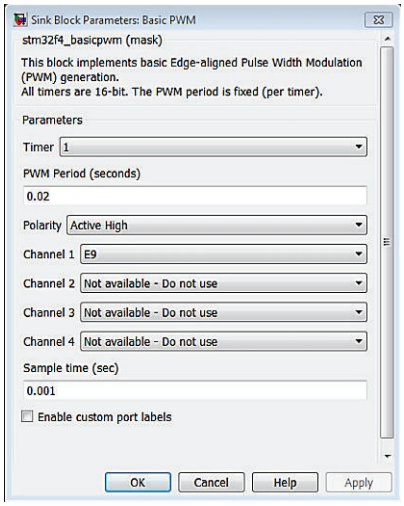


Figura 3.31 – Parámetros del sub-bloque Basic PWM

En los parámetros que definen a Basic PWM, tenemos la opción de modificar el período de la onda que controla al servomotor según la necesidad, por lo que se establece un valor de 20ms (a una frecuencia de 50Hz). Añadiendo a lo anterior, se emplea el *Timer1*, de donde se obtiene la señal de reloj. Según *Active High*, se define que la señal de la onda empieza con un pulso alto. El canal donde se recepta la señal que controla al servomotor perteneciente al microcontrolador, es *E9*. Finalmente, *Sample time*, que no es más que el período de tiempo que va a

definir la velocidad a la cual el microcontrolador va a realizar las funciones y acciones al enviar o recibir los datos en un proceso.

El período de la onda que se emplea para el control por PWM en el servomotor, puede variar entre un valor de 30ms como máximo y 10ms como mínimo, pero generalmente se recomienda emplear un valor intermedio de 20ms, justificando dicha elección por las siguientes razones:

- Con un período de 10ms, el servomotor tiene una velocidad muy rápida y su giro es muy inestable.
- Con un período de 20ms, el servomotor mantiene una velocidad intermedia entre rápida y lenta, y su giro se mantiene estable.
- Con un período de 30ms, el servomotor tiene una velocidad muy lenta, pero su giro se estabiliza de manera rápida.

Considerando tal recomendación, se ha empleado un período de la onda de 20ms, que ha sido ya definido en los parámetros del bloque *Basic PWM*. El ancho de pulso, determina el ángulo del movimiento del servomotor y a razón de que, generalmente un servomotor puede moverse de 0° a 180°, se hace necesario determinar el ancho de pulso respectivo para obtener tales desplazamientos.

Sin embargo, estos valores pueden diferir del tipo de servomotor a utilizarse, y por las características establecidas por el fabricante en el datasheet del servomotor. En la figura 3.32, se tiene el esquema interno del control para la pinza.

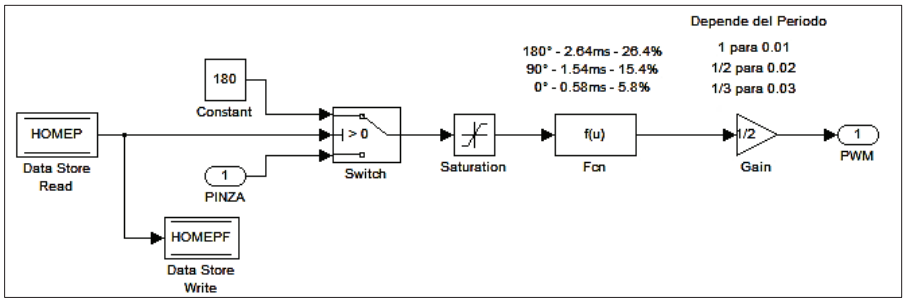


Figura 3.32 – Esquema interno del Bloque PINZA

Con un PWM, se puede variar el ancho de pulso desde un 0 a un 100% de su totalidad, y como el ingreso al programa para realizar la apertura o cierre de la pinza se la realiza mediante valores angulares, es necesario transformarlos a valores porcentuales. Para ello, se define una función que permita obtener el valor angular ingresado a porcentaje, utilizando el sub-bloque *FCN*.

3.4.3.1 Sub-bloque FCN (Linealización)

Es un bloque de expresión general, en el cual es posible introducir funciones y expresiones para determinar un cálculo u obtención de diferentes datos a su salida. Así, el sub-bloque *FCN* nos permite realizar la transformación de los valores angulares a porcentajes necesarios para enviarlos a la entrada de PWM correspondiente al servomotor. Como se observa en la figura 3.33, en los parámetros internos de tal sub-bloque se ha empleado una expresión cuadrática que nos sirve para obtener dicha relación, la cual está dada por:

$$Y = 0{,}00008642u^2 + 0{,}0989u + 5{,}8 \qquad (3.4)$$

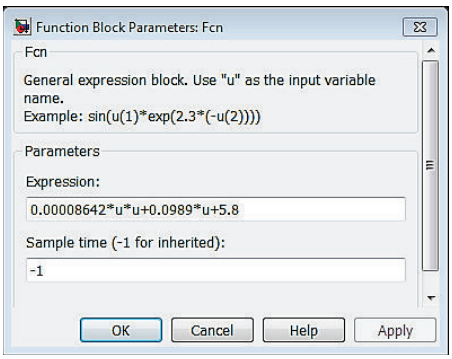


Figura 3.33 – Esquema interno Sub-bloque FCN

La expresión 3.4 mostrada anteriormente es normalizada, y para su cálculo, se asume un PWM con período de 10ms, donde se ingresan valores angulares de 0° a 180° y a la salida se obtiene valores porcentuales de 5.8% a 26,4%. De esta manera se emplean los puntos: P1 (26,4; 180°), P2 (15,4; 90°) y P3 (5,8; 0°), todos ellos expresados en porcentajes.

3.4.3.2 Sub-bloque SATURACIÓN

Se ha determinado mediante pruebas físicas, que el servomotor que controla la apertura y el cierre de la pinza, no puede girar de 0° a 180° y sólo puede moverse aproximadamente de 110° a 170° de su recorrido. No puede cerrarse hasta llegar a 0°, debido a que los dientes ubicados en la superficie de la pinza en 110°, ya llegaron a toparse; y no puede abrirse más, debido a que la pinza al llegar a 170°, ya ha llegado a su límite físico posible y completo definido para la pinza según las condiciones del prototipo.

El sub-bloque Saturación permite definir estos valores, como un límite alto y bajo para la entrada y que en la programación, tales valores se determinan en los parámetros *Upper Limit* (Límite Alto) y *Lower Limit* (Límite Bajo) ubicados en la parte interna del sub-bloque, como se observa en la figura 3.34.

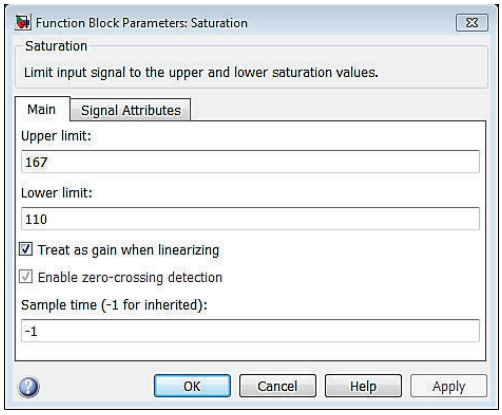


Figura 3.34 – Parámetros del Sub-bloque Saturación

El valor 167 ubicado en *Upper Limit*, significa que el servomotor va a girar hasta 167° y por lo tanto, la apertura de la pinza cubre hasta un poco menos de la superficie física total de su cuerpo. Este valor se justifica debido a razones de seguridad, aun así el límite de 170° descrito anteriormente sirve como referencia en la apertura total para la pinza. El valor 110 empleado en *Lower Limit*, significa que el servomotor, gira hasta 110° de su recorrido, logrando finalmente que la pinza se cierre por completo.

3.4.3.3 Sub-bloque GAIN [44]

El sub-bloque GAIN aplica un factor multiplicador constante a la información de entrada y el producto lo transmite como la información de salida, de tal forma el factor multiplicador es la ganancia. En la programación, el sub-bloque GAIN se encarga de enviar el porcentaje correcto inmediatamente después de compararlo con el período empleado para el funcionamiento de la pinza. En la figura 3.35, se indica la parte interna del sub-bloque. Como se observa, en la sección *Gain* tenemos ½, y en *Multiplication*, tenemos a la multiplicación de esta constante (1/2) con el valor obtenido del sub-bloque FCN. Tal relación, depende del período que se va a emplear para realizar el movimiento del servomotor acoplado a la pinza.

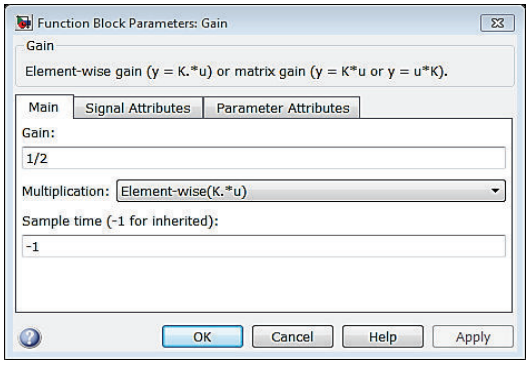


Figura 3.35 – Parámetros Sub-bloque GAIN

Para este caso, el valor del porcentaje se debe dividir para 2, o a su vez, multiplicarse por ½. Esto, debido a que en el ancho de pulso definido para el servomotor que se emplea para un ángulo de 0°, se debe mantener el valor de 0,58ms con un período de 20 ms que es el más recomendable y a razón que se utilizó un período de 10ms para el cálculo de la expresión en el bloque *FCN*, es necesario dividirlo para mantener esa condición.

Finalmente, el valor obtenido después de pasar por cada uno de los procesos de todos los sub-bloques, se envía a la salida PWM que se conecta directamente a la entrada PWM del servomotor para producir su giro.

3.4.3.4 Sub-bloque SWITCH

Como se observa en la figura 3.36 el sub-bloque *Switch*, actúa como un selector de acuerdo al criterio o condición establecido en la entrada media de tal sub-bloque, lo cual implicaría que la 1era y 3era entrada son entradas de datos, mientras que la 2da entrada es la entrada de control. La entrada de control para este sub-bloque depende del valor de *HOMEP* y *HOMEPF*.

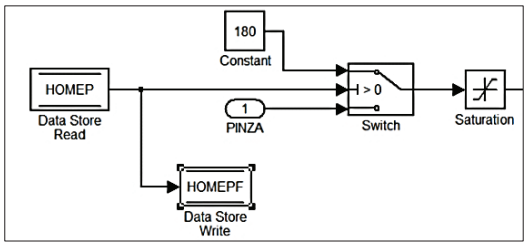


Figura 3.36 – Esquema para el bloque SWITCH

A su vez, la entrada de control debe cumplir la condición (>0), de modo que:

- Si la condición que se envía a la entrada de control es Verdadera, el programa selecciona la parte superior, es decir 180.
- Si la condición que se envía a la entrada de control es Falsa, el programa selecciona la parte inferior, es decir *PINZA*.

Como se ha definido, la variable *HOMEP*, se encarga de dar la señal para abrir la pinza, por lo tanto, su valor está directamente relacionado con las funciones a realizarse en este sub-bloque *SWITCH.* De tal manera, si $HOMEP = 1$, el selector compara esta entrada con la condición (>0). Y al cumplirse, el selector toma la opción de arriba, con lo cual se consigue enviar una constante con valor de 180, que quiere decir 180°, hacia el sub-bloque Saturación que envía sólo los 167° en su variable *Upper Limit* para el giro del motor y finalmente, se produce la apertura de la pinza. Físicamente en la estructura del prototipo, no se dispone de un sensor que envíe una señal en el instante que la pinza se ha abierto, o lo que quiere decir, que nos indique que la pinza ya ha realizado su proceso *HOME*; y por esta razón, se ha definido a la variable *HOMEPF* que nos permite obtener tal señal e indicar en la programación que dicha acción ya se ha desarrollado.

Cuando $HOMEP = 0$, el selector escoge la opción de abajo, y con esto, la señal de entrada de la variable *PINZA* se conecta a la entrada del proceso *PINZA*, para proseguir con el *Sub-bloque Saturación* y del mismo modo, al final enviar otro valor angular definido para la pinza en cualquier instante de la programación.

3.4.4 PROCESOS DE CONTROL

Se han definido como procesos de control a aquellos que proporcionan una secuencia de funcionamiento, o bien, pequeños sistemas de lazo cerrado; que dependen de una señal de control para comenzar a desarrollar algún proceso. Por esta razón, se han empleado máquinas de estado que proporcionen dicha secuencia dependiendo del proceso a realizar. En la figura 3.37 se indica el esquema de procesos que han sido empleados en la programación.

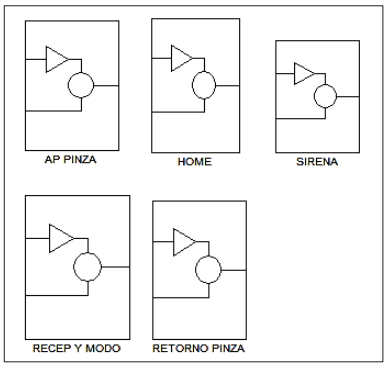


Figura 3.37 – Procesos de Control empleados en la programación

3.4.4.1 Proceso HOME

El proceso HOME permite desarrollar la secuencia necesaria para cumplir el ciclo de funcionamiento del puente grúa que se requiere para realizar el proceso *HOME* descrito en la lógica de control del prototipo. En la figura 3.38 se muestra el bloque interno perteneciente a este proceso.

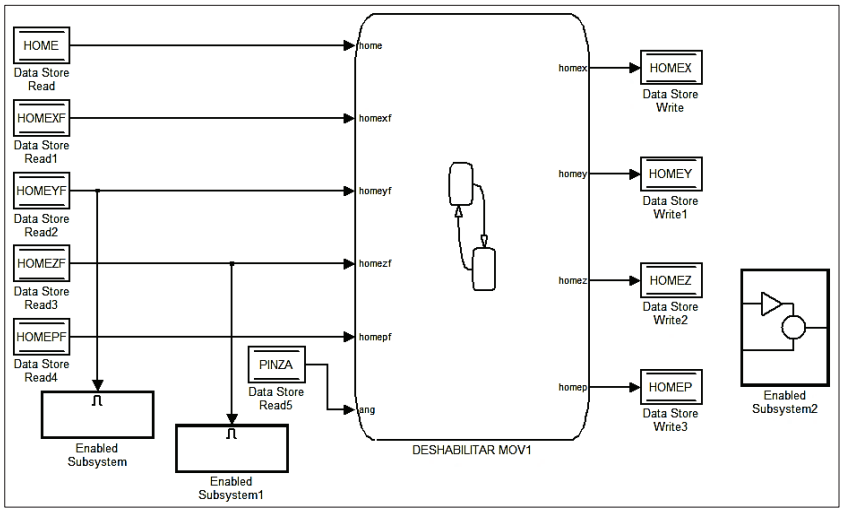


Figura 3.38 – Bloque interno del Proceso HOME

Como se observa, tenemos como entradas a *HOME, HOMEXF, HOMEYF, HOMEZF* y *HOMEPF*. Donde *HOME* es la bandera que envía la señal para que todos los ejes se dirijan al espacio de seguridad HOME, según la lógica de control del prototipo. Mientras que *HOMEXF*, *HOMEYF*, *HOMEZF* y *HOMEPF*, proporcionan la señal en la programación para indicar que la secuencia del proceso *HOME* que desarrollan cada uno de los ejes ya ha finalizado. Y en

cuanto a las salidas tenemos: *HOMEX, HOMEY, HOMEZ y HOMEP*. Éstas, son banderas que indican la inicialización del proceso HOME, de modo que, el proceso que debe cumplirse en cada uno de los ejes empiece a realizarse.

