Universidad Autónoma de Baja California
Facultad de Ingeniería

Programa Educativo de Ingeniero en Mecatrónica

Control de un Mecanismo Rotacional de 1GDL
Utilizando Actuadores de Aleaciones con Memoria
de Forma y la Plataforma Raspberry PI

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el título de INGENIERO EN MECATRÓNICA, presenta:

Carlos Alan Garcia Campos
Asesor: Dr. David Isaías Rosas Almeida

Mexicali, Baja California, México, Abril de 2021.

Agradecimientos

... Esta dedicatoria es para mi madre Ciran Campos Aispuro, por su fortaleza para educarme con empeño y no dejarme solo en los momentos más difíciles, trazando mi camino a seguir hasta llegar al final de una carrera profesional.

A mis tías, Esmeralda y Marian, las cuales ayudaron a mis hermanos y a mí, a poder continuar estudiando desde temprana edad y a complementar nuestra formación.

Para mis amigos y compañeros de clase, cuyas metas se entrelazaron y nos unimos para lograr objetivos mayores.

Una mención especial al profesor Dr. David Rosas Almeida y el Dr. Jesus Rigoberto Herrera Garcia, por su instrucción durante el trabajo realizado en esta tesis y su ayuda para establecer mis metas y el camino a tomar a futuro.

A mi alma mater, la Universidad Autónoma de Baja California, por todo el conocimiento que se me ofreció y permitió el desarrollo de la tesis presentada.

Resumen

RESUMEN de la Tesis de CARLOS ALAN GARCIA CAMPOS, presentada como requisito para la obtención del título de INGENIERO EN MECATRÓNICA. Mexicali, Baja California, México, Agosto de 2021.

CONTROL DE UN MECANISMO ROTACIONAL DE 1GDL UTILIZANDO ACTUADORES DE ALEACIONES CON MEMORIA DE FORMA Y LA PLATAFORMA RASPBERRY PI

Resumen aprobado por:

Dr. David Isaías Rosas Almeida
Director de Tesis

En esta tesis se presenta la aplicación de un algoritmo PI en el control angular de un mecanismo rotacional a través de un protocolo de comunicación Bluetooth. Con el uso de una Raspberry Pi se extrajo del sensor TSS-MBT las coordenadas de su posición angular (dadas en ángulos de Euler) para ser posteriormente procesadas e introducidas a un algoritmo de control. Los actuadores utilizados son resortes con recubrimiento de NiTiNOL, los comúnmente llamados materiales de memoria de forma; se utilizó una técnica de control en función de una salida PWM, acondicionando la señal en una etapa de potencia. Este trabajo tiene como objetivo, el establecer una comunicación inalámbrica estable entre un sensor TSS-MBT y una Raspberry Pi, para la aplicación de tareas de bajo nivel como algoritmos de control.

Palabras clave:
Raspberry Pi, Bluetooth, Nitinol.

Abstract

ABSTRACT of the Thesis presented by CARLOS ALAN GARCIA CAMPOS, in order to obtain the MECHATRONIC ENGINEER degree. Mexicali, Baja California, México, August 2021.

CONTROL DE UN MECANISMO ROTACIONAL DE 1GDL UTILIZANDO ACTUADORES DE ALEACIONES CON MEMORIA DE FORMA Y LA PLATAFORMA RASPBERRY PI

Approved by:

Dr. David Isaías Rosas Almeida
Thesis Advisor

In this thesis it is presented the application of a PI algorithm in the angular control of a rotational Mechanism through a Bluetooth communication protocol. With the use of a Raspberry Pi, the coordinates of its angular position were taken from the TSS-MBT sensor (given on Euler Angles) to process and introduce it in the control algorithm. The actuators used in the system were NiTiNOL coated springs, usually called as Shape Memory alloys, whose control made through a PWM - Technique, conditionate the PWM signal in a power stage. This work has as an objective, to set up an estable wireless communication between the TSS-MBT sensor and the Raspberry Pi, to fulfill low level tasks as control algorithm.

Keywords:
Raspberry Pi, Bluetooth, Nitinol.

Contenido

INTRODUCCIÓN **1**

1 ANTECEDENTES **3**

1.1 Lenguaje de programación interpretado Python 4

1.1.1 El interpretador de Python 4

1.1.2 Diferencias entre Python Series 2 y Python Series 3 6

1.1.3 Objetos en Python 6

1.1.3.1 Operadores Lógicos 8

1.1.3.2 Operadores de Igualdad 9

1.1.3.3 Operadores Secuenciales 9

1.1.4 Flujo de control de información 9

1.2 Protocolos de Comunicación 11

1.2.1 Codificación por formato ASCII 12

1.2.2 Comunicación Bluetooth 14

1.3 Cinemática Rotacional 16

1.3.1 Matrices de Rotación 17

1.3.2 Ángulos de Euler (Pitch, Yaw, Roll) 22

1.4 Desarrollo de algoritmos de control para una señal discreta 25

2 PUESTA EN OPERACIÓN DE LA PLATAFORMA RASPBERRY PI 3 B+ 28

2.1 Características del hardware 29

2.2 Sistema operativo 30

2.3 Configuraciones de software 32

2.3.1 Python 3 IDLE 36

2.3.1.1 Librerías de Python 37

3 CONEXIÓN, CONFIGURACIÓN Y LECTURA DEL SENSOR 3 SPACE MBT 40

3.1 Arquitectura 41

3.2 Transferencia de datos 43

3.3 Pruebas de transferencia de datos entre el sensor y la Raspberry 44

4 CONFIGURACIÓN Y ALGORITMOS PARA EL USO DE LAS TERMINALES DEL GPIO PARA EL CONTROL DE ACTUADORES BASADOS EN PWM 56

4.1 PWM en Raspberry Pi 57

4.2 PWM para el control de una Etapa de Potencia 59

4.3 Código para operación de dos canales de PWM 61

5 IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL EN RASPBERRY PI 3 B+ 64

5.1 Descripción de la planta 64

5.2 Código de control en Python 66

5.3 Respuesta de la planta a la aplicación del algoritmo de control 70

5.4 Desempeño de los controladores P y PI 72

6 CONCLUSIONES Y TRABAJO FUTURO 77

REFERENCIAS 79

ANEXO A. CÓDIGO DEL PROGRAMA EN PYTHON QUE REALIZA EL CONTROL EN LAZO CERRADO. 81

ANEXO B. TABLA ACTUALIZADA DEL CÓDIGO PARA CARACTERES ASCII 93

Lista de Figuras

Figura 1 a) Representación de la conversión entre un lenguaje interpretado (Python) a su respectivo codigo binario. b) Representación sobre la secuencia de conversión de lenguaje compilado a codigo binario. 5

Figura 2 Comparación entre las versiones Python 2 y Python 3. 6

Figura 3 Modelo estándar de comunicaciones simplificada [3]. 11

Figura 4 Arquitectura de un protocolo de comunicación a tres capas. 13

Figura 5 Rotación de un marco $F(pXYZ)$ a $G(pxyz)$ para la representación del movimiento de un punto fijo q. 18

Figura 6 Rotaciones básicas sobre los ejes x-y-z con punto fijo en el origen. 21

Figura 7 Notación de las transformaciones Pitch, Yaw, Roll, respecto a una aeronave. 23

Figura 8 Tarjeta SoC Raspberry Pi 3 Modelo B+ 29

Figura 9 Interface de Instalación de sistemas operativos NOOBS. 32

Figura 10 Interfaz gráfica predeterminada del Sistema Operativo. 33

Figura 11 Panel de configuraciones de la Raspberry Pi 3 B+. 35

Figura 12 Interfaz de programación Thonny Python IDE. 36

Figura 13 Editor del menú de aplicaciones. 37

Figura 14 Componentes y dimensiones de hardware externo [18]. 41

Figura 15 Diagrama de bloques del funcionamiento del sensor TSS-MBT. [18] 42

Figura 16 Formato de transferencia de datos del sensor TSS-MBT. [18] 43

Figura 17 Estructura de comandos para formato en paquetes ASCII.[18] 44

Figura 18 Ángulos de Euler de acuerdo a un cambio unidirecciónal en la posición. 54

Figura 19 Variación de velocidad del giroscopio respecto a los ángulos de Euler. 55

Figura 20 Distribución de las terminales del GPIO para una Raspberry Pi 3 B+ [17]. 57

Figura 21 Etapa de potencia para el control de actuadores de Nitinol. 60

Figura 22 Mecanismo rotacional de un grado de libertad en forma de T con actuadores de Nitinol. 65

Figura 23 Representación de la transferencia de datos en función de las etapas del sistema. 67

Figura 24 Comportamiento del algoritmo de Control Proporcional. 71

Figura 25 Comportamiento del algoritmo de Control Proporcional - Integral. 71

Figura 26 Posición angular y señal de control cuando el mecanismo se mueve a la derecha utilizando un control proporcional. Se muestran 5 experimentos. 73

Figura 27 Posición angular y señal de control cuando el mecanismo se mueve a la izquierda utilizando un control proporcional. 5 experimentos. 74

Figura 28 Posición angular y señal de control para el movimiento del actuador hacia la derecha utilizando un controlador PI. 5 experimentos. 75

Figura 29 Posición angular y señal de control para el movimiento del mecanismo hacia la izquierda utilizando un controlador PI. 5 experimentos. 76

Introducción

El lenguaje de programación Python es en la actualidad uno de los lenguajes más utilizados en el área de la ciencia de datos debido a su fácil acceso y potencia, logrando deshacerse de las complicaciones de la programación en bajo nivel. Al ser un lenguaje de plataforma abierta ofrece un amplio soporte por parte de la comunidad en línea, siendo su flexibilidad y adaptabilidad sus características principales.

Con la llegada de las tarjetas SoC (System on Chip) al mercado, se comenzó la creación de aplicaciones desarrolladas en Python para dichos sistemas, algunos inclusive tomaron el lenguaje de programación como su predeterminado. Una de las más populares en la actualidad es la tarjeta Raspberry Pi, cuya capacidad de procesamiento le permite crear diversas aplicaciones en áreas de la ingeniería, principalmente en el control de sistemas.

No obstante, los programas desarrollados en esta plataforma son variados de acuerdo a su aplicación, así como los instrumentos de medición que se utilizan. Tanto los protocolos de comunicación como la utilización de ciertos dispositivos esta delimitada por la información de las hojas de datos, algunas veces, siendo redundante y complicando el uso del dispositivo para que realice una operación menor.

En este trabajo se presenta una forma de comunicación inalámbrica con un sensor de Yost Labs 3-Space MBT a través de Python, donde, por medio del uso de una Raspberry Pi, se desea controlar la posición angular de una estructura en forma de T con el uso de resortes de aleación de NiTiNOL también conocidos como materiales de memoria de forma.

El diseño y la construcción del sistema del mecanismo fue desarrollado por mis compañeros Lizeth Fernanda Moreno Rodríguez, Jorge Parra López y Luis Fernando Zaid Diaz Mendoza, cuyo trabajo se observa en el diseño del mecanismo, así como, el desarrollo de

la etapa de potencia para el control de los actuadores por medio de un PWM. Mi participación dentro del desarrollo del proyecto se ve reflejado en la comunicación del sensor y los algoritmos PID para el controlador en la Raspberry Pi.

El siguiente documento esta dividido en 7 capítulos dando explicación a la metodología realizada para el desarrollo del trabajo. El primer capítulo (Antecedentes), comenta de manera general los conocimientos previos para comprender el transfondo y el sustento base de la investigación. El capítulo 2 (Puesta en operación de la plataforma Raspberry Pi 3 B+) da la introducción al uso de herramientas didácticas como la Raspberry Pi así como su Set-Up para el desarrollo del capítulo 3 (Conexión, configuración y lectura del sensor 3 Space MBT); que con el uso del formato de transmisión de datos por comas, se extraerá y procesará la información del sensor. El capítulo 4 (Configuración y algoritmos para el uso de las terminales del GPIO para el control de actuadores basados en PWM) explica el funcionamiento de la etapa de potencia y su forma de operarse a través de la Raspberry Pi. El capítulo 5 (Implementación de algoritmos de control en Raspberry Pi 3 B+) introduce los componentes de la planta, y a su vez desarrolla los algoritmos para el controlador PI junto con las pruebas de desempeño para ambas direcciones. Por último, el capítulo 7 (Conclusiones y trabajo futuro) comenta la relación entre los resultados obtenidos y las mejoras posibles para la optimización de hardware y software de la planta.

Capítulo 1

Antecedentes

En un mundo lleno de tecnologías, no es común que la gente se pregunte acerca de los inventos o descubrimientos que fueron necesarios para poder llegar a los dispositivos que hoy en día se usan cotidianamente. Bajo una comunidad consumista, se busca el beneficio individual respecto al conocimiento que nos trajo dichas tecnologías. Sin embargo, nada puede nacer sin inventos o descubrimientos previos.

Este proyecto consiste en la propuesta de metodologías eficaces para la implementación de sistemas de control, con el uso de la tarjeta Raspberry Pi a través de Python. Por lo que es necesario explicar conceptos basicos para poder entender el núcleo del trabajo propuesto. El trabajo a continuación dara explicaciones sobre el lenguaje de programación utilizado, los protocolos de lectura, marcos de referencia de acuerdo a su posición, controladores, etc.

En este capítulo se presenta un resumen de los principales conceptos y desarrollos tecnológicos en los cuales se basa el desarrollo de este proyecto de tesis. Se incluyen conceptos sobre un lenguaje interpretado y uno compilado. La estructura de un modelo estándar para comunicaciones, el formato internacional ASCII, así como la comunicación inalámbrica a corta distancia conocida como Bluetooth.

También se incorporan conceptos geométricos como las matrices de rotación para definir la orientación de un cuerpo rígido, el algoritmo de control Proporcional Integral Derivativo (PID) y la modulación por ancho de pulso PWM que se utilizará para accionar actuadores con aleaciones de Nitinol.

1.1 Lenguaje de programación interpretado Python

Python es un lenguaje de programación creado por Guido Von Rossum en 1991, con la intención de facilitar a los principiantes la comprensión de los principios de la programación; es un lenguaje interpretado, es decir, su código no necesita ser preprocesado mediante un compilador, por lo que no posee tiempo de compilación y en consecuencia se ejecuta al mismo tiempo que el código esta siendo escrito [1]. La portabilidad del lenguaje le provee facilidad de instalación para cualquier sistema operativo (Windows, Linux, Mac), diseñado para su implementación en aplicaciones dedicadas a dispositivos como la plataforma Raspberry Pi.

Python posee características que soportan diferentes estructuras de programación como la orientada a objetos, la programación estructurada, y la vectorial por lo que puede ser acoplado a otros lenguajes de programación.

Siendo un lenguaje de uso libre, su comunidad en línea provee soporte al lenguaje, facilitando el acceso a principiantes interesados en el área. Finalmente es un lenguaje simple de utilizar, lo que lo hace altamente intuitivo y fácil de leer, lo que genera tendencia entre la comunidad, siendo una de las opciones más elegidas entre los programadores principiantes [1].

1.1.1 El interpretador de Python

En la programación estructurada existen diversas formas de poder traducir el código del lenguaje programático al lenguaje máquina, las dos formas más utilizadas son las compiladas y las interpretadas. Ambos métodos convierten el código al lenguaje de procesador (binario) para que este sea ejecutado. La diferencia entre ambos radica en los procedimientos de su transformación. El interpretador traduce el código a binario sin la necesidad de ser compilado, es decir, se ejecuta el código al mismo tiempo que este es escrito, ver el caso a)

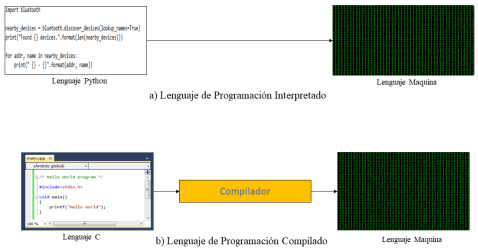

Figura 1. a) Representación de la conversión entre un lenguaje interpretado (Python) a su respectivo codigo binario. b) Representación sobre la secuencia de conversión de lenguaje compilado a codigo binario.

de la figura 1, por el otro lado, en los lenguajes compilados es necesario una previa compilación antes de poder dar instrucciones al procesador a través de estos lenguajes, lo que corresponde al caso b) en la figura 1.

Como se había comentado anteriormente, Python es un lenguaje interpretado que escanea y traduce los comandos como estos estén siendo escritos, el software que realiza estas operaciones se le conoce como interpretador de Python [2]. Este software recibe un comando de un código fuente con extensión ".py", el cual realiza una serie de operaciones con la finalidad de reportar el resultado de sus operaciones en una PVM (Python Virtual Machine).

Existen diferentes tipos de interpretadores para Python dependiendo del tipo de aplicación. Por ejemplo, el interpretador estándar de Python se le conoce como Cython [1], el cual adquiere el código en Python y lo transforma en su equivalente en código C; el código resultante se ejecutará en segundo plano en un entorno JIT (Just-In-Time) de Cython. También existen otros interpretadores como Jython (Java), IronPython (C#), y PyPy, desarrollado este último en su totalidad en un entorno de Python.

Figura 2. Comparación entre las versiones Python 2 y Python 3.

1.1.2 Diferencias entre Python Series 2 y Python Series 3

Dentro de las comunidades en línea que le dan soporte al lenguaje de programación en Python se puede encontrar una evidente separación entre las versiones de Python 2 respecto a los códigos realizados en Python 3, los usuarios se preguntaron entonces, ¿Por qué continua en circulación la versión 2, existiendo versiones mucho mas actualizadas como sería la serie 3 de Python? A la respuesta que se llegó fue un problema de compatibilidad.

Cuando el creador del lenguaje Python (Guido Van Rossum) decidió hacer cambios significativos al lenguaje, se encontró con el problema de que estos serían incompatible con una gran cantidad de código ya existente, por lo que se optó por la creación de una nueva versión de Python, después conocida como la serie 3 [1]. Para conservar los códigos antiguos aún compatibles permanece una única versión de la serie anterior, la versión 2.7. Se puede apreciar en la figura 2 algunas diferencias de sintaxis y ejecución del mismo código en versiones diferentes.

1.1.3 Objetos en Python

Python al ser un lenguaje con sintaxis orientada a objetos, las clases que se utilizan son la base para los tipos de datos que se manejan [2]. Las clases disponibles varían de acuerdo al lenguaje manejado. Por ejemplo, sea la clase *int* para valores enteros, la clase *float* encargada de los valores de punto flotante, y la *str* para la manipulación de caracteres, todas estas

pertenecientes a las clases de asignación de variables, siendo las basicas de cualquier lenguaje de programación.

Dentro del uso de Python, el comando mas importante que este posee es la asignación de enunciados, o mejor conocido por el carácter unicode (=), que se utiliza en la siguiente forma

$$bd\_addr = "XX:XX:XX:XX". \quad (1)$$

Este comando asigna a *bd_addr* como un identificador, y lo asocia con el objeto expresado del lado opuesto de la igualdad, lo que de acuerdo a (1) termina convirtiendo al identificador en una variable tipo *str*. A diferencia de otros lenguajes de programación, no es necesario especificar antes de usar el identificador la clase para la cual tomará valor.

Los identificadores pueden variar y ser distintivos de acuerdo a sus nombres, por ejemplo, *bd_addr* no es equivalente a *Bd_addr*. Python 3 admite una combinación de letras, números y caracteres pertenecientes al código unicode [2]. Aún con la gran variedad de identificadores, Python maneja 33 palabras reservadas que no permiten su uso como identificadores. Algunas de las palabras reservadas más comunes son: *false, as, continue, else, from, in, not, return, none, assert, def, except, global, is, or, try, true, breaj, del, finally, if, lambda, pass, while, and, class, elif, for, import, nonlocal, raise, with.*

Así como existen clases reservadas y no modificables se posee la opción de crear las clases que ayuden a reducir el nivel de complejidad del código. El proceso de creación de un nuevo apartado de una clase recibe el nombre de *Instanciación* [2]**.** La sintaxis utilizada para "*instanciar"* un objeto es invocar a la clase constructora, esto varía de acuerdo a la clase que se utiliza. Por ejemplo, si se tiene el enunciado *sock = BluetoothSocket (RFCOMM)*, en la variable *sock* se está guardando la característica de la clase *BluetoothSocket( )*. Dado que este enunciado requiere parámetros, es necesario especificarlos para que funcione correctamente.

Dentro de las clases de Python, pueden existir casos que estos tengan más de una función disponible, así que, en lugar de crear una clase completamente nueva, se queda como un *atributo* de la clase que se está utilizando. Estos atributos pueden ser invocados usando un punto operador ("."). En el ejemplo anterior, se instanció la variable *sock* en la clase *BluetoothSocket (RFCOMM)*. Entonces, si se quisiera usar uno de los atributos de esta clase se puede usar como *sock.connect((bd_addr,port))*. Evitando una sintaxis del tipo *BluetoothSocket (RFCOMM).connect((bd_addr,port))*. Ambas sintaxis son equivalentes, sin embargo, desde el punto de vista de un programador, la primera forma reduce la longitud del código, siendo mas amigable a la vista.

En cualquier lenguaje de programación es necesario poder expresar relaciones o equivalencias de los comandos que se escriben. Como se vio en los casos anteriores, el uso de un identificador esta limitado al valor que se le asigne. Por consiguiente, como fue en el caso de (1) se le asignó un valor para en caso de requerirse posteriormente, realice una operación o se compare contra otros identificadores. A estos símbolos o palabras especiales que asignan un valor o realizan alguna comparación se le conocen como *operadores* [2]. Hay una gran variedad de tipos de operadores disponibles en Python. A continuación, se dará ejemplos de los tipos de operadores y de acuerdo al operador que tipo de valores se pueden utilizar con ellos.

1.1.3.1 Operadores Lógicos

Python puede utilizar los siguientes operadores para realizar comparaciones entre valores booleanos.

not	Negación
and	Condicional z "
or	Condional .º "

1.1.3.2 Operadores de Igualdad

Si se desea expresar la relación de dos identificadores de acuerdo a su similitud o igualdad se pueden usar los siguientes operadores. Estos regresan valores de naturaleza ¨*True*¨ o ¨*False*¨.

is	Misma Identidad
is not	Identidad Diferente
==	Equivalente a
!=	Diferente a

1.1.3.3 Operadores Secuenciales

Para el manejo de caracteres, matrices y vectores dentro de Python, se permite realizar operaciones y comparaciones dado los operadores siguientes:

$s[j]$	Elemento en la posición j
s[start:stop:step]	Cuenta de $steps$ desde $start$ hasta $stop$
$s + t$	Concatenación de secuencias
val **in** s	Revisa si s se encuentra en val

1.1.4 Flujo de control de información

Como se vio en la sección anterior, el uso de operadores permite la comparación entre dos identificadores siempre que estos posean valores una naturaleza similar. Sí es necesario controlar o delimitar cierta información en Python, se puede realizar mediante el uso de estructuras de control. En el mundo de la informática las mas utilizadas y soportadas por Python son las condicionales y los ciclos de información.[2]

Las **estructuras condicionales**, especifican una forma de ejecutar una sección de código de acuerdo si se cumple (o no), una o varias expresiones booleanas.[2] Python permite el acceso a esta estructura a través de la palabra reservada (*if*), usualmente posee la estructura:

if condición_se_cumple:

realizar_código

Sí se desea finalizar un ciclo de acuerdo a un contador ¨b¨ se puede expresar la condición:

if b>2:

break

Las estructuras cíclicas o repetitivas, permiten la realización de una sección de código una cantidad de n veces, o hasta que ciertas condiciones se cumplan para finalizarse. Python utiliza los ciclos *while* y *for* para este propósito. Los ciclos *while* son aquellos que se repiten una indeterminada cantidad de veces, hasta que la condición del ciclo no sea realizada [2], su condición por lo general es de naturaleza booleana. Por el otro lado, el ciclo *for* se encarga de repetir un segmento del código hasta que este llegue a un valor n especificado. Permite el uso de caracteres, valores numéricos, arreglos vectoriales, etc [2].

La sintaxis de un ciclo *while* toma la siguiente forma:

while condición_se_cumple:

realizar_código

Para un ciclo *for* la sintaxis puede variar de acuerdo a su aplicación, sin embargo, dado una forma estándar en Python se describe como:

for elemento *in* iteración:

realizar_operaciones

Figura 3. Modelo estándar de comunicaciones simplificada [3].

1.2 Protocolos de Comunicación

El propósito fundamental de un sistema de comunicaciones es el intercambio de información entre dos organismos diferentes [3]. Este tipo de tecnología funciona debido a la distribución de elementos en la transición de datos. Como cada modelo estructural, se divide en secciones que representan el intercambio de informacion. Representadas en la figura 3, se componen de 5 elementos necesarios para su comunicación.

1 **Fuente.** Generador de datos a ser transmitidos, celulares y tabletas son ejemplos actuales.
2 **Transmisor.** Medio de transporte de datos, se ven generalmente en forma de chips integrados o circuitos impresos con una antena para el envio de información. Se acostumbra a decodificarse la información a través de un sistema de transmisión.
3 **Sistema de transmisión.** Protocolo de comunicación para entablar conexión con el otro dispositivo. Entre los más utilizados se encuentran el Wi-Fi, Bluetooth, Infrarrojo, etc.
4 **Receptor.** Acepta la señal proveniente de un transmisor y la decodifica para ser tratada en el dispositivo destino. Posee la misma tecnología utilizada para transmitir información entre dispositivos.
5 **Destino.** Toma la información recibida a través del receptor. Podría ser de igual manera, un dispositivo similar al de la fuente.

El modelo anterior, ayuda a comprender lo que implica un sistema de comunicaciones. Sin embargo, a pesar de lo conciso del modelo, el intercambio y procesamiento de información posee un nivel de dificultad mayor. Por ejemplo, si se desea intercambiar información entre dos celulares, debe existir una dirección única para que el celular A se pueda comunicar con un celular B y viceversa. Por lo que, es indispensable una alta cooperación entre dispositivos. De acuerdo a [3], el intercambio de información entre dispositivos con el propósito de cooperación constante se le hace llamar *Comunicaciones Computacionales*. De igual man-

era, si se crea una red de comunicaciones compuesta por dos o más dispositivos conectados entre ellos, se le conoce como *Red Computacional*.

Para el correcto funcionamiento de las comunicaciones y la red con la que se desea trabajar, es vital la comprensión del *protocolo* de comunicación, asi como su *arquitectura*. Un protocolo es un algoritmo a seguir en comunicaciones e informática, cuyo trabajo es ayudar a lograr de manera correcta la conexión entre dos sistemas diferentes. Dichos sistemas deben poder comunicarse en el mismo idioma, en caso contrario nunca se logrará la conexión. Los elementos clave que describen a un protocolo son [3]:

1. **Sintaxis**. Formato o idioma en la que está escrita la información.
2. **Semántica**. Control de flujo de información para coordinación y manejo de errores.
3. **Tempo**. Estandariza la velocidad entre dispositivos y su secuencia de transmisión.

Es importante recordar que debe existir una alta cooperación entre los dispositivos para que se realice la comunicación, lo cual en ocasiones puede ser complicado realizarse en el mismo momento. En su lugar, se acostumbra en dividir una tarea en varias subtareas, cada una trabajando independiente de la otra [3]. A dicho esquema de tareas y subtareas se le conoce como la *arquitectura* del protocolo.

Las formas de dividir la arquitectura son diversas de acuerdo a su aplicación. Sin embargo, existe un modelo estándar bajo el cual están construidos los protocolos más utilizados hoy en día, a esta arquitectura se le conoce como un modelo a tres capas. Véase en la figura 4, en donde se posee una división entre los niveles de operación para la transferencia de datos entre dos dispositivos.

1.2.1 Codificación por formato ASCII

En la informática, existen diversas formas de cuantificar la información. En los sistemas digitales, es común la utilización de sistemas numéricos tales como el binario, octal o hexadecimal para su representación. Si bien, un computador los opera, es el sistema binario el cual

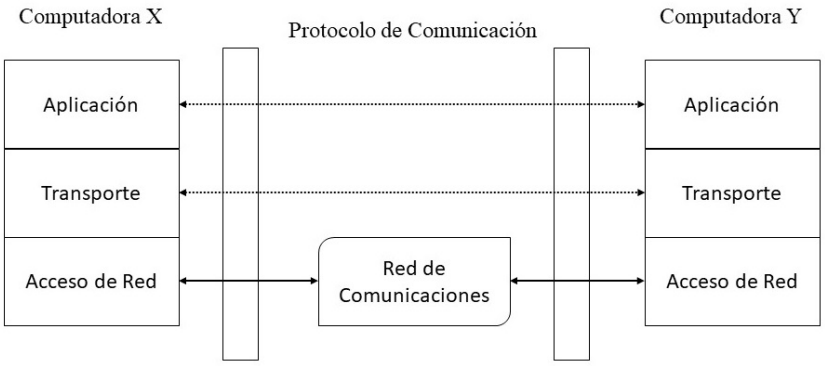

Figura 4. Arquitectura de un protocolo de comunicación a tres capas.

trabaja en sincronía con su procesador. Los sistemas octales y hexadecimales, son normalmente utilizados para representar largos grupos de dígitos binario,[5] facilitando la cantidad de información transmitida.

Los códigos binarios realizan diversas tareas dentro de un procesador, de las cuales una de ellas es la representación de información alfa-numérica. Principalmente son utilizados para expresar datos entendibles por el usuario o algún dispositivo para requiera realizar una conexión con el computador. Uno de los códigos más utilizados se le conoce como codificación en formato ASCII.

El código ASCII, (por sus siglas en ingles "*American Standard Code for Information Interchange*") es una lista numérica que representa los caracteres usados para la digitalización de texto incluyendo el alfabeto estándar, números y símbolos matemáticos.[4] El formato ASCII, es un código limitado a 7-bits, por lo que posee una capacidad limitada de representar 128 caracteres para su uso. Debido a la imposibilidad de representar un mayor número de caracteres para otros idiomas, años después se desarrolló la versión actual de 8-bits (1 byte), bajo el nombre de US ASCII-8 [5]. En el Anexo B se agrega una tabla de los caracteres ASCII actualizada.

1.2.2 Comunicación Bluetooth

La tecnología Bluetooth es una forma de comunicación inalámbrica entre dispositivos a una corta distancia [6]. Posee una estructura uniforme estable entre dispositivos con el uso mínimo de interacción por parte del usuario. Algunas características que describen esta forma de comunicación son su robustez, su baja complejidad, el uso mínimo de potencia, así como el bajo costo de construcción; siendo especialmente conveniente para su implementación en dispositivos móviles.

Su estructura se basa en la conexión estándar de un programa de conexión en redes. Una red estándar consta de 5 conceptos básicos para establecer la conexión de un cliente a un servidor:

(1)
- Escoger el dispositivo a conectar
- Escoger el protocolo de transporte y puerto de conexión
- Establecer la conexión
- Transferencia de datos
- Cierre de conexión

De igual manera, son 5 conceptos los necesarios para levantar un servidor con el que un cliente se desee conectar:

(1)
- Escoger protocolo de transporte y puerto de conexión
- Reservar recursos locales y esperar conexiones
- Aceptar conexiones entrantes
- Transferencia de datos
- Cierre de conexión

Para la comunicación bidireccional por Bluetooth existen 4 protocolos de transporte, RF-COMM, L2CAP, ACL y SCO, de los cuales solo los dos primeros son relevantes por su amplia retrocompatibilidad con diferentes sistemas operativos.