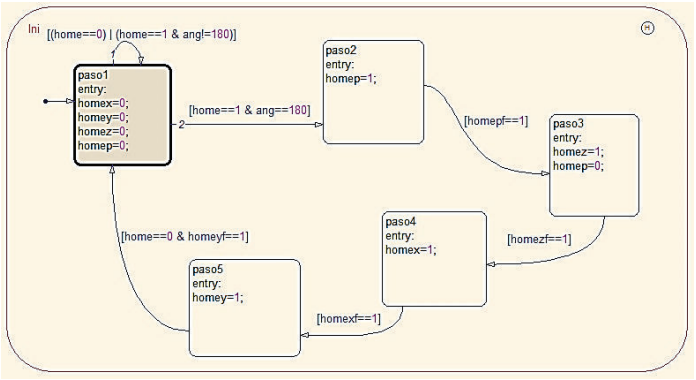


Figura 3.39 – Máquina de estados DESHABILITAR MOV1 (Proceso HOME)

Internamente la máquina de estados cumple el siguiente proceso:

- Paso1: Mientras home==0, es decir que no se ha enviado la señal para inicializar el proceso *HOME*, y si $home == 1 \& ang! = 180$ (diferente de 180°), con lo cual se entendería que se ha enviado la señal mediante la Tablet, pero el ángulo enviado a la pinza no es de 180°. A lo cual, se tiene que *homex*=0, *homey*=0, *homez*=0, *homep*=0; o lo que quiere decir que no se envía la señal para realizar el proceso *HOME.*
- Paso2: Cuando $home == 1 \,\&\, ang == 180$, lo que se entendería que se ha enviado el valor de 180° a *ANG* y correspondientemente a la variable *PINZA.* Con lo cual, se envía la señal a *homep*=1 para que el proceso de apertura de la pinza se realice completamente.
- Paso3: Después que se haya cumplido la condición h*omepf*==1 que quiere decir que la pinza ya se ha abierto; se envía la señal *homez*=1, con la finalidad de que la pinza comience a ascender y también se ha vuelto a enviar la señal *homep*=0, por motivos de seguridad para evitar que se vuelva a iniciar la secuencia. Cuando la pinza termina su ascenso, el fin de carrera envía una señal, con lo cual se tiene *homezf*==1, que indica que el proceso *HOME* para el eje Z ya ha terminado.
- Paso4: Con dicha señal, se habilita la señal *homex*=1, la misma que se encarga de dar la señal para permitir el regreso de pasos de todo el

recorrido realizado en el eje X hasta cero. Una vez terminado el proceso se activa *homexf*==1

- Paso5: La señal de *homexf*, en esta secuencia habilita a *homey*=1, logrando enviar la señal que hace regresar todo el recorrido del eje Y y cuando actúa el fin de carrera, se activa *homeyf*==1. Con lo cual, se mantiene en el ciclo del Paso1 hasta volver a habilitar el proceso *HOME* y el valor en la variable *PINZA* sea 180.

3.4.4.1.1 Subsistemas de Habilitación en el proceso HOME

Con el propósito de mantener la seguridad en casos de mala manipulación en los controles de la Tablet, y para evitar que existan fallas en la secuencia de funcionamiento del prototipo, se han determinado subsistemas de habilitación (*Enabled Subsystem),* los cuales se detallan a continuación.

- ***Enable Subsystem***

Sirve para indicar que ya se han finalizado las posiciones del proceso *HOME* para cada uno de los ejes, y a la vez, para verificar que cada uno de los ejes sea 0. La figura 3.40 muestra la parte interna de este subsistema.

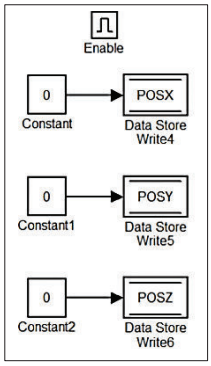


Figura 3.40 – Parte interna del Subsistema (Enabled Subsystem)

El subsistema toma la señal del eje Y, que es el último eje en activarse y dirigirse al espacio *HOME*. Con la señal de *homeyf*==1, se indica en el programa que el proceso *HOME* en el eje Y ha finalizado e internamente, el subsistema toma la señal y envía la constante 0 a las variables *POSX*, *POSY* y *POSZ*. Con tal acción, se consigue resetear o poner en cero a las posiciones e indicar que finalmente cada uno de los ejes ha llegado al espacio de seguridad *HOME*.

- ***Enable Subsystem 1***

En el proceso HOME, la pinza debe cerrarse inmediatamente después de accionar el fin de carrera ubicado al final del ascenso en el eje Z. En ese punto, la variable *homezf* se activa (es decir, *homezf*==1). Con esta señal el subsistema se habilita y envía una constante de valor 0 (es decir, 0°) a escribirse en la variable *ANG*, como se observa en la figura 3.41, y con lo cual, la pinza se cierra.

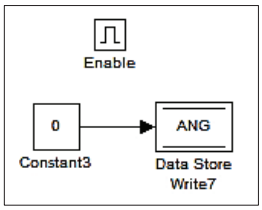


Figura 3.41 – Parte interna el Subsistema (Enabled Subsystem1)

- ***Enable Subsystem 2***

Este subsistema que se indica en la figura 3.42 se ha diseñado a fin de asegurar que la variable *HOME* sea cero, después de haberse cumplido la máquina de estados de dicho proceso.

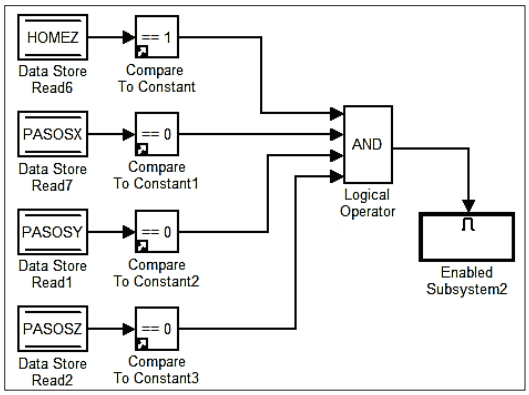


Figura 3.42 – Parte interna del Subsistema (Enabled Subsystem2)

Si se mantiene la señal *home*==1 y, si ésta señal es receptada por el programa, la secuencia determinada en la máquina de estados se podría volver a repetir y por lo tanto, es necesario limitar la posible activación de *HOME*. Para que esto se cumpla, se requiere poner *home*==0, y por ende deben cumplirse las condiciones definidas en la parte interna del mismo.

El proceso interno consta de 4 entradas o variables de lectura, las cuales son: *HOMEZ*, *PASOSX*, *PASOSY* y *PASOSZ*; las mismas que van a compararse con una relación definida. Obteniéndose así, las siguientes condiciones:

- El valor obtenido en la variable *HOMEZ* va a compararse con la expresión (==1). Cuando, *HOMEZ==1*, se entiende que la pinza ya ha alcanzado a subir todo el eje Z.
- Las señales de las variables *PASOSX*, *PASOSY* y *PASOSZ*; se comparan con la expresión (==0). Si se cumple la condición en cada uno de ellas, se daría a entender que el puente ya ha alcanzado la posición 0 en todos sus ejes y se encuentra ubicado en el espacio de seguridad *HOME*.

Al cumplirse estas condiciones el operador lógico *AND* envía una señal y activa a *Enabled Subsystem2*, que internamente envía una constante con valor 0 a escribirse en la variable *HOME*, como se indica en la figura 4.43. De tal forma la variable *HOME* se desactiva, hasta que nuevamente en cualquier instante del funcionamiento, se ordene mediante la Tablet realizar el proceso *HOME*.

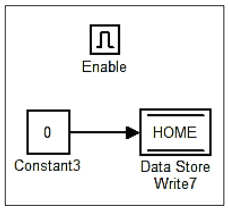


Figura 3.43 – Parte interna del Sistema Habilitador (Enabled Subsystem2)

3.4.4.2 Proceso RECEP Y MODO

Permite que el programa pueda receptar las señales enviadas desde la Tablet y a la vez, declarar el modo automático o manual de desplazamiento del puente en su funcionamiento. En el ANEXO A-5, se indica la parte interna correspondiente a este proceso, en donde se encuentran módulos de recepción de señales para cada una de las variables que deben ser codificadas por el microcontrolador para realizar las operaciones. El proceso de transmisión y encriptación de los datos enviados desde la Tablet se detalla posteriormente en el subcapítulo correspondiente al código fuente del programa para el software Android Studio.

A continuación, se detalla el funcionamiento de cada uno de los bloques.

3.4.4.2.1 Bloque de recepción VEL (USART1_Rx) [26]

Este bloque se encarga de recibir y leer los comandos o datos que se le ha enviado desde la interfaz gráfica de la Tablet. Dicho tramo de datos, poseen una cabecera y una terminación, formato que es receptado por el programa mediante una encriptación de datos, definida en los parámetros del bloque. En la figura 3.44, se indica el bloque de recepción *VEL*. Los parámetros principales que han sido definidos en este bloque, se indican en la figura 3.45.

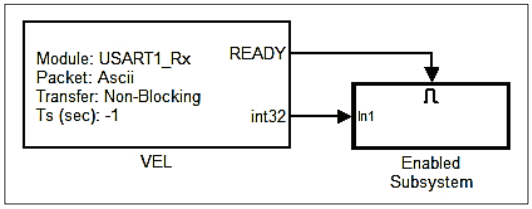


Figura 3.44 – Bloque de recepción VEL

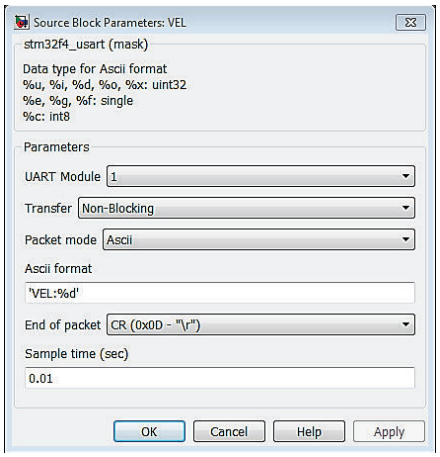


Figura 3.45 – Parámetros del bloque VEL (USART1-Rx)

- ***Packet Mode:*** Define el tipo de lenguaje o formato para hacer posible la encriptación de datos entre los dispositivos.
- ***ASCII Format:*** Es la forma de encriptación/codificación de la recepción de los datos, conocida también como encabezado. Para este caso, se ha determinado el encabezado "VEL:%d". La expresión "%d" significa que los datos enviados desde el código fuente de la Tablet, son reconocidos como números enteros en el bloque.
- ***End of Packet:*** Representa la terminación de un tramo de datos, para indicar al bloque que se ha terminado de enviar el dato desde la Tablet. En este

caso la terminación para todas las variables de recepción en general es LF (0X0A- “\n”).

Como se observa en la figura 3.44, cuando el dato es receptado por el bloque de recepción VEL (USART1_Rx) inmediatamente la salida *READY* (Listo) pone su valor lógico en 1, para indicar que ya se ha recibido un dato y a la vez, para habilitar al subsistema habilitador (*Enabled Subsystem*). Por la salida int32 se envía el dato que se ha recibido en el bloque de recepción y luego, este dato se dirige a la entrada de *Enable Subsystem*, es decir In1. Para finalmente, enviarlo al bloque *Convert* donde se cambia el tipo de dato *int32* a tipo *double* y escribirlo en el espacio de memoria perteneciente a VEL, como se indica en la figura 3.46.

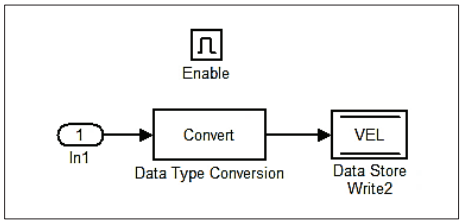


Figura 3.46 – Proceso interno para Enable Subsystem (Bloque VEL)

Se cumple el mismo proceso para el caso de los bloques de recepción *ANG*, *HOME*, *MMANU* y *NANG*, que se observan en el ANEXO A-5. Sin embargo, cada uno de éstos, posee un encabezado diferente que corresponde a la variable a controlar, pero en general una misma terminación. En la sección 3.5.3.2 de este capítulo, se detalla en una tabla la encriptación correspondiente a las variables que se emplean para la recepción de los datos desde la Tablet.

3.4.4.2.2 Modo Manual

En la figura 3.47, se muestra el esquema de funcionamiento de la variable MMANU, en el proceso *RECEP y MODO*.

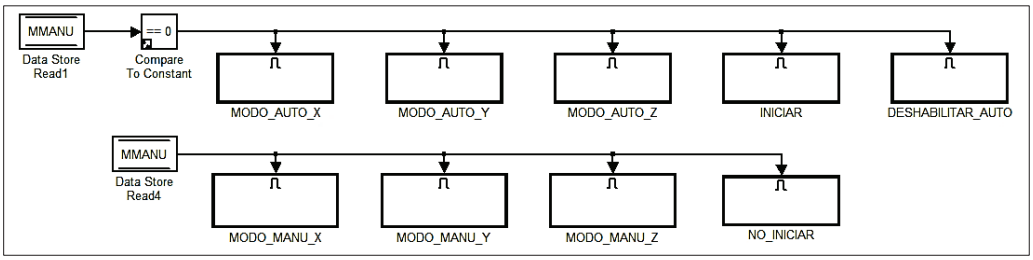


Figura 3.47 – Esquema del proceso para MMANU

La variable *MMANU* define el modo de posicionamiento manual o automático del puente grúa. Para esto, la variable cumple dos estados: Cuando *MMANU*==0, el posicionamiento se realiza de forma automática; caso contrario, cuando *MMANU*==1, el posicionamiento se realiza en forma manual. A continuación, se detalla la lógica de control cuando se cumple el posicionamiento en modo manual. En la figura 3.48, se indica la parte interna del esquema para el desarrollo del modo manual correspondiente al eje X, *MODO_MANU_X*.