El protocolo RFCOMM ("*Radio Frequency Communications*"por sus siglas en ingles), es un protocolo basado en la reproducción de flujos de datos serial. Posee el mismo servicio y confiabilidad ofrecido por el protocolo TCP de Internet. Este protocolo fue diseñado para emular los puertos seriales como el RS-232 haciéndose sencillo su implementación con propósitos industriales, también facilitando a su vez, la conexión en los puertos seriales ya existentes [6]. La diferencia más marcada entre ambos protocolos (TCP y RFCOMM) es la cantidad de puertos disponibles para cada uno. El TCP maneja un rango de puertos hasta de 65,535 disponibles, mientras que el RFCOMM posee una capacidad máxima de 30 puertos.

El L2CAP "*Logical Link Control and Adaption Protocol*", es un protocolo de estructura en paquetes, cuyos niveles de fiabilidad pueden ser modificados de acuerdo a la aplicación del usuario [6]. El protocolo L2CAP es comúnmente comparado con el UDP debido a que manejan estructuras similares (al igual que el RFCOMM y el TCP), sin embargo, las aplicaciones del L2CAP son mucho mayores que las abarcadas por la UDP, inclusive considerándose una extensión de la misma. El L2CAP se enfoca mas en asegurar el orden de entrega de cada paquete. En la utilización del UDP al transmitir dos paquetes existe la probabilidad que el segundo paquete llegue antes que el primero enviado, esto es, debido a ciertos retardos en la ruta del Internet. Mientras tanto, los paquetes en el protocolo L2CAP siempre son entregados en el orden que fueron enviados.

1.3 Cinemática Rotacional

La cinemática se define como la descripción del movimiento de cuerpos sin tomar en consideración la inercia o las fuerzas que causan el movimiento [7]. Un ejemplo es la trayecto de una nave espacial cuyo objetivo es aterrizar en la estación espacial internacional, aquí la cinemática estará mas interesada en la velocidad, aceleración, velocidad angular, aceleración angular de la nave este sin considerar los propulsores generadores de dichas aceleraciones.

El estudio de la cinemática requiere la definición del concepto de marco de referencia. Una forma para definir lo anterior es hacer uso del ejemplo respecto a la nave espacial. Considere un observador desde la plataforma de lanzamiento de la nave el cual la mira despegar. Desde su punto de vista, la nave se aleja a una aceleración constante de la plataforma. Sin embargo, si se considera desde la perspectiva de una nave B cuya dirección y velocidad es igual a la nave A, dará una noción que la nave A no esta en movimiento. A este punto de vista o percepción relativa se le conoce como marco de referencia.

Existen diferentes marcos de referencia según sea el modelo físico que se desea realizar, aquí se introdujo el concepto de marco de referencia inerte. Un marco inerte o de referencia

inerte es aquel en donde el movimiento de la partícula no se encuentra sujeta a fuerzas, y de esta manera continua en línea recta a una velocidad constante [7]. Se puede decir que un marco inerte, está en movimiento constante según sea la cinemática del cuerpo a analizar. Su concepto es de suma importancia en la mecánica clásica, dado que la definición de la primera y segunda ley de Newton solo pueden llegar a ser verdaderas si este marco de referencia existe, sin que este marco acelere o rote.

En contraste con un marco de referencia fijo con respecto al movimiento de un cuerpo, existen los marcos de referencia rotatorios, los cuales, no obedecen las leyes de Newton directamente. Estos se utilizan para reducir la complejidad que puede derivarse de la cinemática de cuerpos con movimientos rotacionales complejos [7]. Si el movimiento que describe la cinemática de un cuerpo involucra rotaciones múltiples y dinámicas complejas, apoyarse de un marco de referencia inerte o fijo no es siempre la mejor opción. En este caso, es mucho mas sencillo usar marcos rotatorios, donde valores como las velocidades o aceleraciones angulares son descritas de una forma más simple en comparación con el comportamiento de cuerpos rígidos.

La forma estándar de poder entender el concepto de marco de referencia rotatorio es imaginarse un cuerpo rígido con un punto p fijo en su origen. El único movimiento posible sobre el cuerpo rígido es de rotación respecto a sus ejes. Si se considera un punto q en movimiento resepecto a un eje Z, dicho punto se puede representar mediante una rotación del marco de referencia fíjo, esto es, en lugar de que el punto este rotando sobre la misma referencia esto se puede representar moviendo la referencia mientras el punto q esté en movimiento, como se observa en la figura 5.

1.3.1 Matrices de Rotación

En el estudio de la mecánica del movimiento es importante conocer la posición de una partícula de acuerdo a diferentes marcos de referencia. Esto es posible con una agrupación de ecuaciones que pueden ser representadas en forma matricial, llamadas matrices de rotación.

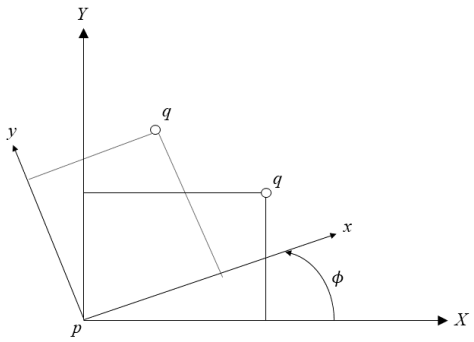

Figura 5. Rotación de un marco $F(pXYZ)$ a $G(pxyz)$ para la representación del movimiento de un punto fijo q.

De acuerdo a [7], se puede definir a la matriz de rotación como el conjunto de ecuaciones que nos permiten cambiar entre marcos de referencia libremente, respecto a un marco de referencia global [8].

Sea ϕ el ángulo rotado sobre el eje z en la figura 5, la posición del punto *p* en las referencias *F* y *G* se pueden expresar en forma vectorial:

$$F_p = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, G_p = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2}$$

Si la rotación con el ángulo ϕ es producto de la razón entre ambos marcos de referencia sobre el eje *z*, la matriz de rotación se presenta como

$$R_z = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

Donde a la matriz R_z se le conoce como matriz básica de rotación sobre el eje *z*. De igual manera podemos conocer las rotaciones sí la rotación se efectúa sobre el eje *x* o el eje *y*:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{4}$$

$$R_y = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}. \tag{5}$$

Si bien las matrices de rotación se están expresando en relación a los ángulos, una alternativa para la aplicación de matrices tridimensionales es la proyección de los ejes de referencia con respecto al marco rotatorio [9]. Usando el concepto de producto punto para la proyección de ejes podemos recrear una matriz tridimensional que cumple las especificaciones de las ecuaciones (3), (4) y (5).

$$R_{eje} = \begin{bmatrix} x \cdot X & y \cdot X & z \cdot X \\ x \cdot Y & y \cdot Y & z \cdot Y \\ x \cdot Z & y \cdot Z & z \cdot Z \end{bmatrix} \tag{6}$$

Regresando con el ejemplo mostrado en la figura 5, la representacion de la rotación del punto *p* en ambas referencias respecto al eje *z*, se puede reducir la expresión según se muestra en (7), permitiendo el traslado entre el marco de referencia global a la referencia rotada

$$F_p = R_z G_p, \tag{7}$$

o su equivalente en forma matricial

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{8}$$

y su forma general para una rotación en cualquier eje es

$$F_{rot} = R_{eje} G_{rot}, \tag{9}$$

de acuerdo al eje de rotación sobre el que se realiza es la matriz correspondiente a utilizar.

Expresiones tales como (6) y (8), se conocen como transformaciones matriciales, por su característica de cambio entre sistema de valores, en este caso respecto referencias.

En la realidad, expresar la rotación respecto a un solo eje sirve de poco, dado que muchos componentes o piezas poseen dinámicas con una alta complejidad. Con la ayuda de las transformaciones matriciales es posible analizar la cinemática de un objeto dado a una cantidad *n* de rotaciones sobre los ejes. En (10) se puede observar la expresión que representa el movimiento de un cuerpo dado a tres rotaciones consecutivas

$$R_{gb} = R_{x\theta} R_{y\psi} R_{z\phi}. \tag{10}$$

Sean las rotaciones de la figura 6, se quiere conocer el movimiento de un objeto con una transformación zxz en base a su matriz de rotación. Esto se expresa:

$$R_{gb} = R_{z\theta} R_{x\psi} R_{z\phi}$$

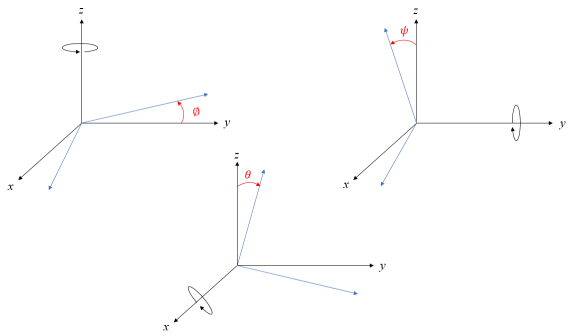

Figura 6. Rotaciones básicas sobre los ejes x-y-z con punto fijo en el origen.

La matriz R_{gb} esta definida como el producto entre las rotaciones básicas de las matrices en los ejes deseados.

$$R_{gb} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nótese que puede existir una diferencia en el signo (implicito en la función seno) de las ecuaciones básicas de rotación, el cual únicamente especifica la dirección de giro en base a su eje.

La ecuación R_{gb} nos explica la rotación de un objeto *M* en base a dos ejes, esto es, realiza rotación tras rotación en base a una secuencia Z-X-Z, cuya secuencia recibe el nombre de Transformación ZXZ de Ángulos de Euler [9].

1.3.2 Ángulos de Euler (Pitch, Yaw, Roll)

Como se había mencionado con anterioridad, una matriz de rotación *R* puede ser descrita como el producto de rotaciones sucesivas respecto a las coordenadas de los ejes principales en un orden especifico [9]. Bien se conoció el ejemplo de una secuencia Z-X-Z, el cual realiza dos rotaciones respecto a eje Z y una respecto al eje X. Los ángulos ϕ,ψ,θ que se utilizaron para representar su rotación si bien se conocen como ángulos de Euler, no son su única forma de representarse.

Una variación de los ángulos de Euler comúnmente utilizados en el estudio de la aeronáutica son los ángulos Roll, Pitch y Yaw. A diferencia de la anterior transformacion, estos ángulos siempre manejan por convención un orden de rotación X-Y-Z. El eje asociado a estos ángulos puede cambiar de acuerdo a diferentes autores [9], sin embargo, en este documento se manejará el nombramiento de ángulos de la siguiente manera:

1 Eje *x*, con ángulo θ (**Pitch**)
2 Eje *y*, con ángulo ψ (**Yaw**)
3 Eje *z*, con ángulo ϕ (**Roll**)

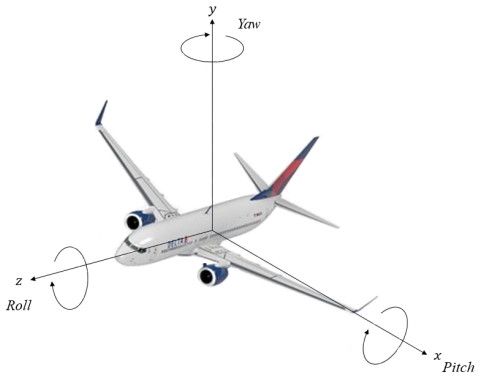

Figura 7. Notación de las transformaciones Pitch, Yaw, Roll, respecto a una aeronave.

Otra nomenclatura frecuentemente utilizada para los ángulos de Euler es la combinación Precisión-Nutación-Rotación Intrínseca. La rotación sobre el eje *z*, de una referencia global se le conoce como en ángulo de precisión. A su vez, se tiene sobre el eje *x* el ángulo de nutación y el ángulo de la rotación intrínseca en el eje *y* [8]. Cabe mencionar que la rotación en los últimos dos ejes es realizada sobre el marco de referencia actual y no el inerte.

Los ángulos de Euler de acuerdo a su dirección de giro, figura 7, se puede parametrizar para su rotación en tres dimensiones dada la siguiente transformación matricial

$$\begin{aligned} R_{gb} &= R_{x\theta}R_{y\psi}R_{z\phi} \\ &= \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \\ &= \begin{bmatrix} c(\psi)c(\phi) & c(\theta)s(\phi)+c(\phi)s(\theta)s(\psi) & s(\theta)s(\phi)-c(\theta)c(\phi)c(\psi) \\ -c(\psi)s(\phi) & c(\theta)s(\phi)-s(\theta)s(\psi)s(\phi) & c(\phi)s(\theta)+c(\theta)s(\psi)s(\phi) \\ s(\psi) & -c(\psi)s(\theta) & c(\theta)c(\psi) \end{bmatrix} \end{aligned} \tag{11}$$

En base a los ángulos de Euler *pitch-yaw-roll* es posible determinar una matriz de rotación global en X-Y-Z. De igual manera, se puede encontrar los ángulos de Euler si se conoce la matriz de rotación. Considérese una matriz de rotación *R* de la forma r_{ij} cómo se describe en (12). Igualando (11) con (12) se deduce:

$$R = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{12}$$

$$\psi = \arcsin(a_{31}) \tag{13}$$

$$\theta = \arctan\left(-\frac{a_{32}}{a_{33}}\right) \tag{14}$$

$$\phi = \arctan\left(-\frac{a_{21}}{a_{11}}\right) \tag{15}$$

Los ángulos resultantes son dependientes de acuerdo a los valores de (11). Esto es, que a su vez dependen de las rotaciones básicas en X-Y-Z. Por lo que es necesario conocer la dirección de rotación con respecto a cada eje para poder tener una matriz (11) equivalente.

1.4 Desarrollo de algoritmos de control para una señal discreta

En la naturaleza, cualquier conjunto que contenga información se le puede denominar como una señal. Considerando a la gran variedad de objetos que existe en la naturaleza, la variedad de información a su vez también deberá ser amplia, por ende, cada señal posee su propia particularidad. En general, se manejan dos tipos de señales en sistemas de control, las señales continuas o de naturaleza análoga, y las señales de tiempo discreto utilizadas en sistemas de control digital.

Primordialmente están las señales analógicas. Toda señal en la naturaleza pertenece a este primer tipo de señal y su característica que la describe es su continuidad. Durante el lapso o periodo de tiempo que la señal dure no presenta corte o intermitencias entre los puntos que la conforman. Señales como la luz, sonido, flujo de corriente electrica, energía, etc., pertenecen a este conjunto, siendo perceptibles más fácilmente que las señales discretas.

Por el otro lado, las señales discretas son utilizadas para cuantificar una señal analógica y reproducirla en una interfaz digital. Su principal particularidad a diferencia de una señal continua, es el lapso hueco que existe entre cada valor que la compone. Este tipo de señales no están definidas en un periodo de tiempo específico, en su lugar, solo se definen durante breves instantes de tiempo por cada valor contenido en la señal.[11]

En el desarrollo de sistemas de control, existen controladores tanto para señales analógicas como discretas. Las señales analógicas son continuas en el tiempo, lo que le permite ser más facilmente representadas por medio de funciones, como se muestra en la ecuación (16). Para las señales discretas, se considera no una función continua, sino un conjunto de puntos o muestras aplicándose al controlador individualmente en cada punto contenido en la señal. Debido a esto, las formas en las que se expresan los algoritmos de sus controladores son diferentes, a pesar de que en esencia realizan la misma función. A continuación, se plantean los algoritmos propuestos para expresar de manera numérica las formas algebraicas contenidas en (16).