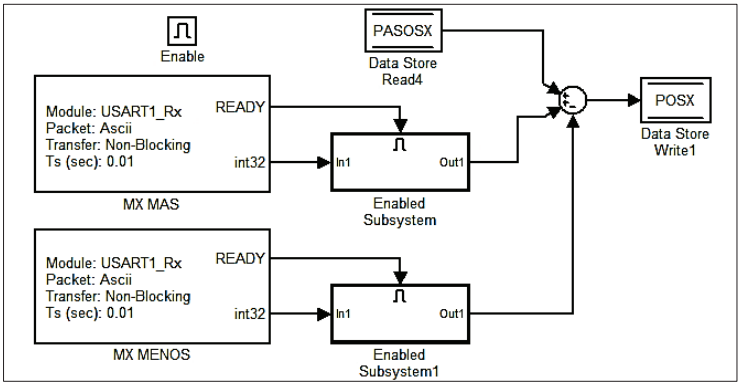


Figura 3.48 – Subsistema habilitador MODO_MANU_X

En la lógica de control, se ha implementado un proceso tal que permite ir sumando un paso cada vez que se recepte un dato en los bloques de recepción *MX MAS* y *MX MENOS* a *PASOS X*, para ir aumentando o disminuyendo pasos en el recorrido del motor PAP en el eje X, hasta alcanzar a POSX. La variable *MXMAS* corresponde al botón *Izquierda* y *MXMENOS* corresponde al botón *Derecha*, como se indica en la sección 3.5.3.2.10 de este capítulo.

Para *MXMAS*, se tiene que si *MX*+ es 1, va a sumar pasos (cuando se pulsa el botón y si MX+ es 0, deja de seguir sumando (cuando se deje de pulsar el botón). De forma homóloga es para *MXMENOS*. De tal forma, se obtiene lo siguiente: *PASOSX* + *MXMAS* = *POSX* o también, *PASOSX* – *MXMENOS* = *POSX*, cualquiera de las dos formas deben alcanzar a *POSX*, que se entiende como la posición que el usuario elige para dirigirse al mantener presionados los botones desde la interfaz de la Tablet. Este proceso también se realiza para el eje Y (*MODO_MANU_Y*) y para el eje Z (*MODO_MANU_Z*) del modo manual.

El subsistema *NO INICIAR* que también forma parte del modo manual, se ha desarrollado por motivos de seguridad, para evitar que se realice algún proceso

de posicionamiento automático cuando se mantenga un estado manual. Para ello, se ha enviado la constante con valor 0 a la variable *INICIAR*, como se ve en la figura 3.49.

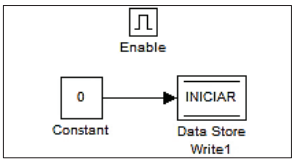


Figura 3.49 – Esquema interno Subsistema NO_INICIAR

3.4.4.2.3 Modo Automático

La figura 3.50 muestra el esquema interno del bloque *MODO_AUTO_X*, donde se observa un bloque de recepción denominado *POSX,* quien recepta el valor de la posición ingresada por la interfaz de la Tablet y la envía al subsistema habilitador *MODO_X*.

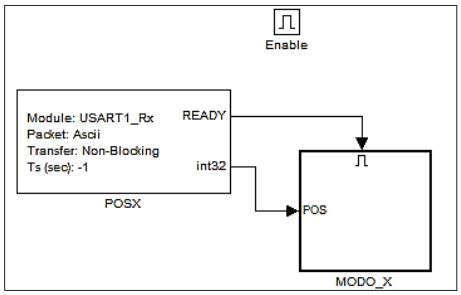


Figura 3.50 – Esquema interno del bloque MODO_AUTO_X

Internamente en el Subsistema *MODO_X*, se tiene la entrada *POS* que pasa por el bloque *Convert* (para cambiar el tipo de dato ingresado a *double)*, y se lo envía al bloque *GAIN* que posee la expresión 100/55. La misma que equivale a los 0,55mm que representa un paso del motor paso a paso. (ver sección 3.4.2.1). La expresión se invierte para posibilitar la transformación de mm a pasos que necesita el motor PAP para desplazarse a lo largo de su recorrido. En la figura 3.51 se observa el esquema interno del subsistema habilitador *MODO_X*.

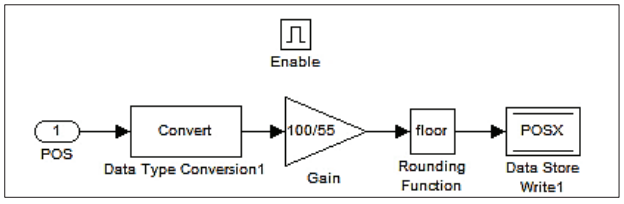


Figura 3.51 – Esquema interno Subsistema MODO_X

Después de la transformación, el dato se dirige a *Rounding Function (Función de Redondeo),* cuya finalidad es redondear el valor obtenido de la transformación a un número entero de pasos. Este dato se escribe en la variable POSX, para conseguir el movimiento del motor PAP en el eje X.

El mismo procedimiento se cumple para el caso del eje Y (*MODO_AUTO_Y*), como para el eje Z (*MODO_AUTO_Z*), excepto que para este último, se emplea el inverso del valor resultante de la transformación de pasos a milímetros realizado (sección 3.4.2.2), obteniéndose el valor de 200/37 para *GAIN* como se muestra en la figura 3.52.

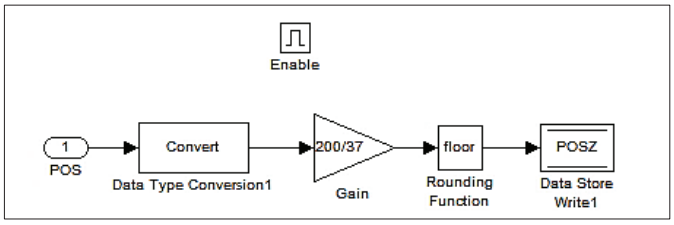


Figura 3.52 – Esquema interno Subsistema MODO_Z

El subsistema habilitador *INICIAR* correspondiente también al modo automático, está conformado por un bloque de recepción *INICIAR* y un subsistema de habilitación denominado *INI*, como se indica en la figura 3.53. Su funcionamiento consiste en convertir el dato de *int 32* receptado en el bloque de recepción a un tipo de dato *double* para escribirlo en la variable *INICIAR* del programa.

La señal proporcionada desde la Tablet, correspondiente a la variable INICIAR permite que el posicionamiento automático de todos los ejes se realice.

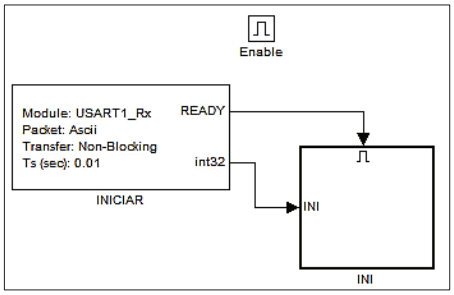


Figura 3.53 – Esquema interno del Subsistema habilitador INICIAR

Finalmente, el subsistema habilitador *DESHABILITAR_AUTO* posee un proceso interno que permite desactivar el modo de posicionamiento automático,

anteriormente empleado para hacer posible el ingreso de una nueva posición a la cual el usuario determina que el puente grúa se dirija. La figura 3.54 muestra el esquema interno correspondiente, el mismo que cuenta con dos subsistemas habilitadores internos que son: *DES_AUTO* y *COGER_PIEZA*, además de una relación lógica que sirve para verificar que el puente grúa ya ha finalizado todo su recorrido al llegar a cierta posición (X, Y, Z), a la cual se le ha ordenado dirigirse mediante la interfaz de la Tablet. Si se cumple que *PASOSZ==POSZ*, *PASOSX==POSX* y *PASOSY==POSY*, el operador lógico *AND* envía una señal de habilitación a los dos subsistemas internos.

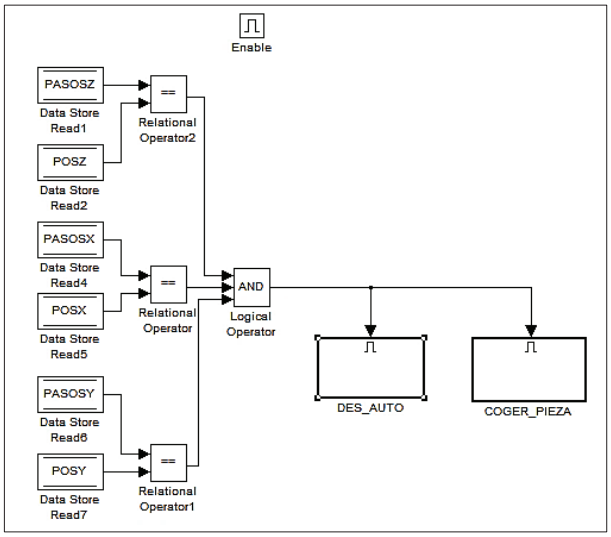


Figura 3.54 – Esquema interno Subsistema Habilitador DESHABILITAR_AUTO

El subsistema interno *COGER_PIEZA,* se encarga de recoger el objeto y desarrollar la lógica de control que necesita cumplir la pinza para el transporte del objeto hacia otra posición. Inicialmente, el subsistema está esperando tomar la pieza, con lo que se obtiene lo siguiente:

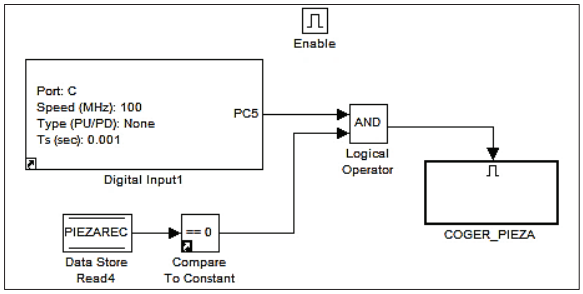


Figura 3.55 – Esquema interno del Subsistema COGER_PIEZA

En la figura 3.55 la señal que recepta la entrada (*Digital Input 1)*, corresponde a una señal proporcionada por el fototransistor (receptor), ubicado en uno de los dedos de la pinza a causa de una interrupción en la continuidad de la señal enviada por el led emisor de luz infrarroja, debida a la presencia de un objeto.

La variable *PIEZA_REC* indica que la pinza aún no ha tomado algún objeto y cuando se ha detectado la presencia de un objeto en la entrada, se prosigue entonces a cerrar la pinza automáticamente, tal proceso se lo realiza en el subsistema habilitador *COGER_PIEZA*. La figura 3.56 muestra el esquema interno correspondiente al subsistema habilitador *COGER_PIEZA.*

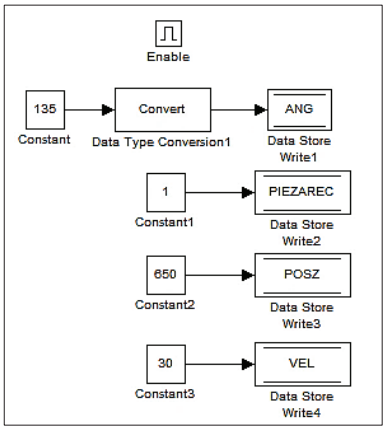


Figura 3.56 – Esquema interno Subsistema habilitador COGER_PIEZA

La constante 135 se envía al bloque *Convert,* y posteriormente, éste se escribe en la variable *ANG* de manera que el servomotor acoplado al eje se mueve los 135°, para cerrar la pinza y tomar el objeto. Luego, la constante 1 se envía a la variable *PIEZA REC*, para indicar al programa que la pinza ya ha tomado el objeto y prosiguiendo, se envía una *Constante2* de 650 a *POSZ*, a fin de que de que la pinza ascienda a la mitad del recorrido total del eje Z (650 pasos) inmediatamente después de haber tomado el objeto. Desde este punto, el objeto se traslada hacia una nueva posición ingresada por el usuario.

Por último, la *Constante3* de valor 30 se envía a la variable *VEL*, para que el proceso de desplazamiento en el traslado con carga se lo realice con velocidad baja. Esto, como medida de seguridad para el funcionamiento del puente grúa.

El subsistema habilitador *DES_AUTO* de la figura 3.54 se encarga de desactivar el modo automático para poder ingresar una nueva posición.

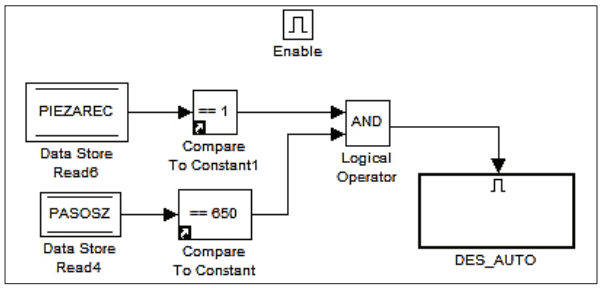


Figura 3.57 – Esquema interno Subsistema Habilitador DES_AUTO

La figura 3.57 muestra que este subsistema depende de las mismas condiciones señaladas para el subsistema *COGER_PIEZA*. De modo que, la señal obtenida en *PIEZA REC* y *PASOS Z*, al compararse con las condiciones (==1) y (==650) respectivamente, y si ambas se cumplen, el operador lógico *AND* envía la señal al habilitador del *subsistema habilitador DES_AUTO* que proporciona la señal de activación, y por ende, envía una constante con valor a 0 a la variable *INICIAR* para que sea posible ingresar una nueva posición, a la cual el puente grúa va a dirigirse. En la figura 3.58, se puede apreciar dicho proceso.

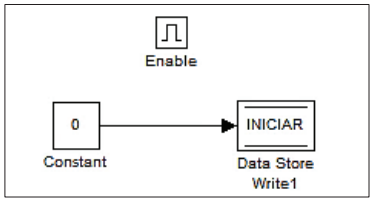


Figura 3.58 – Esquema interno Subsistema habilitador DES_AUTO

3.4.4.3 Proceso AP PINZA

Internamente, su proceso se encarga de realizar todas las acciones necesarias para realizar la apertura de la pinza, el esquema implementado para dicho proceso se muestra en el ANEXO A-6. Como se observa, se tiene un bloque llamado *Elapsed Timer*, que actúa como contador y permite realizar la cuenta de los 4 segundos para habilitar la apertura de la pinza. Este bloque posee como entradas a *Run/Stop*, *Reset* y como salida a *µsecs* (quien, proporciona la cuenta en microsegundos). A continuación se detallan las condiciones para su funcionamiento.

- ***Run/Stop***: Si es 1 se inicia la cuenta; caso contrario, la cuenta se detiene.

- ***Reset:*** Si es 1 permite resetear la cuenta y empieza a contar; y si es 0 permite seguir contando.