La señal de control de un PID para una señal continua se expresa de la forma

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{16}$$

Si bien en un sistema con señales analógicas se trabaja modificando la función de entrada con respecto a su valor de salida, en sistemas digitales se considera en base a una agrupación de valores, aplicando un PID en el valor actual capturado. De esta forma el algoritmo para el control proporcional se representa en (17).

$$err = ref - y(t)$$

$$P = err * K_p \tag{17}$$

El control integral representa un arreglo de la señal utilizando los valores pasados de la misma. Una integral puede ser facilmente deducida a través del modelo de la suma de Riemann. Para el uso de la integral se utilizó una forma numérica reducida de su misma definición. Expresando su algoritmo de la siguiente manera

$$err_{acum} += err * t_c$$

$$I = err_{acum} * K_i \tag{18}$$

donde t_c representa el tiempo en el cual se aplicó el control. En cambio, la derivada describe la razón de cambio existente entre el punto de ajuste y la salida. De acuerdo a la naturaleza particular de cada sistema se puede modificar la forma original como esta descrita en el algoritmo PID. En la planta que se trabaja en esta tesis y se explicará en capítulos posteriores, la derivada de la salida, es dada en forma de medición por el sensor utilizado. Por lo tanto, de acuerdo a la definición de derivada, al ser la referencia una constante, el valor de su derivada queda en cero, obteniendo el valor negativo de su medición correspondiente.

$$D = K_d \frac{de(t)}{dt} = K_d \left(\frac{d}{dt} ref - \frac{dy(t)}{dt} \right) = -K_d * y_v(t) \tag{19}$$

De esta forma, la señal de control para un sistema discreto o digital se describe con el algoritmo

$$u = P + I + D = (K_p * err) + (K_i * err_{acum}) - (K_d * y_v(t)) \tag{20}$$

Capítulo 2

Puesta en operación de la plataforma Raspberry Pi 3 B+

En este capítulo se presentan los procedimientos realizados para la configuración de la tarjeta Raspberry Pi 3 Modelo B+. Se explica la arquitectura bajo la que está construida los SoC (System on Chip), junto con los sistemas necesarios que conllevan a su correcto funcionamiento (Software y Hardware). Se incluyen recomendaciones respecto a la instalación de drivers opcionales y sus diferentes tipos de conexiones de la programación del dispositivo Raspberry Pi (SSH, VNC, Monitor-use, etc.). Finalizando el capítulo, se da una explicación de la instalación de paquetería requeridas por el lenguaje de programación Python.

Se le conoce como SoC (System on Chip), a los controladores de tamaño reducido que pueden realizar una gran cantidad de operaciones y poseen en su mayoría los componentes de un computador normal. Con el aumento exponencial de las nuevas tecnologías mostradas en los últimos años, estos fueron distribuidos a gran escala, siendo de fácil acceso y encontrados en cualquier lugar de manera comercial. Los celulares y tabletas utilizadas cotidianamente poseen este tipo de tecnologías.

Durante los últimos 50 años [12], el área de la electrónica y computación se han ido desarrollando con gran velocidad, debido a la ayuda mutua brindada entre ambas áreas. La invención del transistor y el nacimiento de las nanotecnologías dieron la llegada de los microprocesadores, también conocidos por ser la base de los SoC. Sin embargo, no todo fue desarrollo exclusivo del área electrónica; dentro de la computación se produjeron sistemas operativos y software que optimizaron el hardware brindado por la electrónica. Los cambios más grandes en la computación se vieron en la arquitectura y la organización de las computadoras.

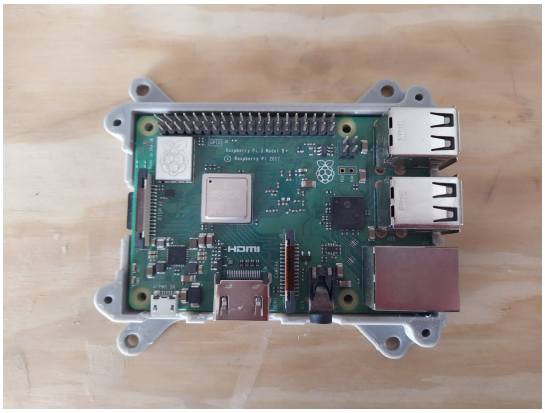

Figura 8. Tarjeta SoC Raspberry Pi 3 Modelo B+

La arquitectura de un computador es aquella que describe la vista del usuario dentro del mismo. Instrucciones del procesador, registros visibles, manejo de memoria, etc., son algunos ejemplos [12], mientras que la organización de un computador es la parte invisible para el usuario realizada en segundo plano por la arquitectura; cache transparente, espacios de memoria (buffer), etc., son algunas de las más representativas.

En la actualidad, el uso de los SoC dejó de ser limitado para sus desarrolladores y aquellas personas que tengan conocimientos especializados para su uso. Con la salida al mercado de las SoC didácticas (Raspberry Pi, BeagleBone Black), ya no fue requerido conocer a detalle su funcionamiento para operarse. Estas tarjetas se desarrollaron con la intención de acercar a los jóvenes a tópicos como la electrónica o la programación. Los experimentos descritos en este documento se realizaron a través de una Raspberry Pi 3 B+ (figura 8), por lo que, las características de software y hardware de secciones posteriores, seran respecto a la tarjeta en cuestión.

2.1 Características del hardware

La tarjeta Raspberry Pi 3 Modelo B+, es un SoC perteneciente a la tercera generación de Raspberry Pi, desarrollada por Raspberry Pi Foundation, siendo este modelo la versión de-

finitiva de la tercera y el introductorio a la cuarta generación. Se le considera un SoC, por su integración de CPU, GPU y memoria todo incluido dentro del mismo chip. Este posee un hardware con las siguientes especificaciones [17]:

1 Procesador: Broadcom BCM2837B0, Cortex-A53 64-Bit SoC @ 1.4GHz
2 RAM: 1GB LPDDR2 SDRAM
3 Memoria: Puerto de soporte micro SD
4 Conectividad: 2.4 GHz - 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2 BLE
5 Gigabit Ethernet a USB 2.0
6 4 Puertos USB 2.0
7 Uso de hasta 40 Pines para GPIO
8 Puerto de video HDMI
9 Puerto de cámara MIPI CSI
10 Salida de estéreo de 4 polos
11 H.264, decodificador MPEG-4 (1080p30)
12 Gráficos 2.0 OpenGL ES 1.1
13 Alimentación: 5V/2.5 A DC vía conector micro USB
14 5V vía puerto GPIO
15 Alimentación a través de puerto Ethernet (PoE)
16 Rango de operación óptimo: Temperatura: $0 - 50^{\circ}$

La amplia variedad de opciones que ofrece para comunicarse (VNC-WiFi, Bluetooth), así como el fácil acceso a sus periféricos programables (GPIO) de la tarjeta, la hacen el programador ideal para realizar con mayor facilidad tareas complejas dentro del rango didáctico.

2.2 Sistema operativo

Un sistema operativo es un programa que actúa como intermediario entre la interfaz de usuario y el hardware de un computador [13]. También, se define como un administrador que se encarga de modular el hardware y que este trabaje correctamente. Los sistemas operativos asumen una gran importancia en la optimización del hardware de un computador, y en muchos casos el computador no es funcional si no existe este sistema.

La tarjeta Raspberry Pi a diferencia de otras tarjetas de programación portátiles (Arduino, NodeMCU) posee un microprocesador en lugar de un microcontrolador. Esto le da una alta

capacidad de procesamiento alcanzando velocidades mucho mayores, sin embargo, coordinar cada tarea individualmente llega a ser complejo, requiriendo de un sistema operativo que le ayude administrando las tareas de manera secuencial. Raspberry posee un sistema operativo gratuito optimizado para el uso de la tarjeta llamado Raspberry Pi OS (anteriormente conocido como Raspbian).

Existen diversas formas de instalar el sistema operativo a la tarjeta Raspberry Pi. Dada su fácil accesibilidad se utilizo el programa de paquetería NOOBS para instalar el sistema operativo. El software NOOBS (*New Out Of Box Software*), es un administrador de sistemas operativos, donde de acuerdo a las necesidades del usuario, se instala el sistema que se requiera. Hay dos formas de utilizar la paqueteria NOOBS, a través de una instalación normal y con la forma Lite. Mientras que la instalación regular suele no requerir conexión a internet para la instalación, NOOBS Lite necesita estar conectada a través de Ethernet a una red para que se complete el proceso, esto es debido a que no posee descargado previamente los sistemas operativos, por lo que accede directamente a la red para instalarlos.

La instalación del sistema operativo se llevó a cabo usando el protocolo de seguimiento de la pagina oficial a través del software NOOBS. Es recomendable una buena conexión a internet y una tarjeta micro SD de 16 Gb para asegurar que no ocurran errores de instalación. La instalación se puede dividir en tres secciones principales:

(1) Archivos de paquetería NOOBS en micro SD
(2) Configuración del chip Raspberry Pi
(3) NOOBS en Rasperry Pi

En primera instancia, se descargó directamente desde la página oficial, su archivo comprimido de la versión de NOOBS Lite. La información contenida se debe transferir a una micro SD formateada, es decir, sin que esta contenga ningún archivo dentro de ella. Si se utiliza una tarjeta micro SD con una capacidad igual o mayor a 64 Gb, necesita ser formateado para ajustarse al tipo de archivo de sistema FAT.

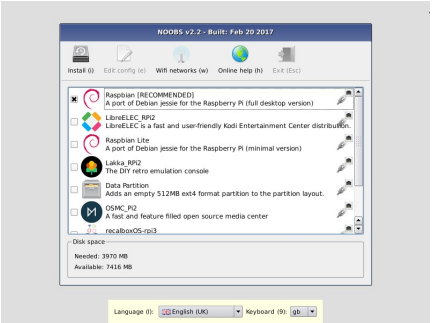

Figura 9. Interface de Instalación de sistemas operativos NOOBS.

El **bootloader** (sistema de arranque) de la Raspberry Pi soporta solo la lectura para archivos tipo FAT (FAT16, FAT32). Dado que las tarjetas SD de 64 Gb o mayores cuando son formateadas por defecto poseen el tipo de archivo exFAT no se puede instalar el sistema operativo, e inclusive no arrancar el instalador NOOBS. Para solucionar lo anterior, se requiere reformatear la tarjeta al tipo de formato requerido ya sea FAT16 o FAT32. Una vez ajustado el formato, se transfieren nuevamente los archivos a la micro SD.

Cuando se arranque la Raspberry Pi por primera vez, se iniciará de forma automática el instalador del menú NOOBS como se muestra en la figura 9. Dentro del instalador, se encontrarán diversos sistemas operativos disponibles para su uso (Raspberry Pi OS, LibreELEC, OSMC, Recalbox, Lakka, RISC OS, etc.). De acuerdo a la versión de NOOBS que se posea, se debe instalar la opción más parecida entre las siguientes: Raspberry Pi OS, Raspbian o Debian. Aceptar los términos y esperar a que termine la instalación. El sistema operativo se habrá instalado correctamente una vez este se reinicie y aparezca la pantalla mostrada en la figura 10.

2.3 Configuraciones de software

Con un sistema operativo ya instalado, se puede trabajar sin inconvenientes si se le mantiene conectado como un computador normal con todos sus periféricos. La Raspberry Pi ofrece diversas herramientas para facilitar su uso y sobre todo la comodidad del usuario a través del

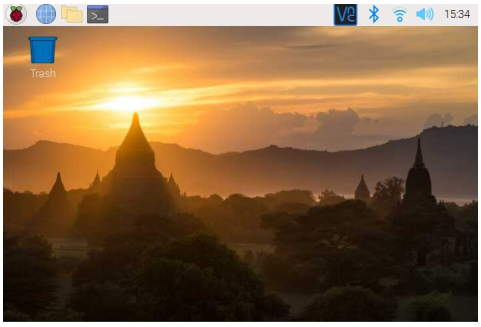

Figura 10. Interfaz gráfica predeterminada del Sistema Operativo.

acceso inalámbrico. Las formas de adquirir acceso inalámbrico más utilizadas son a través de un SSH o un VNC.

VNC proviene de las siglas en inglés *Virtual Network Computing*, el cual es un software de distribución libre que permite observar desde un monitor cliente los procesos realizados por un monitor servidor conectados en una misma red. La Raspberry Pi posee instalada el software por defecto para actuar como un servidor, por lo que es posible programarse con un computador externo sin la necesidad de utilizar accesorios o sus periféricos.

Mientras que un VNC hace referencia a un software, por el otro lado, un SSH es un protocolo. **SSH** o *Secure Shell* es el nombre que recibe un protocolo de comunicación, el cual permite el acceso remoto a un servidor a través de un canal de información encriptada. Conectados a una red por medio de su IP, se transfiere y recibe información en un puerto en común. Dado que la información esta encriptada, este es más seguro que un VNC, sin embargo, el VNC ofrece la facilidad de interfaz gráfica, siendo más amigable con el usuario.

Por defecto, estos comandos están deshabilitados en el sistema operativo Raspberry Pi OS. Para su utilización, se requiere instalar los paquetes internos para la programación inalámbrica, así como, habilitar las interfaces requeridas. En primer lugar, se debe ingresar en la consola de comandos (en Raspberry Pi se le conoce como Terminal) el enunciado

$$sudo\ apt\text{-}get\ install\ xrdp \tag{21}$$

el cual, instalará un paquete requerido dentro del sistema operativo para habilitar el control de un escritorio virtual y su control en forma remota. En la instrucción (21), *sudo* permite acceder a los archivos de raíz en la Raspberry, lo que indica que se otorga el acceso de administrador para la instalación de los paquetes. La sección *apt-get*, enlista el repositorio de paquetes disponibles para el sistema operativo; de acuerdo al tipo de paquete existen diversas instrucciones que se le pueden dar a cada uno. En este caso, se desea instalar el paquete para la creación de un escritorio virtual, por lo que su sintaxis se divide entre la acción que se desea

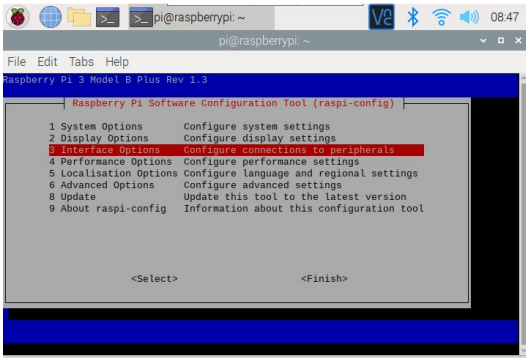

Figura 11. Panel de configuraciones de la Raspberry Pi 3 B+.

realizar y el nombre del paquete (*install xrdp*). Una vez instalado, se recomienda reiniciar el OS para que los programas descargados sean cargados de manera adecuada.

Lo anterior proporcionará las herramientas que se necesitan para conectarse por medio de un VNC, sin embargo, si este medio de comunicación no es habilitado desde la Raspberry Pi, no se dará la transferencia de información. Para activarse las características de comunicación remota, desde el menú principal en la sección de *Preferences*, se selecciona la aplicación de *Raspberry Pi Configuration*. Ahí aparecerá una ventana como en la figura 11, con características diferentes según sea la configuración a efectuar.

Para habilitar su comunicación y los puertos de transferencia, se ingresa a la sección de configuración *Interface*. En ella, se enlistan diferentes configuraciones de los parámetros de la tarjeta. Dado que sus características están desactivadas por defecto, revise que las siguientes configuraciones esten habilitadas, debido a que nos serán útiles para procesos posteriores.

1 VNC
2 Serial Port
3 Serial Console

Si se desea ingresar directamente a las configuraciones a través de la terminal de comandos, ingrese el enunciado *sudo raspi-config*. Esto abrirá de igual manera como lo realizado

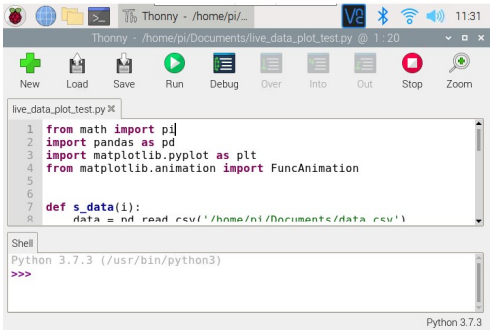

Figura 12. Interfaz de programación Thonny Python IDE.

anteriormente la *Raspberry Pi Configuration* (figura 11), ofreciendo un acceso rápido a la modificación de sus configuraciones.

2.3.1 Python 3 IDLE

El Raspberry Pi OS posee instalado por defecto las herramientas para utilizar Python en su sistema. Como se menciono en el capítulo 1, debido a problemas de compatibilidad se dividió el Python entre las versiones 2 y 3. El sistema operativo incluye instalada ambas versiones de Python, por lo que se recomienda tener cuidado en la instalación de paqueterías y extensiones requeridas por sus lenguajes.

El OS posee instalado por defecto una aplicación con nombre de Thonny Python IDE, ambientado para un entorno en Python 3. Esta aplicación permite la interacción con el lenguaje Python de forma sencilla para aquellos que comienzan su estudio. A pesar de ser una interfaz completa y amigable al usuario, en el capítulo 3 se requerirá la implementación de programas en Python con ejecuciones independientes, siendo necesario la utilización de un IDLE de Python diferente al IDE de Thonny.

Si bien, existen diferentes interfaces de programación para Python, el OS instalado posee una segunda interfaz oculta entre los archivos del sistema, la interfaz por defecto ofrecida por el desarrollador de Python. Al ser un lenguaje de licencia libre, esta interfaz puede resultar un

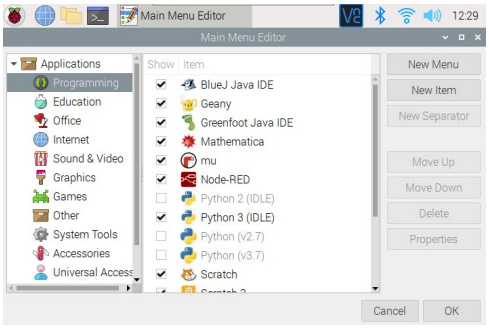

Figura 13. Editor del menú de aplicaciones.

poco menos amigable en comparación con Thonny, dado que operan bajo el mismo lenguaje con versiones idénticas resulta útil su implementación.

Para hacer visible la interfaz de Python 3 IDLE, se debe ingresar a través del menú principal en la pestaña *Preferences* a la aplicación *Main Menu Editor*. En ella, se observan diferentes programas instalados en cada una de las pestañas del menú, véase que existen diferentes aplicaciones no marcadas en cada pestaña del editor. En la pestaña de *Programming*, buscar la opción de Python 3 IDLE y marcarla como se muestra en la figura 13, esto hará visible su interfaz de programación en dicha pestaña. Si esta no aparece, se recomienda reiniciar el sistema para que los cambios se realicen correctamente. En algunos casos, no esta instalado el IDLE en el sistema operativo, lo cual se resuelve escribiendo en la terminal el comando *sudo apt-get install python3* para instalarlo directamente. Este comando también puede ser utilizado para hacer visible la interfaz gráfica, evitando usar el editor del menú principal.

2.3.1.1 Librerías de Python

Las diferencias que existen entre las versiones de Python 2 y Python 3 no se delimitan a su sintaxis, sino que afecta directamente a las funciones utilizadas en el código fuente, por lo que se podrian considerar dos lenguajes de programación diferentes. Como se mencionó con anterioridad, se poseen ambas versiones de Python en el OS, por lo que se es necesario especificar a cuál versión se instalará la librería.

Existen dos formas para instalar una librería de Python en la Raspberry Pi OS; a través del repositorio oficial de Python (*PyPI*), y con el uso de la terminal de comandos (*apt-get*) ofrecida por el sistema operativo (como fue instalado el archivo *xrdp* del VNC). Ambas opciones son viables si se cumplen los requerimentos para su instalación. El *PyPI* es un gran repositorio de librerías y de fácil acceso, sin embargo, el repositorio de Ubuntu (*apt-get*) está ambientado para librerías compatibles con la tarjeta de desarrollo que se trabaja, a diferencia del *PyPI* donde pueden existir librerias creadas para otro tipo de dispositvos o SoC con características específicas.

La instalación de librerías con el uso del repositorio de Ubuntu sigue una sintaxis de la forma (22):

sudo apt-get install python3-NombreLibreria (22)

En la comunidad de internet se acostumbra a utilizar *python* en lugar de *python3* en la sintaxis. Este identificador nos ayuda para separar la versión objetivo de la libreria, siendo la sintaxis *python* para la versión 2 del lenguaje.

El Python 3 instalado en la Raspberry, posee una gran variedad de librerías instaladas, de las cuales, algunas de las librerías utilizadas no se poseen por defecto en el lenguaje. Lo que se presenta a continuación son las librerías utilizadas no incluidas por defecto para el desarrollo del proyecto.

Bluetooth. Librería que da acceso a los controles y protocolos para la utilización de las herramientas bluetooth incluidas en la Raspberry Pi. La librería se puede instalar en la terminal con el comando *sudo apt-get install python3-bluez*.

Pandas. Permite la manipulación de archivos (.txt, .csv). Comando de instalación: *sudo apt-get install python3-pandas.*

Matplotlib. Repositorio para la manipulación de ecuaciones y variables matemáticas utilizados en la creación de gráficos. Comando de instalación: *sudo apt-get install python3-matplotlib.*

Pyplot. Subsección de matplotlib. Creación de gráficos y herramientas para su manipulación.

Animation. Subsección de matplotlib. Crea un gráfico que se actualiza respecto a los cambios detectados en sus vectores.

Capítulo 3

Conexión, configuración y lectura del sensor 3 Space MBT

En el capítulo anterior, se dio la explicación sobre el funcionamiento básico de una Raspberry Pi, así como, la serie de pasos realizados para su operación, sus componentes físicos, el sistema operativo a utilizar, y las preparaciones para pasos posteriores fueron principalmente descritos. Se complementó con la instalación de librerías en Python para la comunicación inalámbrica de un sensor 3-Space MBT. Este capítulo tiene como objetivo conocer el sensor MBT, su estructura, protocolos de comunicación y la propuesta de una metodología para la transferencia de datos a través de Python.

El sensor de Yost Labs 3-Space Sensor Mini Bluetooth (TSS-MBT), es un dispositivo de tamaño reducido con alta precisión, confiablidad y conectivdad inalámbrica, [18] que hace uso de un giroscopio, acelerómetro y un compás con componentes triaxiales para expresar la orientación de un objeto. Con el uso de filtros basados en algoritmos de cuaterniones se puede determinar la orientación relativa respecto a una referencia absoluta en tiempo real. Posee internamente una batería de litio (LiPo) recargable con una duración aproximada de 4 horas, así como, una alta eficiencia en el censado de la orientación.

La orientación puede ser retornada por el sensor en su valor absoluto o respecto a una referencia especificada [18]. Además de los algoritmos basados en cuaterniones, utiliza compensaciones de estabilidad dinámica, cuya funcionalidad es mejorar la precisión y reducir el error del sensor. El protocolo de comunicación del sensor permite el acceso a sus datos y a la configuración de sus parámetros. Con la serie de comandos que se encuentran en [18], se puede acceder a los datos sin procesar, datos normalizados, datos filtrados, etc., según requiera el usuario para una aplicación.

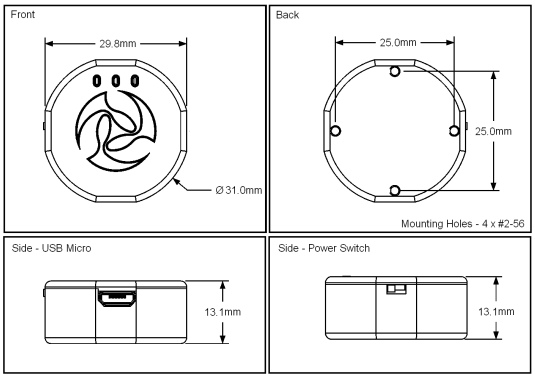

Figura 14. Componentes y dimensiones de hardware externo [18].

Dado su gran versatilidad el modulo TSS-MBT posee un gran rango de aplicaciones en diversas áreas. Algunas de las mas utilizadas se muestran a continuación:

1 Robótica
2 Captura de movimiento
3 Orientación de la posición de un objeto
4 Gaming y control de movimiento
5 Educación
6 Monitoreo de la salud
7 Análisis de vibraciones
8 Realidad virtual y simulaciones de inmersión

3.1 Arquitectura

El hardware externo de operación para el sensor es simple en comparación con la cantidad de operaciones que realiza. Este consta de tres indicadores LED en su parte frontal. El central indica la espera de una conexión entrante, el izquierdo representa que la pila del sensor esta siendo cargada (el indicador es un led de color naranja), cuando la pila esta completamente cargada se enciende el indicador derecho con un color verde. A su vez, en su sección trasera posee 4 orificios para montarse en el mecanismo en donde se va a utilizar, estos son para tornillos del tipo #2-56. En el costado izquierdo se localiza el interruptor de encedido, así como en su parte inferior se encuentra el conector micro USB 2.0 de carga y transferencia de datos, como se observa en la figura 14.

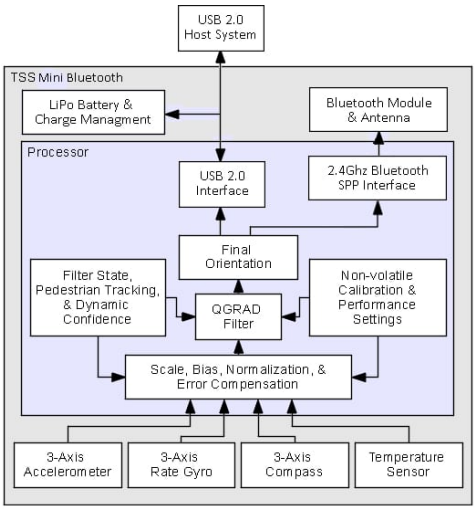

Figura 15. Diagrama de bloques del funcionamiento del sensor TSS-MBT. [18]

En el caso del hardware interno es posible clasificarse entre la información a capturar (sensores) y los módulos de procesamiento de datos. Como se había mencionado anteriormente, el TSS-MBT posee diversos sensores integrados para conocer la orientación de un objeto, los cuales se componen de un acelerómetro, un giroscopio y un compás, todos ellos con componentes triaxiales. También, posee un sensor de temperatura, el cual se utiliza para regular el calor generado por el procesamiento de datos.

Dentro de los módulos de operación se cuenta con una batería LiPo, con su respectivo centro de carga, y el modulo con antena Bluetooth. La operación de estos módulos son dependientes de un procesador además de estas funciones, cuyo procesador ejecuta los algoritmos de los datos que generan toda la información que se puede extraer del sensor. En la Figura 15 se ve de forma más detallada el diagrama de bloques para el flujo de información.

Las especificaciones relacionadas a la sensibilidad del acelerómetro, resolución del giroscopio, así como la precisión de la orientación y otros parámetros eléctricos se pueden encontrar en la hoja de especificaciones técnicas y manual de usuario del TSS-MBT [18].

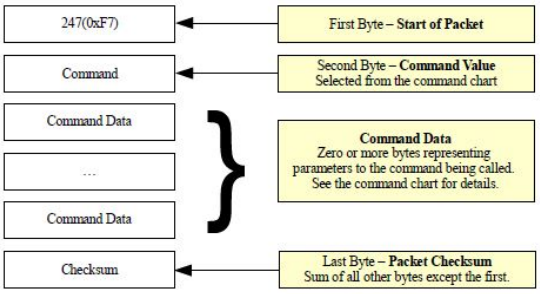

Figura 16. Formato de transferencia de datos del sensor TSS-MBT. [18]

3.2 Transferencia de datos

En este trabajo de investigación, se utiliza la transferencia de datos a través del protocolo de comunicación Bluetooth, en particular con el protocolo RFCOMM - Bluetooth, el cual permite simular un puerto como si fuera una comunicación serial por USB. En [18] se presenta en detalle la forma en que se realiza la transmisión de datos que se resumirá a continuación.

La Figura 16 muestra la forma en que se lleva a cabo la transferencia de datos. En primer lugar se envía el byte con el número decimal 247, el cual indica el inicio de transmisión de datos. Posteriormente se envía un byte de comando (en [18] se encuentra una lista de todos los comandos disponibles), este byte es la dirección única donde se almacenan la información contenida en los registros del sensor. Una vez aceptado el byte de comando, el sensor retorna los bytes de información contenidos en su registro elegido para finalmente mandar un byte de *CheckSum,* cuyo byte suma los bytes anteriores sin tomar en cuenta el byte inicial, sirve para verificar que no se hayan perdido datos durante la transmisión.

En Python transmitir la información en formato binario suele ser complicado y puede conllevar una gran cantidad de procesos. Es por eso que el sensor utiliza un formato para paquetes ASCII, también conocido como formato de separación por comas. La comunicación se realizó de la forma *dar y recibir*, es decir, se le hace llegar al sensor una cadena de caracteres con el formato de transferencia y el sensor retorna (si es necesario retornar), la información

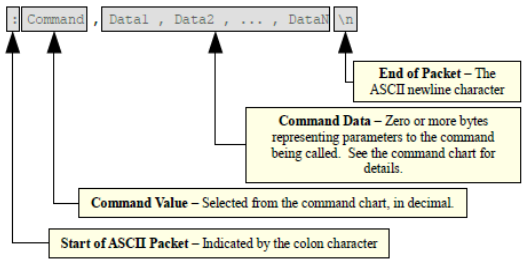

Figura 17. Estructura de comandos para formato en paquetes ASCII.[18]

referente al comando ingresado. Un ejemplo de la nomenclatura utilizada es *":1\n"*, donde el ": "indica el inicio de un paquete en código ASCII, el número "1 .es el comando de la acción deseada (en este caso se desea obtener los ángulos de Euler) y por último el "\n "indicando el fin del paquete de información. En la figura 17 se ilustra el formato de los paquetes de datos en el sensor.

3.3 Pruebas de transferencia de datos entre el sensor y la Raspberry

Para extraer y visualizar los datos del sensor a través del protocolo Bluetooth de la Raspberry Pi se requiere realizar las siguientes acciones: Importar al sistema operativo de la Raspberry todas las librerías necesarias, configurar el puerto de Bluetooth, y hacer la extracción de los datos, almacenarlos en un archivo y graficarlos en pantalla.

Para la instalación de las librerías se deben ejecutar los siguietes comandos desde una ventana del sistema en Raspberry:

1 `import csv`. Librería que permite el manejo de archivos csv.
2 `import sys`. Librería de funciones que trabajan con el sistema en conjunto y pueden modificar sus parámetros.
3 `import time.` Funciones para la manipulación temporal del programa.
4 `from math import pi.` Módulo que contiene funciones matemáticas predefinidas. Incluye la función Pi desde la libreria math.
5 `import pandas as pd`. Se importan funciones para la manipulación de achivos ya creados.

6 `import matplotlib.pyplot as plt`. Se importa un módulo que realiza operaciones matemáticas con gráficos (matplotlib).
7 `from matplotlib.animation import FuncAnimation.` Importa la función FuncAnimation para la creación de gráficos a tiempo real.

`from bluetooth import *.` Manejo del protocolo de comunicación Bluetooth.

Cada una de las librerias enlistadas anteriormente forman un conjunto de herramientas que permiten la adquisición de datos, su procesamiento, almacenamiento y su visualización a través de gráficos.

Como se puede observar, en Python es posible utilizar solo una función de la librería sin la necesidad de cargar todas las funciones de esta, haciendo el proceso del programa más optimizado en comparación con otros lenguajes, esto se hace con la instrucción `from "Librería"import "Función"`.

Cabe recordar que Python es un lenguaje que puede trabajar objetos, por lo tanto, trabajar la librería con su nombre completo puede ser un desgaste visual para el programador. En dados casos, se utiliza la estructura `import "Librería.as "Identificador"` para evitar escribir grandes enunciados de código, como se observa en `matplotlib.pyplot`.

Por último, también se utilizó la sintaxis `from "Librería"import *`. Este enunciado permite utilizar todas las funciones contenidas dentro de la librería, con la característica que a pesar de ser funciones de objetos se importan como funciones globales del programa. Esta acción ofrece seguridad de poder utilizar las funciones dentro de la librería no como un atributo de la misma sino que en su lugar realiza su tarea de forma independiente.