Se ha establecido en la programación las siguientes condiciones de seguridad que deben cumplirse para permitir la cuenta en el bloque:

- Las variables PASOSX, PASOSY y PASOSZ han alcanzado las posiciones POSX, POSY y POSZ, lo cual implica, que el puente grúa se ha detenido hasta la posición determinada por el usuario en cada uno de los ejes.
- Las banderas MOVX, MOVY, MOVZ, que indican el movimiento de alguno de los motores en cada uno de los ejes, no deben activarse. Pero en el caso de que sólo una de ellas se active, como medida de seguridad, el operador lógico *NOR* envía la señal para deshabilitar la cuenta.
- Cuando la variable *ANG* sea igual a 180.

Al cumplirse tales condiciones, el operador lógico AND envía 1 al bloque *Convert* (*Data Type Conversion*), que cambia la entrada de tipo *double* a tipo *int8*, para que el bloque pueda leer el dato a sus entradas y realice la cuenta. Es decir, se realizaría el mismo proceso de convertir pero de modo contrario.

En cambio, para resetear o deshabilitar la cuenta se debe tener en cuenta que el ángulo de la variable *ANG* sea diferente (~=) de 180, con dicha señal se habilita al subsistema habilitador *Enable Subsystem1*, que permite escribir dicho valor en la variable *PINZA*. Otra manera de resetear la cuenta es la variable *NANG*, que indica cuando se ha dejado de presionar el botón correspondiente desde la interfaz gráfica. Con tan sólo una de estas dos señales ocurre el mismo proceso para enviar la información a la entrada *Reset* del bloque.

Con una señal de habilitación comienza a correr la cuenta de los 4 segundos (es decir, 4000000 µsecs), al finalizar la cuenta se habilita *Enable Subsystem2*, quien escribe en la variable *PINZA* la constante 180, según se indica en la figura 3.59.

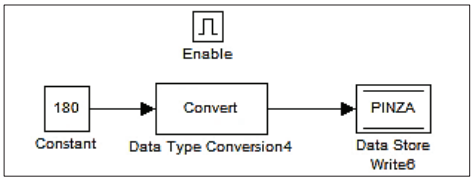


Figura 3.59 – Esquema interno para Enable Subsystem2

Con dicho valor en la variable *PINZA*, se cumple la relación (==180) y también, cuando la variable *PIEZAREC* indique que ya se ha recogido el objeto, se cumple que *PIEZAREC=1.* Como cada uno de los ejes ha alcanzado la posición, se habilita entonces el subsistema habilitador *SOLTAR_PIEZA,* el cual internamente, envía la constante con valor 0 a las variables *INICIAR* (para admitir el ingreso de una nueva posición), *PIEZAREC* (para indicar que ya ha dejado de recoger la pieza), *POSZ* (para volver a la posición del eje Z en 0); y la constante con valor 10 a *VEL,* para que el puente grúa se desplace a velocidad normal, al estar ya sin carga. Esto, se indica en la figura 3.60.

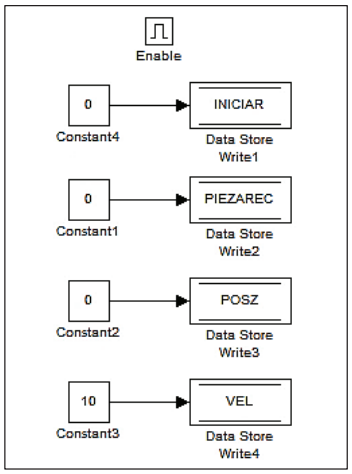


Figura 3.60 – Esquema interno Subsistema habilitador SOLTAR_PIEZA

3.4.4.4 Proceso RETORNO PINZA

Este proceso se ha desarrollado con la finalidad de permitir el retorno automático de la pinza al espacio de seguridad *HOME*, sólo si no se lo ha empleado por el lapso de 2 minutos. El esquema completo para este proceso se lo observa en el Anexo A-7. A continuación, se detalla el proceso de funcionamiento:

Se envía una constante (con valor igual a 1) al bloque *Convert*, con la finalidad de convertir el tipo de dato de *double* a *int8* para ser receptado por la entrada *Run/Stop* del bloque *Elapser Timer.* El contador vuelve a resetearse después de que ha transcurrido un tiempo mayor o igual a 1 segundo (>=1000000μsecs). Esto, con la finalidad de crear un generador de onda cuadrada, para simular una señal de reloj de 1 segundo. Para ello, se ha desarrollado un comparador donde

se condiciona lo siguiente: sí, la cuenta es mayor a 5 segundos (>500000μsecs) se envía 1, pero si ésta es mayor o igual a 1 segundo (>=1000000μsecs) se envía 0. De ese modo, se genera una onda cuadrada con ½ segundo en alto y ½ segundo en bajo.

Adicionalmente, el proceso está conformado por una máquina de estado denominada *CUENTA*. Para que su entrada *contar* sea igual a 1 (es decir, se active); se deben cumplir las condiciones:

- Cuando el valor del ángulo en la variable *PINZA* sea igual a 180°, es decir, siempre que la pinza se encuentre totalmente abierta. Por tanto, la pinza puede dirigirse al espacio *HOME*, sólo al permanecer en dicho estado
- Si se cumple que *MOVX, MOVY* o *MOVZ* sean igual a 0, implicando que no existe movimiento de los motores en cada uno de los ejes.

Una vez habilitada la entrada *contar*, se detalla el proceso de funcionamiento para la máquina de estado *CUENTA*, cuyo esquema se muestra en la figura 3.61.

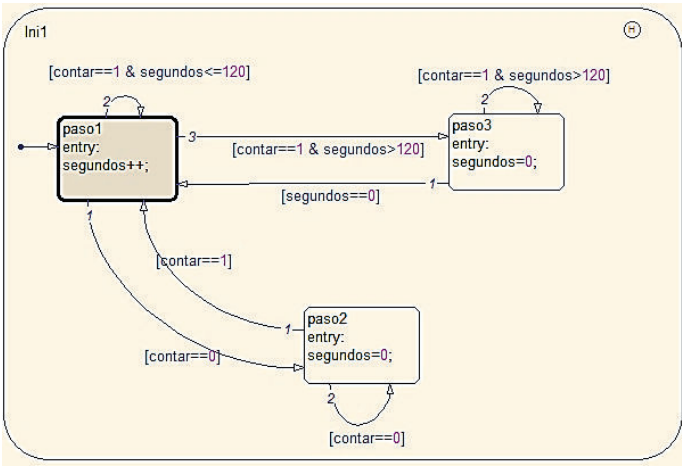


Figura 3.61 – Máquina de estados CUENTA (Proceso RETORNO PINZA)

- Si se cumple que $contar == 1 \,\&\, segundos <= 120$ (es decir, 2 minutos), los segundos continúan aumentando hasta alcanzar los 120 segundos.
- Cuando la variable *contar*==0, entonces *segundos*=0, y cuando *contar*==1, se habilita el paso1, con lo cual se realiza nuevamente el proceso. Es decir, la cuenta se reinicia.

- Al cumplirse las condiciones $contar == 1 \,\&\, segundos > 120$, entonces *segundos*=0. Lo que resultaría el resetear la cuenta, para regresar al paso1 y empezar la cuenta; siempre y cuando, la variable *contar* sea igual a 1.

Cuando se cumpla la condición *(==120 segundos)* se habilita un 1L a una de las entradas del operador lógico *AND* y a la vez, este dato se dirige hacia un bloque *Memory,* que sirve para desplazar un período al envío de datos, ya que en un momento la variable de la señal de los 120 segundos se pone en 0, pero por la función de dicho bloque, se mantiene el 1 anterior, con lo cual *Home* vuelve a ponerse en 0. Esto, permite mantener sólo por un momento habilitada a la variable *HOME*, ya que no es necesario mantenerla habilitada siempre. La señal que llega para habilitar al Subsistema habilitador *Enable Subsystem1*, va a emplear dicha acción para enviar la constante con valor 0 a escribirse en la variable *HOME*, como se observa en la figura 3.62.

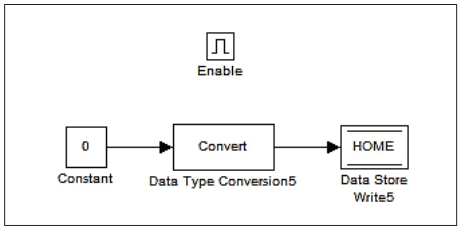


Figura 3.62 – Esquema interno Subsistema Habilitador Enable Subsystem1

Al habilitarse las señales en las dos entradas, el operador lógico *AND* se activa y su señal, habilita al Subsistema Habilitador *Enable Subsystem*, cuya finalidad es enviar una constante con valor 1 a la variable *HOME* para que se realice el proceso *HOME* en los tres ejes, como se observa en la figura 3.63.

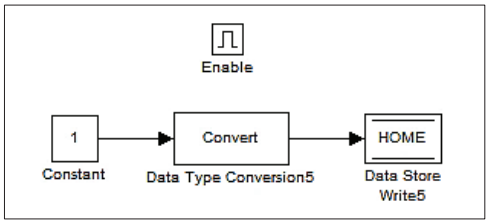


Figura 3.63 – Esquema interno Subsistema Habilitador (Enable Subsystem)

El operador *NOT* sirve como medida de seguridad para evitar que la acción de ambos procesos (escribir 1 y 0 en la variable *HOME*) se realicen a la vez.

3.4.4.5 Proceso SIRENA

En la figura 3.64, se observa el esquema interno correspondiente a este proceso. Su finalidad es activar un Buzzer a una de las salidas del microcontrolador, para indicar al usuario mediante una señal auditiva que se está utilizando el puente grúa. Para ello, se emplean las banderas *MOVX*, *MOVY* y *MOVZ*, que indican cuando se está moviendo el eje X, Y o Z en el funcionamiento del puente grúa. Se emplea un operador lógico OR, para receptar cualquiera de estas señales, mientras que, el bloque *Memory* junto al bloque *SUM*, tienen como finalidad sumar algebraicamente la señal obtenida en la entrada al dato anterior, para mantener una señal continua en la entrada OR, que en este caso es 1L. Esto resulta ser un método de enclavamiento hasta conseguir un 0L en cualquiera de las banderas *MOVX*, *MOVY* y *MOVZ*. [44]

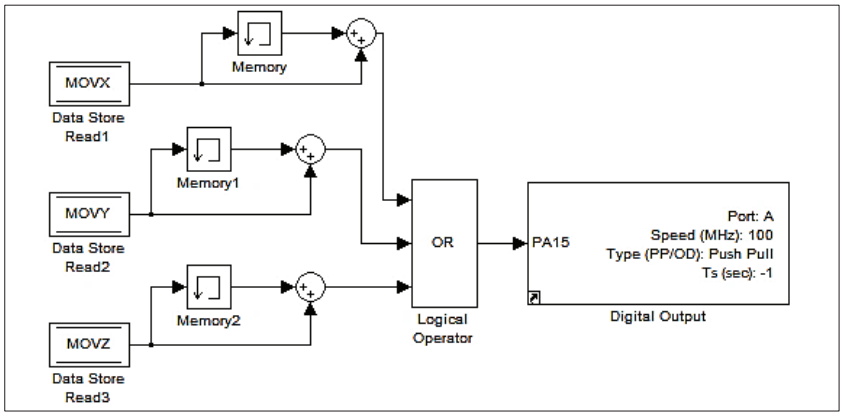


Figura 3.64 – Esquema interno proceso SIRENA

3.5 INTERFAZ GRÁFICA PARA EL CONTROL INALÁMBRICO

El desarrollo de la interfaz gráfica para el control inalámbrico del prototipo, comprende el uso del software *Android Studio 1.0.2*, el cual nos permite desarrollar aplicaciones para la plataforma *Android* que generalmente la manejan la mayor parte de dispositivos móviles. Las líneas de comando empleadas para esta aplicación, se las analiza conforme se vaya desarrollando la programación para la interfaz gráfica detallada en este subcapítulo.

3.5.1 ENTORNO DEL SOFTWARE ANDROID STUDIO 1.0.2

El software Android Studio nos permite crear miles de aplicaciones con diferentes características entre sí, para la plataforma Android. La interfaz gráfica desarrollada mediante dicho software, comprende en su totalidad, sólo las funciones básicas de aquellas que generalmente proporciona la versión de la plataforma, en donde se implementa la aplicación creada mediante el software. La figura 3.65 muestra el entorno de trabajo del software Android Studio.

El entorno de trabajo en Android Studio comprende el uso y la adaptación de botones, sonidos, imágenes y también, el ingreso de texto, barras, entre otros controles especiales, que pueden ser empleados en la creación de aplicaciones.

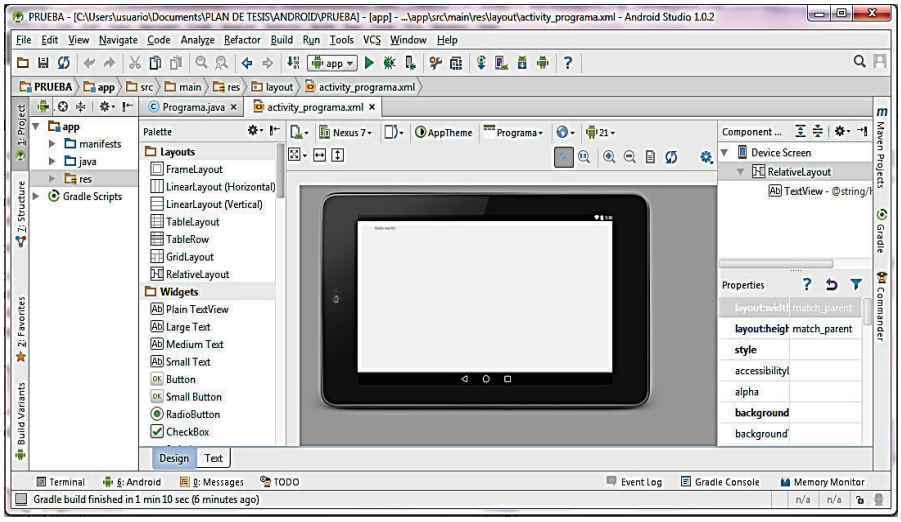


Figura 3.65 – Entorno de trabajo en el Software Android Studio

La inicialización para el desarrollo de la interfaz gráfica, luego de ingresar al entorno de trabajo en el Software Android Studio, comprende 4 pasos:

Paso 1: Definir el nombre de la aplicación (a criterio del Usuario), como se observa en la figura 3.66.