Lo primero a realizar es buscar la dirección del sensor por el puerto Bluetooh, obtener su dirección y hacer la conexión. Como se explicó en el Capítulo 1, se necesita de una dirección y un puerto para poder entablar la comunicación. Es común que se conozca solo el nombre del dispositivo y no la dirección, por lo que la primera acción es encontrar el dispositivo con

el nombre deseado y posteriormente obtener su dirección. Para encontrar los dispositivos Bluetooh cercanos se usa la función `discover_devices(),` se almacena en un vector toda la información acerca de los dispositivos cercanos a la Raspberry Pi. Usando el nombre del dispositivo, se puede buscar dentro del vector la dirección Bluetooth del sensor, en caso de no encontrarse el programa se termina.

El siguiente fragmento describe las configuraciones Bluetooth para la conexión con el sensor.

```
def BluetoothConfig():
    target_name = "YostLabsMBT"
    target_address = None
    nearby_devices = discover_devices()
    for bdaddr in nearby_devices:
        if target_name == lookup_name( bdaddr ):
            target_address = bdaddr
            break
    if target_address is not None:
        print("Found target bluetooth device with address ", tar-
get_address)
        time.sleep(1)
```

```
else:
    print(Çould not find target bluetooth device nearby ")
    time.sleep(1)
    sys.exit(0)
```

Ahora que se conoce la dirección del sensor se debe buscar el puerto común de conexión. Este puede ser fácilmente encontrado con el uso de un *Service Discovery Protocol* (SDP). El SDP es un protocolo de intercambio de información en donde se le pregunta a un servidor, por cual puerto se es posible realizar la comunicación. En Python se debe buscar si dicho "Servicio.[es]ta disponible en el dispositivo, si es encontrado, se retorna el nombre, dirección y puerto de comunicación. Teniendo la dirección y su puerto solo hace falta definir el tipo de protocolo Bluetooth a utilizar. Python solo soporta L2CAP y RFCOMM, por lo que, debido a su estructura fácil de entender se utilizó el RFCOMM o también conocido como simulador RS-232. El siguiente código realiza las acciones anteriores.

```
service_matches = find_service(address = target_address)
if len(service_matches) == 0:
    print(Çouldn’t find the service ")
    time.sleep(1)
    sys.exit(0)
first_match = service_matches[0]
port = first_match["port"]
```

```
    name = first_match["name"]
    host = first_match["host"]
    print(Çonnecting to ", target_address)
    time.sleep(1)
    global sock
    sock = BluetoothSocket( RFCOMM )
    sock.connect((host, port))
```

Para el manejo de archivos se utilizaron dos funciones diferentes. La primera crea un archivo con sus encabezados por cada valor que se desee guardar (`CSV_Config()`), mientras que la segunda función indexa los valores en el archivo con la división realizada en sus previas configuraciones (`CSV_Write()`). Estas se muestran en el siguiente código.

```
def CSV_Config():
    global fieldnames
    fieldnames = ["time", z1 ", z2 ", z3 "]
    with open ('/home/pi/Documents/c_data.csv', 'w') as csv_file:
        csv_writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
        csv_writer.writeheader()
def CSV_Write():
```

```
    with open('/home/pi/Documents/c_data.csv', 'a') as csv_file:

        csv_writer = csv.DictWriter(csv_file, fieldnames = field-
names)

        info = {

            "time":   t,

            z1 ":   y1,

            z2 ":   y2,

            z3 ":   y3

            }

            csv_writer.writerow(info)
```

La transferencia de información se realiza siguiendo una secuencia de envio, ajuste y división de los datos. De acuerdo a la tabla de comandos se obtienen los ángulos de Euler con sintaxis (":1\n"). Este comando es enviado al sensor en formato ASCII y se retorna en bytes a través de un socket. Si bien es posible recibir los datos ajustando el buffer del socket en *recv*, esto aumenta a su vez el tiempo de muestreo entre cada dato. En cambio, si se toma cada carácter y se concatenan hasta que la cadena de datos este completa, es posible obtener el tiempo mínimo de muestreo entre cada valor.

Cada cadena de los ángulos de Euler posee un formato

Euler Angles = ("Pitch,Yaw,Roll\r\n")

Con el siguiente código se separa cada valor para almacenarse en el archivo csv.

```
data = " "
t1 = time.time()
try:
    print("Loading Data...")
    while True:
        sock.send(":1\n")
        val = " "
        while 1:
            data = sock.recv(1).decode()
            t2 = time.time()
            t = round(t2-t1, 6)
            val = val + data
            if data == "\r":
                break
        {
            l = len(val)
            y = val[0:l]
```

```
            cf = y.find(',')
            y1 = float(y[0:cf])
        }
        yk = y[cf+1:l]
        cf2 = yk.find(',')
        y2 = float(yk[0:cf2])
        {
            yf = yk[cf2+1:l]
            cf3 = yf.find('\r')
            y3 = float(yf[0:cf3])
        }
        CSV_Write()
except KeyboardInterrupt:
    pass
```

Con la excepción `KeyboardInterrupt` se puede terminar la ejecución del ciclo principal y así pasar a la última etapa, graficar los datos respecto al tiempo. Con el uso de la función `FuncAnimation` se manda llamar los datos del archivo extraídos en `s_data` para que se muestren en pantalla.

```
def s_data(i):
    data = pd.read_csv('/home/pi/Documents/c_data.csv')
    {
        t = data['time']
        y1 = data['y1']
        y2 = data['y2']
        y3 = data['y3']
    }
    plt.cla()
    {
        plt.plot(t, y1, label = 'Pitch')
        plt.plot(t, y2, label = 'Yaw')
        plt.plot(t, y3, label = 'Roll')
    }
    plt.legend(loc='upper left')
    plt.ylim([-pi,pi])
    plt.grid(True)
```

La transferencia se termina con los últimos enunciados siguientes:

```
ani = FuncAnimation(plt.gcf(), s_data, interval = 1)

plt.show()

sock.close()
```

En las figuras 18 y 19 se puede observar los gráficos creados a partir de los datos del sensor. Nótese que los valores en los ángulos de Euler permanecerán constantes, hasta que una fuerza externa perturbe su posición, cambiando la coordenada del eje. A su vez, se tomo una muestra del comportamiento de velocidad en el giroscopio, esto se realizo respecto a la misma variación de los ángulos de Euler.

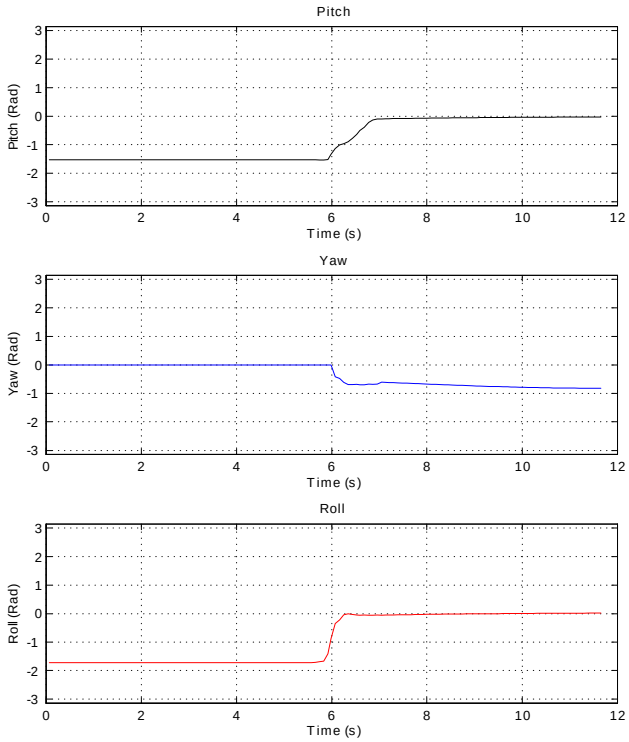

Figura 18. Ángulos de Euler de acuerdo a un cambio unidirecciónal en la posición.

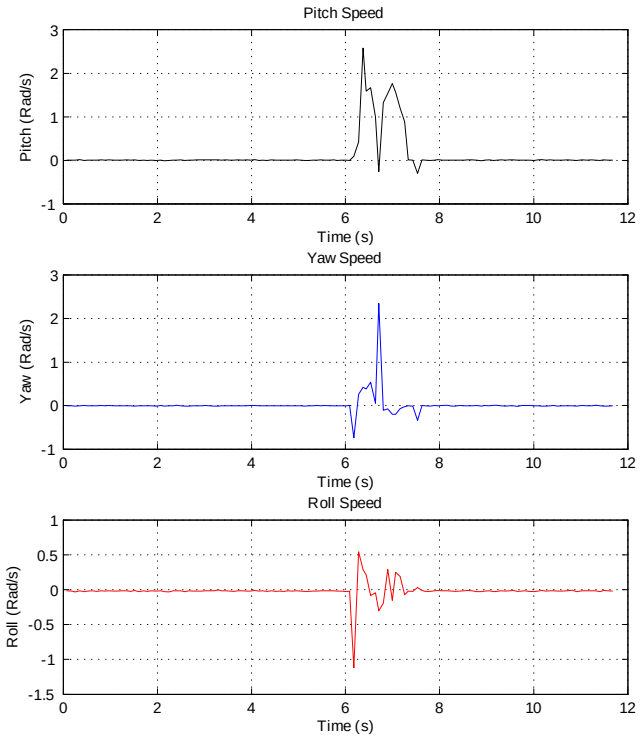

Figura 19. Variación de velocidad del giroscopio respecto a los ángulos de Euler.

Capítulo 4

Configuración y algoritmos para el uso de las terminales del GPIO para el control de actuadores basados en PWM

En el sistema mecánico que se va a controlar, que se explicará en detalle en capítulos posteriores, se utilizan resortes de Nitinol como actuadores, los cuales son una aleación con memoria de forma. Este tipo de materiales recupera su forma original al alcanzar cierto nivel de temperatura, la cual puede ser generada de diferentes formas. En nuestro caso, esta temperatura se generará por el paso de una corriente a través del material.

Para controlar el nivel de temperatura se debe de controlar el nivel de corriente, lo cual puede hacerse utilizando la modulación por ancho de pulso, mejor conocida como PWM. Esta técnica de control de temperatura en aleaciones con memoria de forma se ha reportado en diferentes trabajos, por ejemplo en [14], [15] y [16].

Ya que la modulación por ancho de pulso es muy utilizada para el control de velocidad de motores y otras aplicaciones, muchos dispositivos, como la Raspberry Pi, tienen funciones predeterminadas y terminales dedicadas para la generación de las señales PWM, las cuales pueden ser utilizadas para la activación de los elementos de potencia que suministran la corriente en los actuadores.

En este capítulo se describen aspectos básicos de la modulación por ancho de pulso PWM así como su implementación en la plataforma Raspberry Pi.

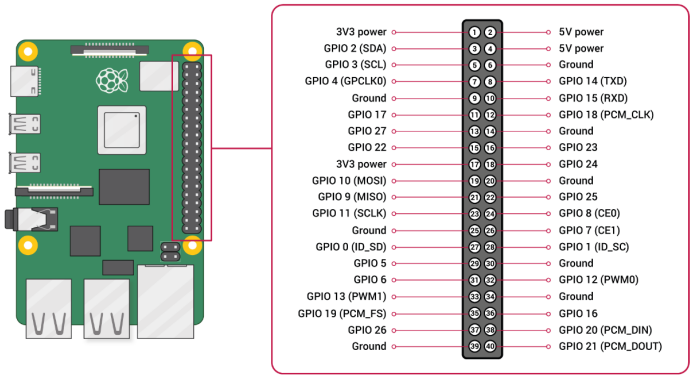

Figura 20. Distribución de las terminales del GPIO para una Raspberry Pi 3 B+ [17].

4.1 PWM en Raspberry Pi

La microcomputadora Raspberry Pi 3 B+ posee terminales reprogramables llamadas GPIO (General Purpose Input/Output). Estas terminales se pueden utilizar como entradas o salidas digitales según se requiera en el proceso a realizar; el GPIO se muestra en la figura 20. Obsérvese que dentro de las terminales que componen el GPIO también se encuentran terminales de polarización de 5V, 3.3 V y GND, así como terminales dedicadas a protocolos de comunicación y otras funciones específicas.

En lo que se refiere al PWM, la Raspberry puede generarlo por hardware y por software. Si se desea generarlo por hardware el GPIO cuenta con terminales predefinidas las cuales se conectan a un módulo en la tarjeta que genera el PWM, esta estrategia permite una mayor velocidad en comparación con el PWM generado por software. Para el caso del PWM generado por software, se cuenta con mayor flexibilidad en la programación y se pueden asignar otras terminales en GPIO para la salida del PWM, aunque la frecuencia máxima de operación es menor a la alcanzada por hardware.

En la aplicación de control en lazo cerrado que se desarrolla en esta tesis no se necesita una frecuencia de PWM alta; en general se utilizan frecuencias entre 500Hz y 1kHz, por lo que se ha elegido utilizar un PWM generado por software, siendo más sencillo de programar en comparación del PWM por hardware, .

En Python, los puntos más importantes a considerar para generar un PWM son, la librería, el modo de enumeración y la frecuencia que se utiliza. Python al ser un lenguaje de acceso gratuito, permite la constante actualización de las funciones que contiene. Es vital conocer la librería para la versión del compilador y del sistema operativo de la Raspberry que se están utilizando.

La librería para la configuración del GPIO que se encuentra instalada dentro del repositorio de Raspberry es la *RPi.GPIO.* Esta libreria permite de manera sencilla definir una terminal del GPIO como entrada o salida, ofreciendo también la creación de una variable objeto de la terminal para generar un PWM por software.

Dentro de la librería, se necesita definir la terminal que generará el PWM, para lo cual se realizan los procesos siguientes. Python utiliza la sentencia *RPi.GPIO.setmode* para definir el criterio numérico que debe representar a la terminal; por lo tanto, se debe tomar en cuenta si utilizar el número de terminal designado por el procesador o el número bajo la nomenclatura de la tarjeta. En la figura 20, cada terminal del GPIO posee un número de identificación (GPIO 19, GPIO 21, etc.), a dicho identificador se le conoce como enumeración por BCM, siendo la salida directa del procesador ARM de la Raspberry Pi. Por otro lado, está la enumeración por BOARD, el cual se define como el número asignado por la posición que se tiene en los cabezales de las terminales.

Por último, se define la frecuencia a utilizar del PWM. Para un resorte de NiTiNOL, se utilizó la frecuencia a 500Hz para sus pruebas de operación. A continuación, se muestra el fragmento de programa en Python en donde se hace la configuración y prueba del PWM. En este código `res1` es una variable objeto para la creación del PWM, `s_left` es el puerto

por el cual se generará la señal y `res1.start(100)`es el porcentaje del ciclo de trabajo bajo el que comenzará a operar la señal en el sistema.

```
 s_left = 16 #se utiliza la terminal 16 como salida del PWM

 GPIO.setmode(GPIO.BOARD) # se utiliza la numeración por board

 GPIO.setup(s_left, GPIO.OUT) #se define la terminal 16 como sa-
lida

 res1 = GPIO.PWM(s_left,500) # se define al PWM en la terminal
16 con una frecuencia de 500Hz

 res1.start(100) #Se inicia la operación del PWM con el 100% del
ciclo de trabajo
```

4.2 PWM para el control de una Etapa de Potencia

Para los algoritmos de control desarrollados en sistemas digitales o analógicos que generan señales con muy baja potencia, es indispensable el uso de etapas de potencia para activar los actuadores de la planta, como motores, válvlulas y, en nuestro caso particular, pasar la corriente necesaria para generar calor en los resortes de Nitinol.

La etapa de potencia utilizada en esta tesis (figura 21), fue desarrollada por el estudiante Jorge Parra López de programa educativo de Ingeniero en Mecatrónica. Esta estapa de potencia está diseñada para controlar 4 resortes de Nitinol en forma simultánea e independiente. Las entradas se localizan en la parte superior de la tarjeta, con nombres de PWM1 hasta PWM 4. Las salidas están acomodadas verticalmente en orden descendente, es decir, que la primera salida S4 funciona con el PWM 4, la S3 con PWM 3 y así sucesivamente.