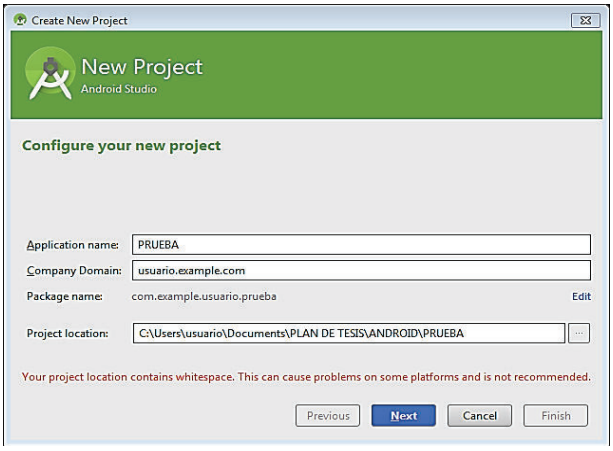


Figura 3.66 – Definición del nombre de la aplicación en Android Studio

Paso 2: Determinar el nivel API (o la versión de la plataforma), donde se desempeña nuestra aplicación. Tal como se mencionó en el Capítulo 1, la versión de la plataforma Android (o el nivel API) seleccionada al inicio de la creación del proyecto debe ser menor o igual a la versión Android que posee el dispositivo al que se va a instalar la aplicación desarrollada en el Software Android Studio. Esto, a fin de que la aplicación pueda soportar las características de la plataforma del dispositivo. Si no se cumple tal requerimiento, la aplicación no puede desempeñarse correctamente y perdería, parcial o totalmente, su funcionalidad en ciertos procesos.

Para el desarrollo de nuestra interfaz gráfica, se dispone de una Tablet *(Marca Speed Mind)*, con un Sistema Operativo Android 4.2. El nivel API correspondiente a dicha versión es API 17, pero se lo ha fijado al nivel API 15 versión de la plataforma para Android *4.0.3 (Ice Cream Sandwich)*, dos versiones menores tal como se recomienda. Sin embargo, esto no proporciona mayor funcionalidad, ya que nuestra aplicación puede desempeñarse muy bien en versiones menores. La figura 3.67 da a conocer el nivel de cobertura de esta versión de plataforma, mientras que la figura 3.68, muestra las características disponibles para el usuario

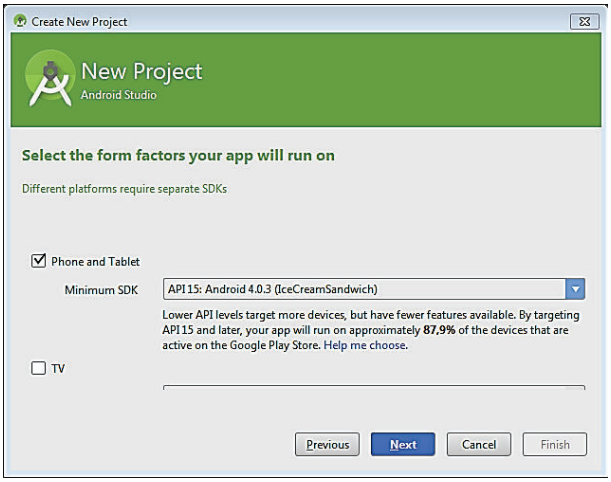


Figura 3.67 – Selección del API en el Software Android Studio

Si se requiere mayor información acerca de las aplicaciones disponibles para el usuario pertenecientes a esta versión, se recomienda visitar el sitio web: *https://developer.android.com/about/versions/android-4.0.html*

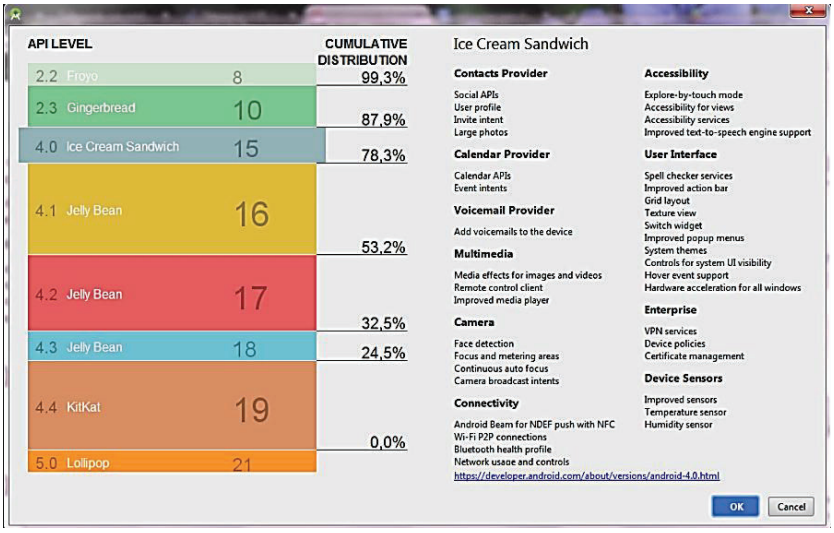


Figura 3.68 – Características disponibles para el usuario con un API 15

A pesar del extenso número de aplicaciones que dispone el nivel API 15, haremos uso de las más básicas para el proyecto, limitándonos a describir únicamente las herramientas que fueron empleadas en el desarrollo de la interfaz gráfica para el control inalámbrico del prototipo. El resto de aplicaciones no requieren de un estudio exhaustivo, por lo que se omite entrar en detalle.

Paso3: Selección de la pantalla donde se edita el nuevo proyecto. La figura 3.69 muestra el cuadro de selección de la pantalla donde se crea la interfaz. Se elige el formato más sencillo y básico para la programación, es decir, *Blank Activity* (Actividad en Blanco).

Figura 3.69 – Selección de la pantalla en el Software Android Studio

Paso 4: Selección del tamaño de la pantalla en la cual se va a realizar la aplicación, y que se selecciona, según la medida de la pantalla del dispositivo a emplearse. La figura 3.70 muestra el entorno que permite definir el tamaño del dispositivo, desde la pestaña *Nexus* disponible en las características del software.

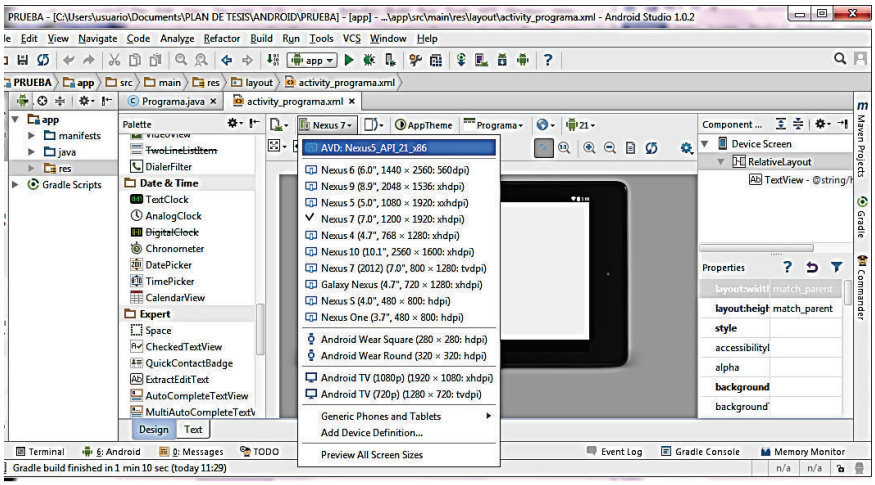


Figura 3.70 – Selección del tamaño de pantalla en Android Studio

Se seleccionó la opción Nexus 7 (7.0" ,1200 x1920: xhdpi), correspondiente al tamaño de la pantalla de 7" que posee la Tablet, según las características mencionadas en el capítulo 2 de este proyecto.

3.5.1.1 Herramientas básicas en Android Studio 1.0.2 [27] [45]

Las herramientas básicas que sirven de gran utilidad para desarrollar aplicaciones Android son: botones, layouts y etiquetas de texto. A continuación, se proporciona un detalle de las más importantes, empleadas para la programación.

3.5.1.1.1 Botones (Button)

Son objetos descendientes de la clase View (parte visual de los componentes), y llegan a ser los elementos que componen la interfaz de usuario de una aplicación. A continuación, se detallan los tipos de botones:

- ***Toggle Button:*** Este tipo de botón una vez presionado, mantiene su estado hasta volverlo a presionar nuevamente. Se define por la posibilidad de mantener dos estados (presionado/no presionado) para su utilización.
- ***Switch:*** Este tipo de botón mantiene el mismo funcionamiento que el caso anterior y difiere de éste, porque visualmente se puede apreciar el cambio de estado en un mismo espacio.
- ***ImageButton:*** En este botón se define una imagen en lugar de un texto en su interior y cumple una función básica.

Estos objetos responden al evento denominado "*onClick"* en el código fuente para el desarrollo de la aplicación. De ellos, se emplean únicamente aquellos del tipo *ImageButton* y *ToggleButton,* debido a que las necesidades para nuestra interfaz gráfica, no requieren acciones adicionales que no correspondan a los tipos de controles anteriormente mencionados.

3.5.1.1.2 Imágenes, etiquetas y cuadros de texto

Estas herramientas junto a los elementos del tipo Button, proporcionan el desarrollo de aplicaciones simples pero muy útiles a la vez, las mismas que se detallan a continuación:

- ***Imágenes (ImageView):*** Con este elemento es posible visualizar imágenes en una aplicación, haciendo posible definir ciertas características, como por ejemplo: determinar el tamaño y la posición del texto.
- ***Etiquetas de texto (TextView):*** Sirven para establecer un formato de texto especial en la aplicación, con el fin de mostrarlo al usuario.

- ***Cuadro de Texto (Control EditText):*** Permite la introducción y edición de texto en cualquier aplicación, por parte del usuario.

Cada una de éstas se utilizan en el desarrollo de la interfaz gráfica, añadiendo además, posibles configuraciones que van a depender del tipo de proceso que se pretende conseguir con el control respectivo.

3.5.1.2 Espacio de Trabajo en Android Studio 1.0.2 [27] [28] [45]

La parte más importante en la creación de un nuevo proyecto desarrollado en Android Studio, comprende el espacio de trabajo (Workspace) que se observa en la parte izquierda de la pantalla. En este espacio, se tienen las carpetas del programa principal desarrollado y también, los recursos empleados en la interfaz gráfica, como se puede observar en la figura 3.71.

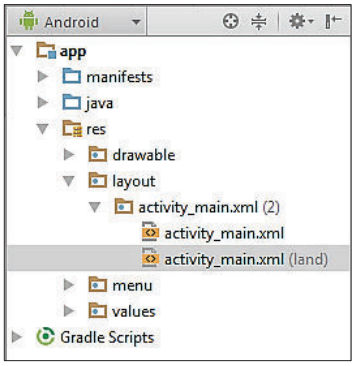


Figura 3.71 – Carpetas del programa empleados en la interfaz gráfica

Cada carpeta o módulo de aplicación, contiene los conjuntos de fuente completos para dicho módulo. En la carpeta principal, llamada "***app"*** del espacio de trabajo, se tienen todos los elementos correspondientes a la aplicación, y al desplegarla, se observa una subcarpeta llamada ***"Res"***, que contiene todos los recursos empleados en la interfaz del control (es decir, la pantalla) y que están almacenadas en la subcarpeta "***drawable"***, donde se encuentran los ficheros de imágenes que han sido modificadas en formato (JPG o PNG) y tamaño, para ser acopladas al espacio permitido por las dimensiones de la pantalla de la Tablet.

La subcarpeta ***Layout***, contiene ficheros XML con vistas de la aplicación, que nos permiten configurar las diferentes pantallas que conforman la interfaz de usuario de la aplicación. Al interior de esta subcarpeta, en ***activity_main.xml*** (land

activity), se puede observar el diseño de la interfaz realizada en la pantalla según se muestra en la figura 3.72. A su vez, en la figura 3.73 se aprecia una parte del diseño realizado con imágenes y cuadros de texto, en forma de código para poder ser compilado por el software.

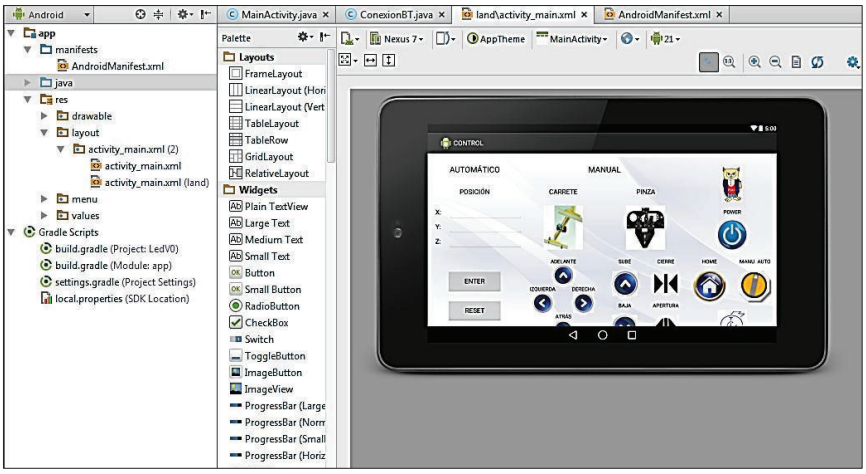


Figura 3.72 – Diseño de la interfaz gráfica realizada

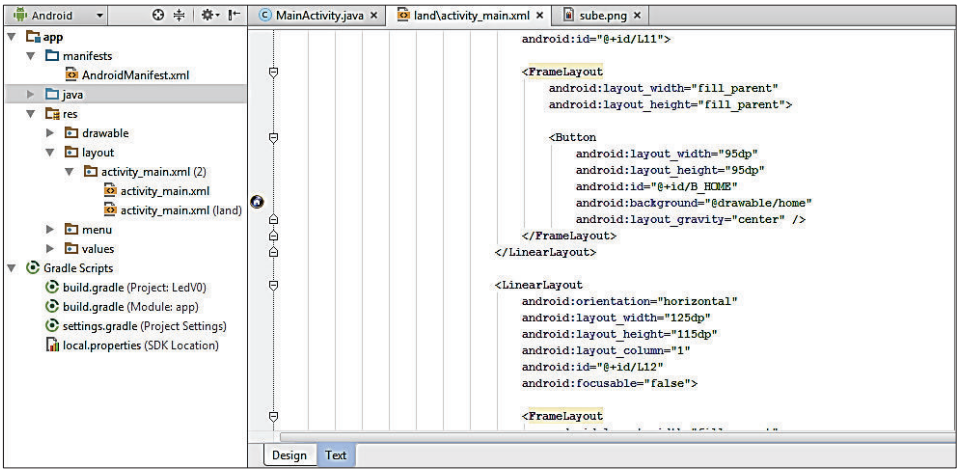


Figura 3.73 – Diseño de la interfaz gráfica desarrollada en forma de código

3.5.2 DESARROLLO DE LA INTERFAZ GRÁFICA

Está compuesto de dos partes, que son: Pantalla y Código fuente del Programa. Las mismas se describen en secciones posteriores a este subcapítulo.