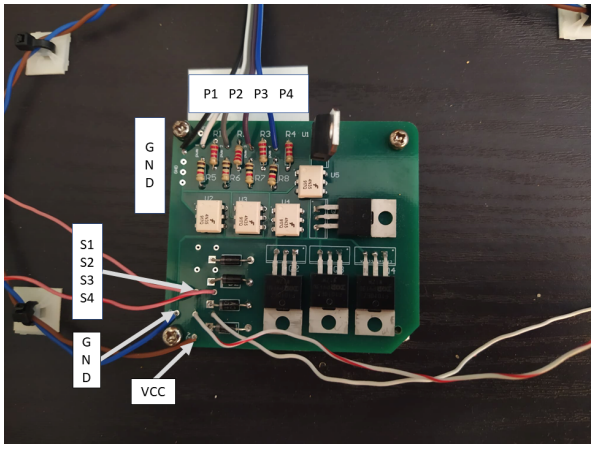

Figura 21. Etapa de potencia para el control de actuadores de Nitinol.

Su forma de operación es la siguiente, la etapa de potencia esta conectada a una fuente de alimentación directa a 12 V, donde, en base al ancho de pulso del PWM será la proporción de voltaje entregado por la etapa de potencia en la salida. El diseño de la tarjeta fue realizado de tal manera que opere por medio de un PWM inverso, esto es, para un ancho de pulso de 0 digital, la tarjeta entrega el 100 % de su capacidad (12 V), y para un ancho de pulso máximo de 1 digital se entrega un 0 %. Esta convención de la tarjeta puede ser modificada fácilmente dentro del programa que genera el PWM.

4.3 Código para operación de dos canales de PWM

El circuito de potencia del sistema, está diseñado para entregar un voltaje de salida en base a la proporción del PWM. Generando dos PWM por software, la Raspberry acciona los puertos P3 y P4 de la etapa de potencia, los cuales transfieren el voltaje a las salidas en los resortes izquierdo y derecho respectivamente. En Python, las configuraciones preliminares de las señales se realizan de la siguiente manera:

```
def Rasp_Config():

    s_left = 16

    s_right = 12

    GPIO.setwarnings(False)

    GPIO.setmode(GPIO.BOARD)

    GPIO.setup(s_left, GPIO.OUT)

    GPIO.setup(s_right, GPIO.OUT)

    global res1,res2
```

```
    res1 = GPIO.PWM(s_left,500)
    res1.start(100)
    res2 = GPIO.PWM(s_right,500)
    res2.start(100)
```

El ciclo de trabajo es controlado en función de la señal de control descrita en el capítulo 5, en el siguiente fragmento del código se muestra su operación, donde `u` hace referencia a la señal de control y las variables `dt1` y `dt2` al ciclo de trabajo de los PWM.

```
def act_control():
    if u>0:
        dt1 = (1 - (abs(u)/10))*100
        dt2 = 100
        res1.ChangeDutyCycle(dt1)
        res2.ChangeDutyCycle(dt2)
    else:
        dt1 = 100
        dt2 = (1 - (abs(u)/100))*100
        res1.ChangeDutyCycle(dt1)
        res2.ChangeDutyCycle(dt2)
```

Se probaron las configuraciones del código y el funcionamiento de la etapa de potencia, en el control de velocidad de un motor de corriente directa obteniendo resultados satisfactorios.

Capítulo 5

Implementación de algoritmos de control en Raspberry Pi 3 B+

En este capítulo se presenta la implementación del sistema de control en lazo cerrado que comprende un mecanismo rotacional de un grado de libertad con actuadores basados en resortes de Nitinol, un sensor de orientación TSS-MBT de la empresa Yost Labs, un driver de potencia basado en modulación por ancho de pulsos PWM y la microcomputadora Raspberry Pi 3 B+.

5.1 Descripción de la planta

La planta a controlar es un mecanismo rotacional de un grado de libertad, la cual se muestra en la figura 22, es importante mencionar que el mecanismo fue diseñado y construido por la alumna del programa de Ingeniero en Mecatrónica Lizeth Fernanda Moreno Rodríguez. El mecanismo consiste en un eslabón en forma de T con resortes de Nitinol sujetados de sus extremos a una base. Al instalar los resortes en la estructura estos se pueden estirar y contraer debido al movimiento angular generado por el eslabón [15]; pueden recuperar su forma original si se les excita aumentando su temperatura calentando los resortes o en este caso con el paso de una corriente eléctrica a través del material.

Al igual que en [16], para generar un aumento controlado de la temperatura en los resortes de Nitinol se hará circular una corriente eléctrica a través del resorte la cual, por el efecto Joule, generará calor. Para controlar la cantidad de corriente que circula por cada resorte se utiliza la modulación por ancho de pulsos (PWM) en conjunto con un circuito de potencia, el cual consiste de un transistor mosfet configurado como interruptor y un optoacoplador para

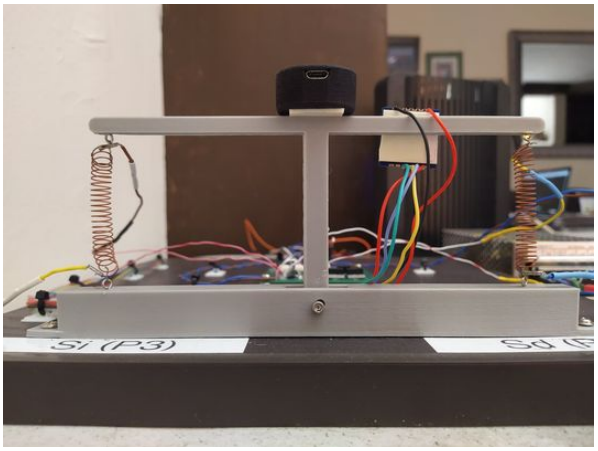

Figura 22. Mecanismo rotacional de un grado de libertad en forma de T con actuadores de Nitinol.

que se pueda conmutar la operación del interruptor con un voltaje de 3.3V, el cual es el nivel de salida de las señales digitales de la microcomputadora Raspberry Pi.

En la parte superior del eslabón, en el centro, se encuentra el sensor de orientación TSS-MBT el cual, debido a las características de comunicación inalámbrica y su batería interna, no necesita ninguna conexión con cables para su operación. En la medición de los ángulos se empleó el formato de los ángulos de Euler, en particular, el ángulo Roll, que corresponde a la rotación del sensor sobre un eje Z. En la figura 22 se muestra la ubicación que posee el sensor, que representa un ángulo muy cercano a cero, así como la condición inicial a utilizar como referencia. Tomándose como referencia la base del eslabón, la convención de signos que maneja el ángulo Roll, es la siguiente: una inclinación a la izquierda genera valores positivos del ángulo mientras que una inclinación a la derecha genera ángulos negativos.

El algoritmo de control, la captura de datos de orientación del sensor y la generación de las señales de control en el formato de PWM se llevan a cabo en la microcomputadora Raspberry Pi 3B+.

En la figura 23 se muestra el diagrama de conexiones y transferencia de datos respecto a cada etapa.

5.2 Código de control en Python

La estrategia de la programación que se llevó a cabo en esta tesis es hacer funciones de cada tarea con sus respectivos parámetros de entrada y variables de salida, las cuales seran llamadas hacia un programa principal en el momento que se necesiten. Siguiendo la estructura por funciones, a continuación se muestra la función que realiza las operaciones del controlador Proporcional Integral Derivativo (PID).

La estructura del código es la siguiente:

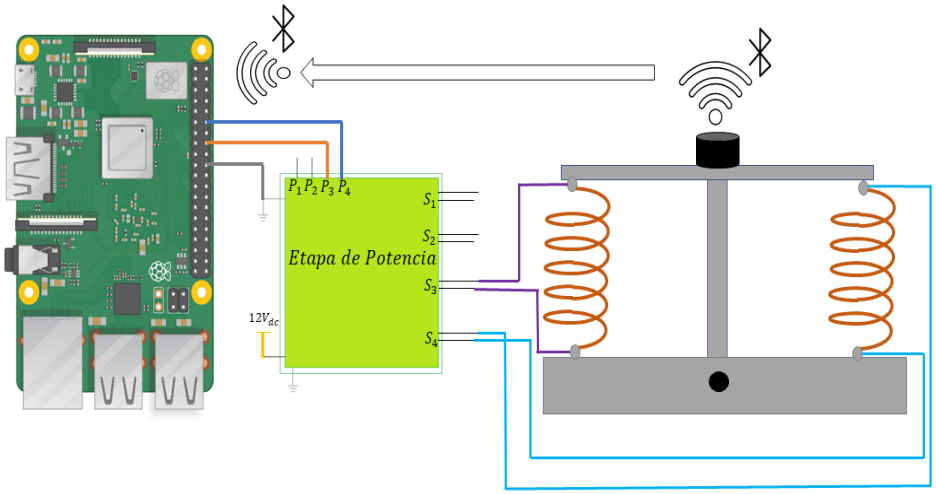

Figura 23. Representación de la transferencia de datos en función de las etapas del sistema.

```
 Kp = 15

 Ki = 1

 Kd = 0

 err_acum = 0

 def data_PID(err_acum, Kp, Ki, Kd):

     t2 = time.time()

     global tc

     tc = round(t2 - t1, 6)

     q = 0.015        #Referencia en Radianes

     err = q - y3   #Diferencia entre la referencia y la sali-
da

     P = err*Kp       #Control Proporcional

     err_acum += (err*tc)

     I = err_acum*Ki #Control Integral

     D = y3s*Kd          #Control Derivativo

     global u

     u = P+I-D           #Señal de control
```

```
u = min(max(u, -1), 1) #Limites de la Señal de control

CSV_Write()
```

El código comienza definiendo los valores de las ganancias del controlador e inizializando la variable `err_acum`, ya que ambos son parámetros que necesita la función. El siguiente renglón corresponde a la declaración de la función del controlador PID. Lo primero que se hace es obtener el tiempo real del sistema con la función `time.time()`, la cual retorna el número de segundos transcurridos desde el epoch (punto de tiempo de inicio en común; para sistemas Linux es desde el 1 de enero de 1970, 00:00:00 (UTC)) por lo que su diferencia con la variable `t1,` variable `tc`, se considera como el lapso de tiempo transcurrido entre las llamadas a la función y se considera como el tiempo de muestreo.

Posteriormente se define la referencia `q` así como su error entre la salida del sistema (variable `y3)`. Enseguida se declara la parte proporcional del controlador multiplicando el error por la ganancia K_p. La integral se resuelve utilizando el algoritmo de la suma de Riemann. En cada iteración, se multiplica el error por el tiempo de muestro, donde, el acumulado de dicha multiplicación representa la integral. La acción integral de la señal se expresa con la multiplicación del error acumulado respecto a una ganancia K_i. Para implementar la parte derivativa del controlador se multiplica la velocidad angular (la cual se obtiene directamente del sensor) con respecto a la ganancia K_d. Finalmente con la suma de la accion proporcional, la acción integral y la resta de la accón derivativa, se obtiene la señal de control.

Para disminuir el riesgo de dañar los resortes de Nitinol por sobrecalentamiento se agrega una saturación a la señal de control y se almacenan los datos que en un archivo CVS.

5.3 Respuesta de la planta a la aplicación del algoritmo

de control

Antes de verificar el desempeño del controlador en la planta se realizaron pruebas del comportamiento del sistena en lazo cerrado sin energizar la etapa de potencia. Las pruebas consisten en establecer una referencia diferente de cero y mover el mecanismo manualmente hasta que logre llegar a la referencia indicada. En este proceso se observan las mediciones de la posición y de la velocidad angular, así como de la señal de control. Estas pruebas se realizaron con los controladores P y PI con ganancias de $K_p = 15$ y $K_i = 0{,}5$.

El resultado del controlador P se muestra en la figura 24, esta figura muestra la posición angular , la velocidad angular y la señal de control, respectivamente. En el intervalo de tiempo de 0 a 12 segundos, el eslabón se mantiene estatíco, por lo que el controlador manda una señal constante. Posteriormente, en el siguiente intervalo (12 a 42 segundos) se manipula la estructura de tal forma que alcance la señal de referencia, como se puede observar en la señal de control, esta disminuye hasta alcanzar un valor proximo a cero. Una vez que la señal de control se mantega, se suelta el eslabón y este retornará su posición inicial en proporción a las ganancias del controlador.

Se realizó el mismo experimento para el control PI, con la finalidad de verificar el funcionamiento del integrador, los resultados se muestran en la figura 25. Al igual que en el caso anterior, se comprobó el correcto funcionamiento del algoritmo de control para su acción integral.

Como se observa, la señal de control disminuye en proporción mientras que la posición angular aumenta acercandose a la referencia (linea negra punteada). Estos resultados indican que los algoritmos de control operan de manera adecuada y estan listos para probar su funcionamiento en la planta. Nótese también, en la figura 25, que la respuesta de la señal de control es, en comparación con la figura 24, mucho más sensible al movimiento de la posición angular debido al uso de la variable integral en el controlador.

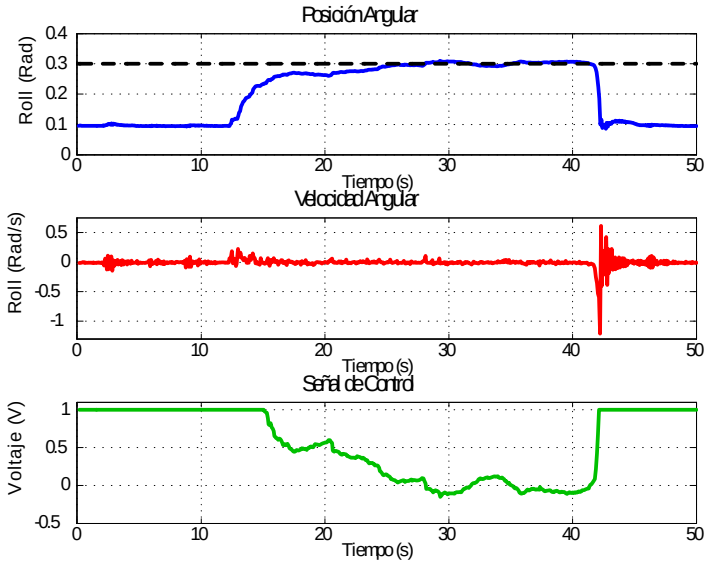

Figura 24. Comportamiento del algoritmo de Control Proporcional.

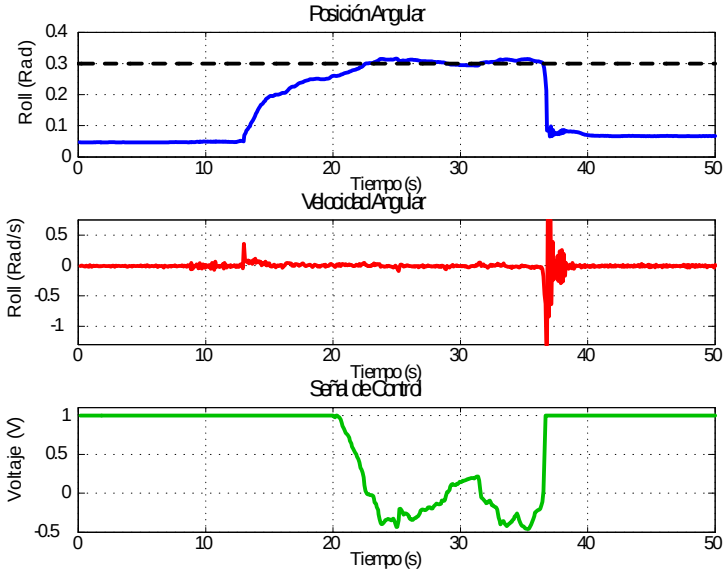

Figura 25. Comportamiento del algoritmo de Control Proporcional - Integral.

5.4 Desempeño de los controladores P y PI

En este apartado se muestran los resultados experimentales obtenidos de la aplicación del control Proporcional y Proporcional - Integral en la planta, es importante mencionar que no se incorpora la acción derivativa ya que la respuesta del controlador proporcional - integral fue suficiente para cumplir el objetivo de control.