3.5.2.1 Pantalla

El diseño de la interfaz gráfica implica el posicionamiento y modificación del tamaño, tanto en imágenes como en cuadros de texto; así como, la correcta disposición de los botones. El diseño se divide en tres partes: modo Automático, modo Manual y Mandos Proceso, como se indica en el esquema de la figura 3.74. La disposición de los controles en la interfaz gráfica se la observa completamente en el ANEXO B-1.

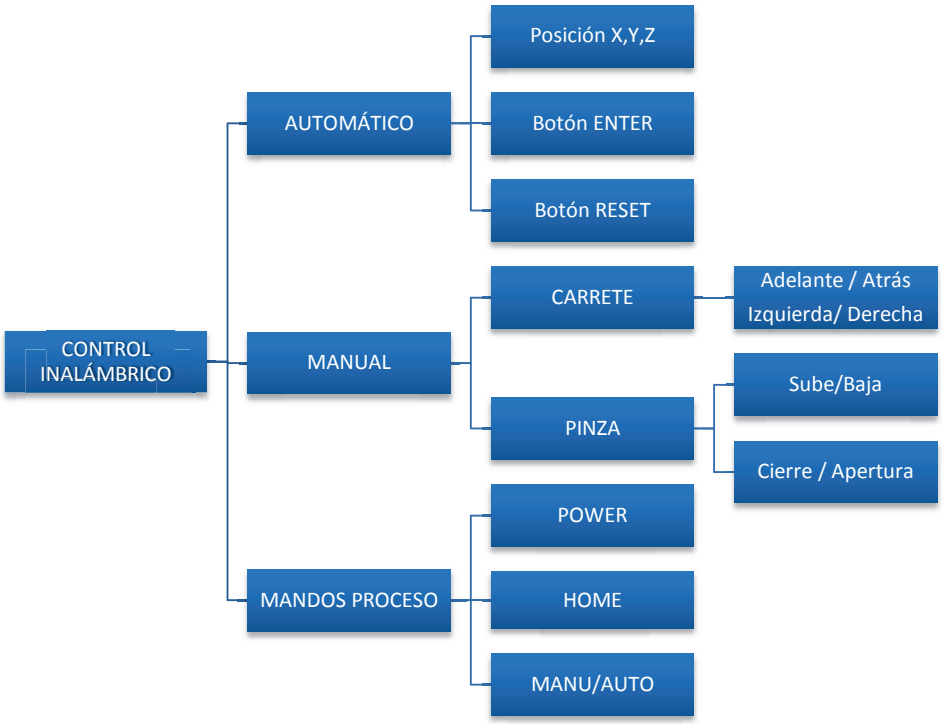


Figura 3.74 – Esquema estructural para la interfaz gráfica de la Tablet

Al emplear cuadros de edición de texto en una interfaz gráfica, o lo que sería una función ***Edit.Text***, para ingresar el texto se emplea el teclado del dispositivo en el cual se va a desarrollar la aplicación y entre los parámetros al momento de construir la interfaz, es posible configurar el tipo de caracteres o datos que se van a poder ingresar en dichos espacios. De modo que, con la opción ***InputType***, correspondiente a tales parámetros, se ha definido que sólo se puedan ingresar números enteros, restringiendo el ingreso de caracteres ajenos a éstos al momento de introducir los valores en la Tablet. [27]

3.5.2.2 Modo Automático

El modo automático para el control inalámbrico, permite el ingreso de la posición a la cual se quiere trasladar el puente mediante cuadros de texto, denominados **X:** , **Y:**, **Z:** ubicados bajo la etiqueta POSICIÓN de la interfaz gráfica. Adicional a esto, se dispone de dos botones, ENTER y RESET. El botón ENTER, puede oprimirse una vez ingresada la posición determinada por el usuario en cada uno de los cuadros de texto. Esto se realiza, aunque no se haya ingresado algún valor en cualquiera de las posiciones. El botón RESET, borra el contenido ingresado en los cuadros de texto, permitiendo al usuario ingresar nuevos datos.

3.5.2.3 Modo Manual

El Modo Manual consta de dos mandos, que son: CARRETE y PINZA. En el mando CARRETE, se han definido las funciones de avance, retroceso, izquierda y derecha, que corresponden a los ejes X,Y ; los mismos que, movilizan al puente dentro de toda la estructura del prototipo. Por otra parte, el mando PINZA, se encarga de controlar todo el sistema de movimiento y la acción de apertura/cierre de la pinza, por ende éste controla el desplazamiento del motor PAP en el eje Z.
De esta forma, el usuario puede controlar el desplazamiento de cada uno de los ejes con tan sólo mantener presionado el botón correspondiente.

3.5.2.4 Mandos Proceso

Entre los mandos de proceso, tenemos los botones POWER y HOME; y también el selector MANU/AUTO. El botón POWER debe presionarse inmediatamente después de haber establecido la conexión con el módulo Bluetooth HC-05 del microcontrolador STM32F407, su finalidad en el programa, es enviar un valor a la entrada de la señal de reloj creada en el Subsistema *GENERAR PASOS* para producir el desplazamiento de los motores PAP inicialmente en los ejes X, Y, Z. Por otra parte, el botón HOME proporciona la señal para que inicie el proceso HOME. En tanto que, el selector MANU/AUTO, permite al usuario definir el modo de posicionamiento, ya sea manual o automático, para el control inalámbrico.

3.5.3 CÓDIGO FUENTE DEL PROGRAMA [46]

El código fuente empleado para la programación comprende la forma de cómo se envía el dato desde la Tablet, y a su vez, la relación de funcionamiento de los controles y elementos para la transmisión de valores y señales hacia el microcontrolador. El código de programa para este caso se compone de dos partes, que son: *Conexión BT* (Bluetooth) y *Main Activity* (Actividad Principal).

3.5.3.1 Conexión BT

Esta sección comprende la configuración del módulo Bluetooth, para utilizar la aplicación desarrollada en Android Studio y la funcionalidad Bluetooth de la Tablet, a fin de establecer la comunicación serial entre éstos dispositivos.

El código fuente designado para la recepción y envío de datos desde la Tablet hacia el módulo Bluetooth, es una codificación de caracteres, actividades, parámetros y acciones que comúnmente se establecen para crear hilos de conexión, transmisión, recepción y desconexión comunmente usados en cualquier conexión Bluetooth desarrollada en lenguaje de programación java para Android.

En cuanto a la conexión, existen hilos o etapas de espera en los cuales el Bluetooth de la Tablet recepta los datos de transmisión desde los comandos de la aplicación y escribe las funciones a enviarse hacia el módulo Bluetooth del prototipo. A la vez, en el código fuente se desarrollan mensajes de texto que el usuario puede observar en la aplicación, como por ejemplo: “Conectado con” cuando ya se ha establecido la conexión con el módulo Bluetooth, “No conectado” cuando el módulo no se ha enlazado a algún dispositivo, “Error de conexión” cuando existe falla en la conexión establecida, y “Se perdió conexión”, cuando el usuario ha superado la distancia máxima a la cual se puede mantener la conexión Bluetooth entre dispositivos, tal y como se muestra a continuación.

```
//Instrucción para indicar que se ha establecido la conexión con el
módulo Bluetooth

public synchronized void connect(BluetoothDevice device) {
     if (D) Log.e(TAG, "Conectado con: " + device);
     if (EstadoActual == STATE_CONNECTING) {
     if (HiloDeConexion != null) {HiloDeConexion.cancel();
     HiloDeConexion = null;}  }
```

```
//Instrucción para indicar que si no se escribe algún dato para ser
enviado al servicio BT, se muestre el mensaje "No conectado"


public  void sendMessage(String message) {
  if (Servicio_BT.getState() == ConexionBT.STATE_CONNECTED) {
        if (message.length() > 0) {
         byte[] send = message.getBytes();
            if(D) Log.e(TAG, "Mensaje enviado:"+ message);
                 Servicio_BT.write(send); }}
else Toast.makeText(this, "No conectado", Toast.LENGTH_SHORT).show();
    }

//Instrucción para enviar el mensaje "Error de conexión" en la aplicación

    private void connectionFailed() {
        setState(STATE_LISTEN);
        Message msg = mHandler.obtainMessage(MainActivity.Mensaje_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(MainActivity.TOAST,"Error de conexi□n");
        msg.setData(bundle);
        mHandler.sendMessage(msg);    }


//Instrucción para notificar la acción "Se perdió conexión" en la
aplicación

    private void connectionLost() {
        setState(STATE_LISTEN);
        Message msg = mHandler.obtainMessage(MainActivity.Mensaje_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(MainActivity.TOAST,"Se perdio conexión");
        msg.setData(bundle);
        mHandler.sendMessage(msg);
        msg = mHandler.obtainMessage(MainActivity.MESSAGE_Desconectado);
        mHandler.sendMessage(msg);    }
```

Se ha empleado en las definiciones del código fuente un *Identificador Único Universal* (***UUID***, *Universally Unique Identifier*) perteneciente al módulo Bluetooth RN-42 propio de la Tablet para crear un código único empleado en una comunicación serial, con la finalidad de que se establezca el envío de los datos desde la Tablet y éstos, sean recibidos con dicho código en el módulo Bluetooth.

Se define en la programación el código *UUID* correspondiente al módulo RN-42, como se observa:

```
// Instrucción para configurar UUID para módulo BT RN42

 private static final String TAG = "Servicio_Bluetooth";
 private static final boolean D = true;
 private static final String NAME = "BluetoothDEB";
 private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-
8000-00805F9B34FB");
```

3.5.3.2 Main Activity

Esta sección de la programación, desarrolla el código fuente necesario para determinar la función que va a cumplir cada botón dispuesto en la interfaz gráfica. Para ello, se designa un formato ASCII, que posee un *"encabezado"* y una *"terminación"* para establecer el envío de datos desde la Tablet a la recepción de los mismos en el microcontrolador STM32F407.

Tabla 3.4 – Botones, variables correspondientes y sus encriptaciones

				ENCRIPTACIÓN	
	BOTONES / COMANDOS		VARIABLE	ENCABEZADO	TERMINACIÓN
MANDOS PROCESO	POWER		VEL	"VEL:%d"	\n
	HOME		HOME	"HOME:%d"	\n
	MANU/AUTO		MMANU	"MMANU:%d"	\n
MODO AUTOMÁTICO	ENTER		INICIAR	"INICIAR:%d"	\n
	POSICIONES	X	POSX	"POSX:%d"	\n
		Y	POS Y	"POSY:%d"	\n
		Z	POS Z	"POSZ:%d"	\n
MODO MANUAL	CARRETE	ADELANTE	MY+	"MY+:%d"	\n
		ATRÁS	MY-	"MY-:%d"	\n
		IZQUIERDA	MX+	"MX+:%d"	\n
		DERECHA	MX-	"MX-:%d"	\n
	PINZA	CIERRE	ANG	"ANG:%d"	\n
		APERTURA	NANG ANG	"NANG:%d" "ANG:%d"	\n \n
		SUBE	MZ-	"MZ-:%d"	\n
		BAJA	MZ+	"MZ+:%d"	\n

El primero, comprende la descripción del tipo de dato que va ser recibido por el microcontrolador, mientras que el segundo, permite indicar al microcontrolador que ya se ha finalizado de enviar el dato. En la Tabla 3.4, se muestran los botones y comandos de la interfaz gráfica, a los cuales, se van a enviar datos mediante un formato ASCII y su encriptación.

La encriptación de la recepción de los datos en el módulo Bluetooth HC-05 y el envío de datos desde la Tablet, corresponde a un código ASCII, de tal forma cada una de las variables mencionadas en la tabla se receptan en los bloques de recepción de los *Procesos de Control* realizados en el software SIMULINK.

Al poseer un encabezado con un formato correspondiente a ”%d”, significa que el dato enviado desde el código fuente se recepta como un entero en el bloque de recepción respectivo. De los comandos y botones que forman parte del control inalámbrico descritos en la tabla, se ha omitido al botón RESET (referente al Modo Automático), puesto que éste cumple una función independiente a la transmisión de datos que cumplen las anteriores. Puesto que, cumple sólo la función de borrar los valores ingresados en cada uno de los cuadros de texto correspondientes a los ejes X, Y, Z de la interfaz gráfica.

Continuando con el desarrollo del código fuente, primero se instancian las denominaciones o nombres de los botones que se van a manipular en la interfaz gráfica. Con la función ***OnCreate***, y mediante el método ***findViewById()*** se definen dichos elementos con el **Id** respectivo para que sean compilados en el código fuente del programa y a la vez, se define también el tipo de botón que se emplea con cada uno de ellos, tal como se indica a continuación:

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

      final ToggleButton BotonLed =
(ToggleButton)findViewById(R.id.B_MMANU);
        final Button bPower = (Button)findViewById(R.id.B_POWER);
        final Button bHome = (Button)findViewById(R.id.B_HOME);
        final Button bApertura = (Button)findViewById(R.id.B_APERTURA);
        final Button bBaja = (Button)findViewById(R.id.B_BAJA);
        final Button bCierre = (Button)findViewById(R.id.B_CIERRE);
        final Button bSube = (Button)findViewById(R.id.B_SUBE);
        final Button bAtras = (Button)findViewById(R.id.B_ATRAS);
        final Button bIzquierda = (Button)findViewById(R.id.B_IZQUIERDA);
        final Button bDerecha = (Button)findViewById(R.id.B_DERECHA);
        final Button bAdelante = (Button)findViewById(R.id.B_ADELANTE);
        final Button bEnter = (Button)findViewById(R.id.B_ENTER);
        final Button bReset = (Button)findViewById(R.id.B_RESET);
        final EditText nX = (EditText)findViewById(R.id.N_X);
        final EditText nY = (EditText)findViewById(R.id.N_Y);
        final EditText nZ = (EditText)findViewById(R.id.N_Z);
```

Los elementos ***EditText (nX, nY, nZ)*** corresponden a los cuadros de edición de texto definidos para los ejes X, Y, Z; en los que el usuario ingresa los datos.

A continuación, se detallan las acciones a cumplir para cada uno de los botones y su código de programación empleado en cada caso, cuando se pulse el botón respectivo en la pantalla de la Tablet.

3.5.3.2.1 Botón POWER

En su código se ha empleado la función *setOnClickListener*, que representa al evento de pulsar el botón en la pantalla, para luego enviar un mensaje (sendMessage) hacia la variable VEL, cuyo valor es de 18, y finalmente la terminación, como se observa:

```
bPower.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        sendMessage("VEL:18\n");
    }
})
```

Este dato se recepta por el bloque de recepción correspondiente, pasa por el bloque Convert y se escribe en el espacio de memoria denominado VEL, para ser empleado en el programa.

3.5.3.2.2 Botón RESET

Su código fuente está compuesto a la vez por la función *setOnClickListener* correspondiente al evento de oprimir el botón y a la vez, proporciona la acción "*setText*", donde se escribe un espacio vacío (" ") en cada cuadro de edición de texto. Con esto, se pretende dar la acción de borrado a los tres espacios cada vez que se presione el botón RESET. A continuación, se indica la línea de código.

```
bReset.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            nX.setText("");
            nY.setText("");
            nZ.setText("");
        }
})
```

3.5.3.2.3 Botón HOME

Se mantiene la misma función del evento al oprimir el botón, pero se envía el valor entero 1 al bloque de recepción encargado de enviar el dato, se convierte y pasa a escribirlo en el espacio de memoria HOME del programa. Su código es:

```
bHome.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                sendMessage("HOME:1\n");
            }
        })
```

3.5.3.2.4 Selector MANU/AUTO

En la función del selector MANU/AUTO se ha empleado el control del tipo *ToggleButton*, correspondiente al denominado *BotonLed*, instanciado al inicio de la programación y que se observa en el código fuente.

```
        BotonLed.setOnClickListener(new View.OnClickListener() {
            public void onClick(View vv) {

                if(BotonLed.isChecked())     {
                    if(D) Log.e("BotonLed", "Encendiendo..");
                    sendMessage("MMANU:1\n");
findViewById(R.id.B_MMANU).setBackgroundResource(R.drawable.manu2);
                }
                else {
                    if(D) Log.e("BotonLed", "Apagando..");
                    sendMessage("MMANU:0\n");
findViewById(R.id.B_MMANU).setBackgroundResource(R.drawable.auto2);
                }
            }}}
```

El estado del *BotonLed*, se verifica y al ser pulsado, envía el valor 1 a la variable MMANU. Tal estado se mantiene hasta que se vuelva a presionar el botón y para esto, se envía el valor 0 a la variable MMANU; en ambos casos, se envía el dato al bloque de recepción para que sea convertido y se escriba en su correspondiente espacio de memoria. Con estas dos acciones, el usuario puede manipular y elegir el modo de posicionamiento manual/automático para el puente grúa. Adicionalmente para cada estado se ha definido la función ***setBackgroundResource*** que permite cambiar la imagen del botón, y que para este caso, tal imagen se sobrepone. Esto, con el fin de dar la impresión de tener un selector en la pantalla, en lugar de un botón para cada modo, tal funcionamiento se observa en la figura 3.75.

Figura 3.75 – Función del Selector Manual/Automático

3.5.3.2.5 Botón APERTURA

La función ***SetonTouchListener***, permite realizar una acción cuando se presiona el botón y cambiar a otra acción cuando se lo deja de presionar. Esto, a fin de mantener la lógica de control de la pinza, que consiste en mantener presionado el botón durante un cierto tiempo para poder abrirla. El código fuente está dado por:

```
bApertura.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("NANG:1\n");
                    sendMessage("ANG:180\n");
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("NANG:0\n");
                }
                return false;
            }
        })
```

Cuando se presiona el selector se activa ACTION DOWN, produciendo las siguientes acciones: enviar 1 a la variable NANG y a la vez el valor 180 a la variable ANG. Con esto, el proceso de apertura se realiza, mientras que, cuando se ha dejado de presionar se envía 0 a la variable NANG cuya función es indicar que se ha dejado de presionar el botón de apertura.

3.5.3.2.6 Botón CIERRE

La finalidad del botón CIERRE es enviar el valor 0 a la variable ANG, para permitir el cierre de la pinza. En su código, mantiene la función correspondiente al evento de oprimir el botón y así, enviar un dato al bloque de recepción que pertenece a este proceso. El código desarrollado se indica a continuación:

```
bCierre.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                sendMessage("ANG:0\n");
            }
        })
```

3.5.3.2.7 Botón Baja

Las acciones SUBE y BAJA de la pinza se desarrollan en el recorrido del eje Z, el sentido de la posición a la cual la pinza empieza a moverse se toma como referencia “+”, ya que se deben sumar los pasos. La variable para esta acción es MZ+. En el código fuente, la función *OnTouchListener()* permite escribir el valor 1

en la variable MZ+, cuando se presione el botón, a fin de que los pasos aumenten y la pinza comience a desplazarse hacia abajo. Al dejar de presionar, se envía el valor 0 a la variable indicando en la programación, que no se quiere seguir bajando. A continuación se indica el código fuente empleado para esta acción:

```
bBaja.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("MZ+:1\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MZ+:1\n");
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("MZ+:0\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MZ+:0\n");
                }
                return false;
            }
        })
```

La función ***for double*** (es decir, for (double a=0; a<500; a++){ }), permite enviar nuevamente la señal a la variable después de un tiempo, con la finalidad de asegurarnos que el dato sea recibido por el microcontrolador.

3.5.3.2.8 Botón SUBE

Esta acción busca decrecer los pasos hasta llegar a 0 de modo que la pinza comience a ascender en el eje Z. Para ello, se envía el valor 1 a la señal MZ-, y al dejar de presionar, se envía el valor 0 a la entrada de MZ- para indicar que el usuario ha dejado de pulsar el botón desde la interfaz de control. Del mismo modo, se vuelve a enviar la función ***for double*** por motivos de seguridad en la recepción de datos. El código fuente es el siguiente:

```
        bSube.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("MZ-:1\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MZ-:1\n");
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("MZ-:0\n");
                    for (double a=0; a<500; a++){
                    }
```

```
                sendMessage("MZ-:0\n");
            }
            return false;
        }
    })
```

3.5.3.2.9 Botones ADELANTE y ATRAS (Eje Y- y Eje Y+)

En la acción ADELANTE, se ha definido la variable MY+, puesto que el número de pasos debe crecer; mientras que para el caso de la acción ATRÁS, se ha definido la variable MY-, debido a la acción de decrecer los pasos. El código fuente correspondiente a las dos acciones, son semejantes a los empleados para las funciones SUBIR y BAJAR de la pinza. A continuación, se definen los códigos respectivos:

- ADELANTE

```
bAdelante.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent event) {
            if(event.getAction() == MotionEvent.ACTION_DOWN) {
                sendMessage("MY+:1\n");
                for (double a=0; a<500; a++){
                }
                sendMessage("MY+:1\n");
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                sendMessage("MY+:0\n");
                for (double a=0; a<500; a++){
                }
                sendMessage("MY+:0\n");
            }
            return false;
        }
    })
```

- ATRÁS

```
bAtras.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent event) {
            if(event.getAction() == MotionEvent.ACTION_DOWN) {
                sendMessage("MY-:1\n");
                for (double a=0; a<500; a++){
                }
                sendMessage("MY-:1\n");
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                sendMessage("MY-:0\n");
                for (double a=0; a<500; a++){
                }
                sendMessage("MY-:0\n");
            }
            return false;
        }
    })
```

3.5.3.2.10 Botones IZQUIERDA/DERECHA (Eje X+ y Eje X-)

Para la acción IZQUIERDA se ha empleado la variable MX+ referente a la acción de crecer los pasos, mientras que, para la acción DERECHA, se ha empleado la variable MX-, por su acción de decrecer los pasos en el recorrido. Cada acción cumple el mismo código definido para el caso ADELANTE/ ATRÁS, y sólo se han modificado las variables relacionadas a estas acciones. A continuación, se indica el código para cada caso:

- IZQUIERDA

```
bIzquierda.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("MX+:1\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MX+:1\n");
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("MX+:0\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MX+:0\n");
                }
                return false;
            }
        })
```

- DERECHA

```
bDerecha.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("MX-:1\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MX-:1\n");
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("MX-:0\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("MX-:0\n");
                }
                return false;
            }
           })
```

3.5.3.2.11 Botón ENTER

Actúa en el modo de posicionamiento automático, enviando los datos ingresados en los cuadros de edición de texto (NX, NY y NZ) de la interfaz gráfica. El código fuente para este propósito es el siguiente.

```
bEnter.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendMessage("POSX:"+nX.getText()+"\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("POSY:"+nY.getText()+"\n");
                    for (double a=0; a<500; a++){
                    }
                    sendMessage("POSZ:"+nZ.getText()+"\n");
                    for (double a=0; a<500; a++){
                    }
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    sendMessage("INICIAR:1\n");
                }
                return false;
            }
        })
```

La función ***Get.Text*** usada en el código del programa corresponde a la acción de tomar el texto ingresado por el usuario; y como se observa, empleamos las variables POSX, POSY y POSZ para la encriptación del dato hacia el microcontrolador, el número ingresado (nX, nY, nZ) y su respectiva terminación. Al pulsar el botón ENTER, se envían cada una de las posiciones, pero al dejar de pulsar el botón, se habilita la acción de la variable INICIAR, que se ha empleado para que se envíe el valor 1, es decir, se habilite el proceso de inicio del modo automático. Con esto, se asegura de ingresar primero los valores en las 3 o sólo 2 coordenadas, para iniciar el desplazamiento del puente, hacia dicha coordenada al iniciarse el proceso de funcionamiento. Caso contrario, la acción para el eje X ya empezaría a realizarse.

3.5.3.2.12 Designación del nombre del módulo Bluetooth

Para poder realizar la conexión inalámbrica entre dispositivo (Tablet) y Microcontrolador (módulo Bluetooth), es necesario establecer una comunicación serial o punto a punto, para definir la dirección de un dispositivo en específico al cual, la Tablet va a enviar los datos y mandos. Para ello, en el desarrollo del código fuente del programa es necesario definir la dirección MAC del dispositivo

con el cual se va a establecer dicha conexión. Una dirección MAC, es una codificación única en el mundo que posee cualquier dispositivo móvil que disponga Bluetooth y sirve para identificarlo al realizar cualquier tipo de aplicación. Para identificar dicha dirección se empleó la aplicación AMARINO, obteniéndose así la dirección MAC: “98:D3:31:90:09:AC” correspondiente al módulo HC-05. En el código fuente se ha ingresado tal dirección, a fin de establecer una conexión automática con el módulo Bluetooth, para la transmisión y recepción de datos. A continuación, se indica el código fuente desarrollado para este propósito.

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.Conexion:
        if(D) Log.e("conexion", "conectandonos");
String address = "98:D3:31:90:09:AC";//Dirección Mac del hc-05
 BluetoothDevice device = AdaptadorBT.getRemoteDevice(address);
                Servicio_BT.connect(device);
```

CAPÍTULO 4

PRUEBAS Y RESULTADOS

4.1 INTRODUCCIÓN

Al finalizar la implementación, construcción y montaje de todas las partes que constituyen el prototipo de puente grúa, se realizaron pruebas de funcionamiento tanto eléctricas como mecánicas, a fin de comprobar que se cumpla a cabalidad la lógica de control con los comandos enviados desde la interfaz gráfica de la Tablet hacia el prototipo, obteniéndose así los siguientes resultados.

4.2 PRUEBAS PRÁCTICAS DE FUNCIONAMIENTO

Con el fin de verificar los resultados esperados en la secuencia de funcionamiento, tanto del puente como del polipasto, y valiéndonos de la interfaz de control inalámbrica desarrollada en la aplicación de la Tablet se comprobó que ambos se desplazaban de manera correcta e instantánea al oprimir el botón correspondiente a los diferentes movimientos posibles de cada eje, esto es: Adelante, Atrás, Izquierda, Derecha / Sube-Baja.

Por otro lado, como se ha explicado anteriormente la secuencia de pasos empleada en los motores PAP producen que el motor gire en sentido horario o anti horario de acuerdo al comando enviado desde la Tablet, y se verificó que ésta sea la secuencia correcta tomando como referencia el espacio de seguridad HOME, lugar que además se considera como punto de referencia o de partida para los tres ejes del puente grúa (posición inicial).

Adicionalmente, se verificó que la secuencia de la lógica de control cumple con todos los parámetros y condiciones planteadas en el Capítulo 3.

Al realizar la prueba práctica, se pudo observar que el puente se desplaza con cierta distancia de desfase y mediante pruebas pertinentes se comprobó que el

problema había sido causado por el mal acoplamiento de los piñones en cada eje del motor. La solución a este problema fue simple y se procedió a fijarlos, empleando silicona caliente para tal fin.

Se presentó además un inconveniente en el desplazamiento vertical de la pinza de sujeción, puesto que ésta al no disponer de un sistema de elevación de carga que proporcione el ascenso y descenso de la pinza de forma recta y lineal, al llegar al fin de su recorrido tiende a balancearse o variar su posición. Se intentó reducir este problema, sujetando los cables suspendidos desde la parte superior del carrete hacia la pinza con un par de amarras ubicados a una distancia de 20 cms desde la parte superior del cable, consiguiéndose reducir notablemente el problema.

4.3 PRUEBAS AL SOFTWARE DE PROGRAMACIÓN

Existen varios programas que permiten desarrollar aplicaciones para dispositivos móviles, entre los más importantes se destacan: App Inventor y Android Studio.

Para el desarrollo de la aplicación que controle al prototipo, se optó por utilizar el Software de programación Android Studio. Pese a su complejidad en la programación, presenta mayores ventajas que el primero y de hecho, las primeras pruebas del proyecto, fueron realizadas con el Software de programación App Inventor presentándose falencias como: tiempo de respuesta muy lento debido a la sobrecarga de imágenes en el diseño, pocas herramientas de programación y de desarrollo de aplicaciones. Debido a estos problemas, se optó por utilizar el Software de programación Android Studio.

Para comprobar los diagramas de bloques que se emplean en cada una de las operaciones, se hizo necesaria la conexión desde el puerto mini USB de la tarjeta STM32F4DISCOVERY hacia la computadora, mediante un cable USB tipo A a mini-B; con esto, se pudieron realizar ciertas modificaciones en el programa.

El enlace punto-punto, entre la Tablet y el módulo Bluetooth HC-05, se establece de manera automática, esto implica que la Tablet no buscará ni accederá a dispositivos ajenos al módulo. El alcance máximo desde el cual, la Tablet puede mantener comunicación con el módulo Bluetooth, es de aproximadamente 10

metros; esto, sin considerar obstáculos a su alrededor. Si se presentan obstáculos, éstos interfieren en la comunicación y por ende, la distancia de alcance tiende a reducirse. Es de suponerse, que el usuario va a controlar el prototipo a una distancia que le permita visualizar los movimientos que realiza; por ello, se supone que el usuario no saldría del rango establecido como máximo. En forma práctica, y si el presente proyecto, fuese aplicado a un puente grúa real, la distancia máxima de 10 metros que se cubre con el módulo Bluetooth HC-05 no sería suficiente y se optaría por cambiarlo, por otro módulo Bluetooth de mayor alcance, como por ejemplo: un módulo Zigbee, que proporciona una distancia de 10 a 75 metros de alcance.

En cuanto al sistema de sujeción de objetos que lo componen, la pinza, así como el servomotor acoplado a ésta, y el sensor infrarrojo ubicado entre los dedos de la pinza, se realizó una serie de pruebas para comprobar su funcionamiento, destacándose las pruebas realizadas al sensor infrarrojo en cuanto al emisor se refiere. Se utilizó una cámara digital, para constatar la emisión de luz infrarroja que emite éste dispositivo, ya que se trata de un tipo de radiación electromagnética invisible al ojo humano pues su longitud de onda es mayor a la del espectro de luz visible.

La capacidad de carga para el polipasto, está limitada por su propia característica constitutiva y por las características mecánicas de la pinza.

El avance obtenido en modo automático al ingresar una coordenada de posición en la Tablet, no fue del todo precisa, por lo que se hizo necesario calibrar el avance en la programación.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- El objetivo principal por el cual se decidió realizar este prototipo se cumplió y a la vez, se desarrolló el control mediante bases sólidas de programación valiéndose de la tecnología actual. Esto implica ser un referente para futuros cambios y adaptaciones a la tecnología a través del tiempo, esto debido a que los avances tecnológicos están en constante desarrollo.
- Los dispositivos móviles como celulares y Tablets, vienen configurados de fábrica como maestros. Para poder establecer una comunicación con tales dispositivos es necesario configurar el módulo Bluetooth como esclavo.
- El módulo Bluetooth HC-05 viene configurado por defecto como esclavo, aunque puede ser configurado a modo maestro accediendo a su firmware.
- El presente proyecto, puede ser aplicable a circunstancias reales de mayor escala como lo son los puentes grúa empleados en la industria, para lo cual se debe emplear otro tipo de componentes de mayor potencia y capacidad.
- El prototipo puede ser controlado únicamente por un dispositivo móvil a la vez, esto por motivos de seguridad y debido a que la comunicación establecida es del tipo punto-punto. Para ello, se debe primero haber finalizado la conexión con un dispositivo, para luego establecer el enlace con un nuevo dispositivo.
- Entre las características de la Tablet se requiere que el dispositivo móvil sea de 7” para que la interfaz gráfica no se distorsione. La interfaz gráfica, ha sido desarrollada específicamente para esta medida de pantalla.

- El Software de programación Android permite desarrollar e implementar de forma práctica y sencilla, mediante una interfaz amigable para el usuario, cualquier tipo de aplicación compatible con cualquier dispositivo móvil y que puede ser empleada en campos como la Informática, Electrónica y las Telecomunicaciones, empleando para ello un moderno y novedoso lenguaje de programación.
- El microcontrolador STM32F407 utilizado, permite cumplir los objetivos planteados, brindando mayores herramientas y ventajas en cuanto a programación y depuración del programa, para un sin número de aplicaciones hoy en día.
- La herramienta SIMULINK, de MATLAB, permite la implementación de sistemas y procesos especiales en un entorno didáctico y asequible al usuario, facilitando la programación de tales procesos mediante bloques.
- La utilización de motores PAP con el fin de obtener movimientos por pulsos para el desplazamiento del puente y del carro, conllevan a la elección de diferentes sistemas de transmisión que son empleados para variar el torque y la velocidad del motor, o lo que es semejante a un reductor de velocidad.
- El prototipo ha sido diseñado para levantar piezas generalmente cúbicas, y que no excedan de los 5 cm de ancho, máxima distancia que cubre la apertura de la pinza.
- El método utilizado para el posicionamiento automático, consiste el conteo del número de pasos que debe recorrer el motor a lo largo de cada eje, siendo este un sistema sin retroalimentación.

5.2 RECOMENDACIONES

- Se puede conectar la tarjeta STM32F4 DISCOVERY de dos formas, la primera utilizando un cable USB de tipo A a mini-B, y la segunda, utilizando un cable USB de tipo A a micro-B. Se debe tomar en cuenta que el cable utilizado para el primer caso, nos permite compilar el programa en el microcontrolador y a la vez, permite alimentar con un voltaje de 5V a toda

la tarjeta. Para el segundo caso, solamente es posible alimentar la tarjeta a través de su respectivo puerto micro USB, pues estructuralmente la conexión de este puerto no tiene relación con el depurador de la tarjeta.

- En el desarrollo de la interfaz gráfica, al momento de seleccionar la plataforma de desarrollo es importante consultar si es necesario una característica especial, que solo esté disponible en una versión. Puesto que en los dispositivos con versiones inferiores a la seleccionada no se podrá instalar la aplicación. Por lo tanto, se recomienda seleccionar la menor versión posible que la aplicación pueda soportar.
- Se recomienda utilizar el control inalámbrico (Tablet) hasta con un 17% de la carga de la batería, luego de lo cual, se deberá proceder a cargar al dispositivo. En este estado, no se debe utilizar al dispositivo, pues se presentan interferencias en la comunicación.
- Se recomienda antes de emplear el prototipo, ubicarlo en una superficie plana, para evitar que se produzca vibración excesiva en el desplazamiento del puente a lo largo de su recorrido.
- Como medida de seguridad se debe limitar el descenso del sistema de izaje mediante un fin de carrera y no mediante programación del software, como se lo hizo en este proyecto.
- Como base para la experimentación de nuevos sistemas de control en este prototipo, se puede considerar cambiar la señal acústica para que alerte únicamente en el descenso o ascenso de la pinza cuando esté con carga y no en todo el recorrido, puesto que la sirena empleada (buzzer) produce ruido excesivo. Puede considerarse reemplazar el sonido por una señal acústica intermitente.
- Con la ayuda de técnicas avanzadas de programación en Android y a un trabajo multidisciplinario que involucre varias áreas de estudio se puede desarrollar una interfaz gráfica más avanzada, con un mejor diseño y que incluya funciones mucho más complejas, como: poder adaptar la aplicación a cualquier dispositivo sin que el tamaño de pantalla sea un limitante, e incluso, poder manipular los controles mediante comandos de voz.

- Del mismo modo, se puede acceder a varias aplicaciones como Google Play, donde se puede subir la aplicación y disponer de ella en todo momento.
- La pinza utilizada en el proyecto es usada únicamente con fines didácticos, y sirve para representar una mejora que se puede dar en la sujeción de objetos. En la vida real, éste elemento puede ser reemplazado por otro que brinde mayores prestaciones en cuanto a seguridad y funcionalidad.

BIBLIOGRAFÍA

[1] E. PALACIOS, Microcontrolador PIC16f84 Desarrollo de Proyectos, España: Alfaomega, 2008.

[2] J. VEGA, Microcontroladores motorola-freescale, Colombia: Alfaomega, 2007.

[3] «Dispositivos Lógicos Programables,» http://perso.wanadoo.es/pictob/microprg.htm, [En línea]. Available: http://perso.wanadoo.es/pictob/microprg.htm#arquitectura_de_un_sistema_basado_en_cpu. [Último acceso: 09 junio 2015].

[4] S. M. Canel, «Microcontroladores ARM,» 19 noviembre 2007. [En línea]. Available: http://www.electron.frba.utn.edu.ar/upload/Materias/95-0429/archivos/Cap10_2009_ARM7_apunte.pdf. [Último acceso: 17 junio 2015].

[5] «EcuRed,» [En línea]. Available: http://www.ecured.cu/index.php/ARM. [Último acceso: 19 junio 2015].

[6] Firtec, Introduccion a la Arquitectura ARM: ARM Cortex M4, Santa Fé: Firtec, 2014.

[7] «Robotica Ludica,» septiembre 2012. [En línea]. Available: http://roboticaludica.com/?p=1008. [Último acceso: 19 junio 2015].

[8] Webelectronica. [En línea]. Available: http://serverpruebas.com.ar/news14/nota01.htm. [Último acceso: 09 junio 2015].

[9] «ROBOTS,» http://robots-argentina.com.ar/robots.htm, [En línea]. Available: http://robots-argentina.com.ar/MotorServo_basico.htm. [Último acceso: 09 junio 2015].

[10] «Laboratorio de Electronica,» http://www.info-ab.uclm.es/labelec/solar, [En línea]. Available: http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm. [Último acceso: 09 junio 2015].

[11] «Sensores, características,» [En línea]. Available: http://artemisa.unicauca.edu.co/~gavasquez/res/Sensores.pdf. [Último acceso: 09 junio 2015].

[12] «Clasificación de los Sensores,» http://proton.ucting.udg.mx/~redblade/Paginas/inicio/inicio.html, [En línea]. Available: http://proton.ucting.udg.mx/~redblade/Paginas/Robotica/Tareas/sensores/Sensores.html. [Último acceso: 09 junio 2015].

[13] Recursostic, «Sensores de contacto,» [En línea]. Available: http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena11/4quincena11_contenidos_3b.htm. [Último acceso: 09 junio 2015].

[14] T. FLOYD, Dispositivos electrónicos, 8va ed., México: Pearson, 2008.

[15] V. GONZÁLEZ, «Fundamentos de Robótica,» http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/index.htm, marzo 2002. [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/terminal.htm. [Último acceso: 09 junio 2015].

[16] L. GONZÁLEZ, «Mecanismos de trasmisión de movimiento,» [En línea]. Available: http://almez.pntic.mec.es/jgonza86/centro.htm. [Último acceso: 09 junio 2015].

[17] CEJAROSU, «MecanESO,» http://concurso.cnice.mec.es/cnice2006/material107/index.htm, 2005. [En línea]. Available: http://concurso.cnice.mec.es/cnice2006/material107/operadores/ope_cremall

era.htm. [Último acceso: 09 junio 2015].

[18] S. CAMBLOR, «IES PEDRO DE TOLOSA,» http://iesptolosa.net/, [En línea]. Available: http://iesptolosa.net/ies/dptos/dpto_tecnologia/1eso/Mecanismos.pdf. [Último acceso: 09 junio 2015].

[19] D. R. PLANAS, «NTP 736: Grúas tipo puente (I): generalidades,» [En línea]. Available: http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/701a750/ntp_736.pdf. [Último acceso: 09 junio 2015].

[20] E. B. JIMÉNEZ, «Guía de Productos Industriales de Aplicacion en Ingeniería Mecánica,» http://dim.usal.es/areaim/guia%20P.%20I/index.htm, [En línea]. Available: http://dim.usal.es/areaim/guia%20P.%20I/PAGINA%20SEGUNDA.htm. [Último acceso: 09 junio 2015].

[21] «Slideshare,» 23 Abril 2014. [En línea]. Available: http://es.slideshare.net/ARTRmichael12/grua-puente. [Último acceso: 09 junio 2015].

[22] E. Consultora, «Estrucplan on line,» 2001-2002. [En línea]. Available: http://www.estrucplan.com.ar/Producciones/entrega.asp?IdEntrega=113. [Último acceso: 09 junio 2015].

[23] M. E. ARCIA, «About en español,» [En línea]. Available: http://tabletas.about.com/od/Conoce-las-tabletas/a/Tablet-Que-Es-Y-Por-Que-Es-Tan-Popular.htm. [Último acceso: 09 junio 2015].

[24] «MikroElektronika,» [En línea]. Available: http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/. [Último acceso: 09 junio 2015].

[25] «MultiON,» 2013. [En línea]. Available:

http://www.multion.com.mx/micrositios/matlab/inicio.html#descripci%C3%B3n. [Último acceso: 09 junio 2015].

[26] L. Aimaging Co., «Waijung Blockset,» 2015. [En línea]. Available: http://waijung.aimagin.com/index.htm?minimum_requirements.htm. [Último acceso: 09 junio 2015].

[27] T. J. GIRONÉS, El gran Libro de Android, 3ra ed., México: Alfaomega, 2013.

[28] «Developers Android,» http://developer.android.com/index.html, [En línea]. Available: http://developer.android.com/guide/components/fundamentals.html. [Último acceso: 09 junio 2015].

[29] STMicroelectronics, «UM1472 Manual User STM32F407/417 lines,» http://www.st.com/web/en/home.html, 2014. [En línea]. Available: http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00039084.pdf. [Último acceso: 09 junio 2015].

[30] «Guia de Iniciación STM32F4 Discovery,» [En línea]. Available: http://www.disca.upv.es/aperles/arm_cortex_m3/curset/guia_iniciacion_STM32F4_discovery.pdf. [Último acceso: 09 junio 2015].

[31] C. CELLERI, Control Indistrial, Quito: Escuela Politécnica Nacional, 1984.

[32] L. VIÑAS, Circuitos y dispositivos electrónicos, 6ta ed., Barcelona: Alfaomega, 2001.

[33] J. SINGH, Dispositivos Semiconductores, México: McGRAW-HILL, 1997.

[34] STMicroelectronics, "L298 DUAL FULL- BRIDGE DRIVER," 2000. [Online]. Available: http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf. [Accessed 09 junio 2015].

[35] «ELECTRONILAB,» http://electronilab.co/, [En línea]. Available: http://electronilab.co/wp-content/uploads/2014/02/Comparacion-HC05-y-HC06-Electronilab.pdf. [Último acceso: 09 junio 2015].

[36] A. CRUZ, «Tutorial de Uso Driver L298N,» Electronilab, 2014. [En línea]. Available: http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/. [Último acceso: 2015 junio 09].

[37] «NEOTEO,» 20 noviembre 2011. [En línea]. Available: http://www.neoteo.com/modulo-bluetooth-hc-06-android/. [Último acceso: 09 junio 2015].

[38] Electronilab, "HC Serial Products," [Online]. Available: http://electronilab.co/wp-content/uploads/2014/02/Comparacion-HC05-y-HC06-Electronilab.pdf. [Accessed 09 junio 2015].

[39] A. DURÁN, «HETPRO,» http://hetpro-store.com/, [En línea]. Available: http://hetpro-store.com/TUTORIALES/bluetooth_hc-06_app_arduino/. [Último acceso: 16 junio 2015].

[40] R. Asesores, «PROMETEC,» 2014. [En línea]. Available: http://www.prometec.net/bt-hc05/. [Último acceso: 09 junio 2015].

[41] LINOTUX, «HC03/05 Embedded Bluetooth Serial Communication Module,» abril 2011. [En línea]. Available: http://www.linotux.ch/arduino/HC-0305_serial_module_AT_commamd_set_201104_revised.pdf. [Último acceso: 09 junio 2015].

[42] «MathWorks,» 1994-2015. [En línea]. Available: http://es.mathworks.com/products/stateflow/features.html?s_tid=gn_loc_drop. [Último acceso: 09 junio 2015].

[43] «Simulink,» [En línea]. Available: http://ocw.upc.edu/download.php?file=15015404/tema_5_simulink-5156.pdf.

[Último acceso: 09 junio 2015].

[44] Galeon, «Simulink,» [En línea]. Available: http://www.galeon.com/mcoronado/MODELAMIENTO/10SIMULINK.pdf. [Último acceso: 09 junio 2015].

[45] «Sgoliver.net blog,» 2015. [En línea]. Available: http://www.sgoliver.net/blog/interfaz-de-usuario-en-android-controles-basicos-i/. [Último acceso: 09 junio 2015].

[46] TELMEXhub. [En línea]. Available: https://github.com/TELMEXhub/Repositorio_Proyecto1.

[47] «SENSORES,» 11 diciembre 2010. [En línea]. Available: http://thelastlabproject.blogspot.com/. [Último acceso: 09 junio 2015].