En los experimentos se establecen dos referencias; $q = 0{,}07rad$, que provoca un movimiento a la izquierda, y $q = 0{,}015rad$, que gernera un movimiento a la derecha. Las ganancias que se utilizaron en los controladores son $K_p = 15$ y $K_i = 1$. Para cada referencia se realizaron 5 pruebas consecutivas, cuyos resultados se muestran en las figuras 26, 27, 28 y 29.

La posición angular del eslabón comienza en un valor alrededor de los $0{,}045rad$. Este valor nos ayuda a establecer una conveción en las referencias anteriores, siendo la referencia del resorte izquierdo la referencia positiva y la referencia del resorte derecho la referencia negativa.

La metodologia para el desarrollo de las pruebas de desempeño consistio en reproducir lo mas cercano posible los resultados del primer experimento. Su proceso consiste en los pasos siguientes:

1. Una vez revisado que todo en el sistema este encendido y conectado correctamente, se ejecuta el programa de control y se grafica la posición angular en tiempo real.
2. 12 segundos después se enciende la fuente conectada a la etapa de potencia; la estructura se comenzará a mover hasta lograr la referencia establecida.
3. Una vez alcanzada la referencia, se apaga la fuente y se refrigera la planta con el proposito de conseguir las condiciones iniciales lo más identicas posibles al experimento inicial.
4. Se repite el siguiente experimento desde el primer punto.

El tiempo de espera antes de encender y apagar la fuente fueron elegidos de manera arbitraria, siendo dependientes del comportamiento de la grafíca. A pesar de no haberse agre-

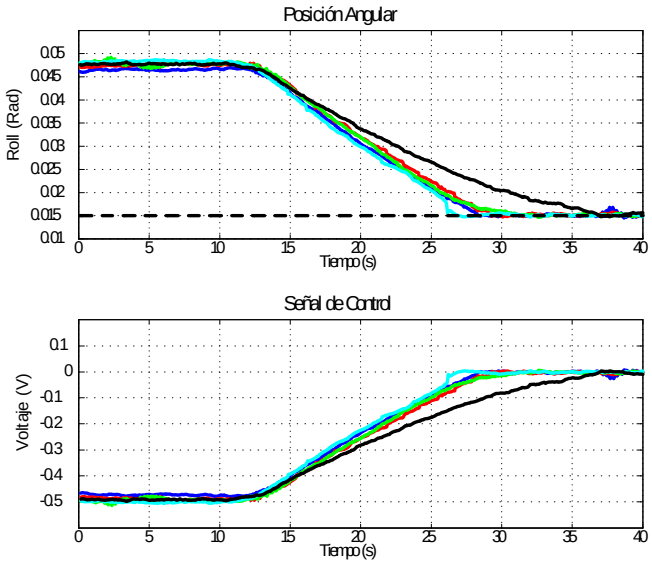

Figura 26. Posición angular y señal de control cuando el mecanismo se mueve a la derecha utilizando un control proporcional. Se muestran 5 experimentos.

gado en las gráficas implicitamente, el tiempo total que le toma subir y bajar al eslabón sin necesidad de refrigerarse es de un aproximado de 100 segundos para el controlador P y 80 segundos para el PI. Al no considerarse el calentamiento del resorte en este estudio se omitio dicha información. En las figuras 26 y 27 se presentan las gráfícas del comportamiento del controlador proporcional. Se pueden observar algunas variaciones entre los experimentos que se pueden atribuir a la no linealidad de los actuadores de Nitinol y a la falta de la medición de temperatura, siendo complejo de manejar unas condiciones iniciales similares entre cada experimento.

Los resultados experimentales utilizando el controlador PI se muestran en las figuras 28 y 29, los cuales fueron llevados a cabo de la misma forma que los experimentos con el controlador P. El tiempo que le toma llegar a la referencia en el controlador PI a diferencia del controlador P, es menor debido al aumento constante que se genera en la señal de control debido al integrador. Aunque el incremento de la señal de control ayuda en disminuir el tiempo que le toma llegar a la referencia, no es la forma optima de tratar con el algoritmo del

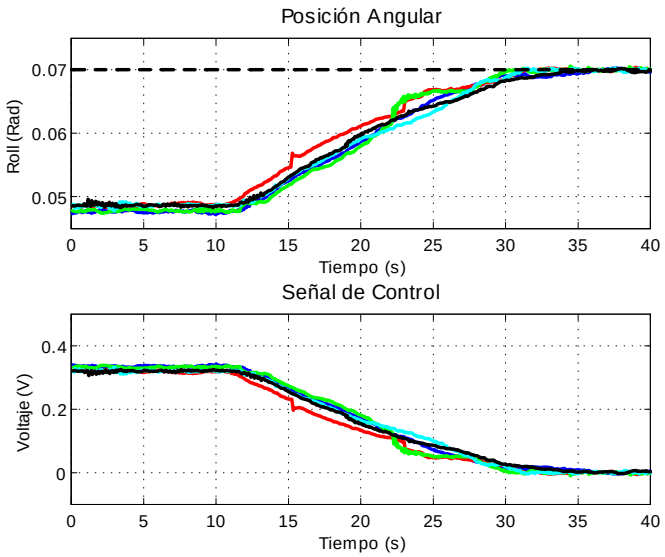

Figura 27. Posición angular y señal de control cuando el mecanismo se mueve a la izquierda utilizando un control proporcional. 5 experimentos.

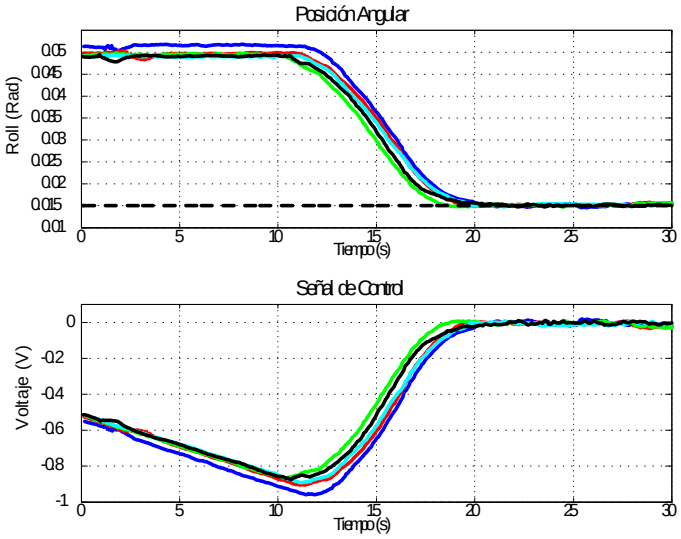

Figura 28. Posición angular y señal de control para el movimiento del actuador hacia la derecha utilizando un controlador PI. 5 experimentos.

integrador. Un incremento constante no controlado en la señal de control puede llegar a ser peligroso para la planta, logrando que el valor alcanzado sea enorme en comparación con la energía que los equipos puedan suministrar. Para prevenir daños en el sistema y/o equipo se le agregan limites a la señal de control, previniendo que el sistema suministre voltajes fuera de dicho rango. Los limites bajo los que trabaja la señal de control de este sistema son

$$-1 <= u <= 1.$$

Siendo el voltaje suministrado suficiente para su operación, dado que los resortes tienen la capacidad de operar a bajo voltaje.

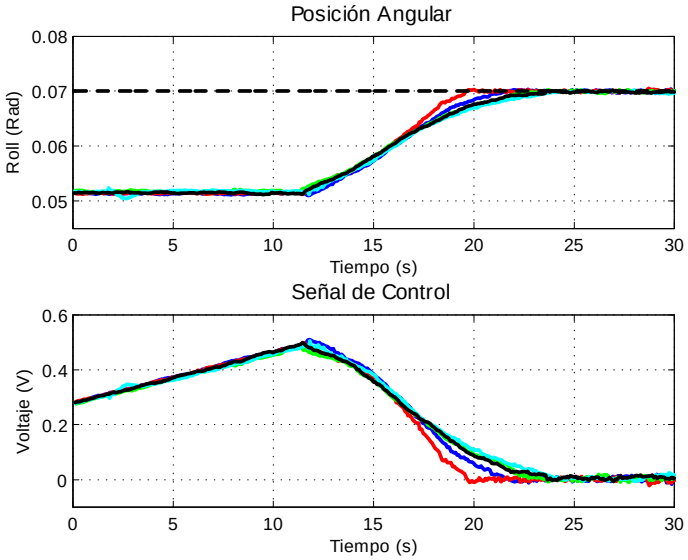

Figura 29. Posición angular y señal de control para el movimiento del mecanismo hacia la izquierda utilizando un controlador PI. 5 experimentos.

Capítulo 6

Conclusiones y trabajo futuro

En esta tesis se realizó el control de la posición angular de un mecanismo rotacional de un grado de libertad, el cual utiliza un par de resortes de Nitinol como actuadores y la microcomputadora Raspberry Pi para la implementación de los algoritmos de control, así como el uso de un sensor de orientación que se comunica en forma inalámbrica con el controlador. El trabajo esta dividio en tres etapas primordiales: instrumentación, potencia y la implementación implementación de algoritmos de control.

En lo que respecta a la instrumentación se puede concluir que el sensor 3 Space MBT es una buena opción para su uso en sistemas de control de naturaleza mecánica, ya que la comunicación es eficiente y rápida. Además se pueden obtener mediciones de la posición y velocidad angular en forma directa, lo cual evita el uso de derivadores en los controladores de tipo PID, y así evitar el uso de observadores de estado cuando se utilizan técnicas de control en base a un modelo en variables de estado. Debido a los algoritmos internos de filtrado que contiene el sensor, se minimiza la necesidad de un procesamiento adicional en cuestión del ruido en la plataforma en donde se implementan los algoritmos de control.

La etapa de potencia con la estrategia del control de corriente a través de la modulación por ancho de pulsos (PWM) es una adecuada para la activación de los actuadores basados en resortes de Nitinol. Sin embargo, es muy importante cuidar que la temperatura no exceda sus límites de operación, ya que los resortes pueden dañarse fácilmente si se sobrepasa la temperatura de umbral. Por otro lado, es muy importante implementar procedimientos de seguridad por software y por hardware para que los resortes no sobrepasen dicha temperatura. En este sentido, se considera que un trabajo futuro es la incorporación de sensores de temperatura y de estrategias para resolver estos problemas de sobrecalentamiento.

La microcomputadora Raspberry Pi es una plataforma que puede ser muy útil para la implementación de algoritmos de control en lazo cerrado, ya que que cuenta con el GPIO que le permite la conexión de diferentes sensores así como la generación de señales de control a través de PWM. Sin embargo, con el sistema operativo que se utilizó, no se puede garantizar una ejecución en tiempo real, lo que puede afectar a la estabilidad de sistemas en lazo cerrado que necesiten tiempos de muestreo pequeños o sistemas con mayor precisión. Un trabajo futuro es continuar con el estudio de diferentes sistemas operativos que permitan una ejecución en tiempo real de los algoritmos de control, así como, mejorar el control de procesos en la Raspberry Pi para realizar muestreos con sensores a tiempo real.

Como una conclusión global del trabajo se puede decir que la plataforma Raspberry Pi, junto con el compilador Python, es una buena opción para la implementación de sistemas de control en lazo cerrado en plantas con una dinámica lenta.

Referencias

[1] Nelli Fabio (2015). Python Data Analysis: Data Analysis and Science using Pandas, matplotlib, and the Python Programming Language. New York, United States: Apress Inc.

[2] Goodrich T. Michael, Tamassia Roberto, Goldwasser H. Michael (2013). Data Structures and Algorithms in Python. Hoboken, New Jearsey, United States: John Wiley & Sons Inc.

[3] Stanllings William (1997). Data and Computer Communications. New Jearsey, United States: Prentice Hall Inc.

[4] Barnes-Svarney Patricia (1995). The New York Public Library: Science Desk Reference. New York, United States: MACMILLIAN Inc.

[5] Maini K. Anil (2007). Digital Electronics: Principles, Devices and Applications. Chichester, West Sussex, England: John Wiley & Sons Ltd.

[6] Huang Albert S., Rudolph Larry (2007). Bluetooth: Essentials for programmers. New York, United States: Cambridge University Press Inc.

[7] Daqaq Mohammed F. (2019). Dynamics of Particles and Rigid Bodies - A Self-Learning Approach. New Jearsey, United States: ASME Press and John Wiley & Sons Ltd.

[8] Jazar Reza N. (2011). Advanced Dynamics: Rigid Body, Multibody, and Aerospace Applications. West Sussex, England: John Wiley & Sons, Inc.

[9] Spong Mark W., Hutchinson Seth, Vidyasagar M. (2020). Robot Modeling and Control. New York, United States: John Wiley & Sons, Inc.

[10] Katsuhiko Ogata. (1998). Ingeniería de Control Moderna 3ra Edición. Mexico: Pearson Education.

[11] William S. Levine. (2011). The Control Handbook: Control System Fundamentals, 2nd Edition. United States: CRC Press.

[12] Furber Steve (2000). ARM System-on-Chip Arquitecture. Boston, Massachusetts, United States: Addison-Wesley Professional.

[13] Greg Gagne, Peter Galvin, Abraham Silberschatz (2005). Operating System Concepts. Hoboken, New Jearsey, United States: John Wiley & Sons Inc.

[14] Dunlop, G.R. FEEDBACK CONTROL OF A NITINOL WIRE ACTUATOR. New Zeland: University of Canterbury, Department of Mechanical Engineering.

[15] Dunlop, Reg., Garcia Angelo C (2002). A Nitinol Wire Actuated Stewart Platform. New Zeland: University of Canterbury, Department of Mechanical Engineering.

[16] Sabancı, K , Koklu, M , Ekıncı, S . (2015). Determination of Nitinol Fibers Performances by Means of Embedded Systems . International Journal of Applied Mathematics Electronics and Computers.

[17] Raspberry Pi Corp. Raspberry Pi 3 Model B-Plus: Product Brief. Referencias Adicionales: www.raspberrypi.org/documentation/

[18] Yost Labs. 3-Space Sensor Mini Bluetooth: Miniature Attitude & Heading Reference System with Pedestrian Tracking User’s Manual. Portsmouth, Ohio, United States.

Anexo A. Código del programa en Python que realiza el control en lazo cerrado.

```
import csv
 import sys
 import time
 from math import pi
 import pandas as pd
 import RPi.GPIO as GPIO
 import matplotlib.pyplot as plt
 from matplotlib.animation import FuncAnimation
 from bluetooth import *
 {
    Kp = 15
    Ki = 1
    Kd = 0
```

```
    err_acum = 0
}
def BluetoothConfig():
    target_name = "YostLabsMBT"
    target_address = None
    nearby_devices = discover_devices()
    for bdaddr in nearby_devices:
        if target_name == lookup_name ( bdaddr ):
            break
    if target_address is not None:
        print("Found target bluetooth device with address ",
    target_address)
        time.sleep(1)
    else:
        print(“could not find target bluetooth device nearby”)
        time.sleep(1)
        sys.exit(0)
```

```
    service_matches = find_service(address = target_address)
    if len(service_matches) == 0:
        print(“couldn’t Find the Service”)
        time.sleep(1)
        sys.exit(0)
    first_match = service_matches[0]
    port = first_match["port"]
    name = first_match["name"]
    host = first_match["host"]
    print(“connecting to ”, target_name)
    time.sleep(1)
    global sock
    sock = BluetoothSocket( RFCOMM )
    sock.connect((host, port))
def CSV_Config():
    global fieldnames
```

```
    fieldnames = ["time Gyro", "time Euler Angles", ƶ3", ƶ3s",
"tc", ü"]

    with open (/home/pi/Documents/Data.csv", "w") as csv_file:

        csv_writer = csv.DictWriter(csv_file, fieldnames = field-
names)

        csv_writer.writeheader()

 def CSV_Write():

    with open (/home/pi/Documents/Data.csv", .ª") as csv_file:

        csv_writer = csv.DictWriter(csv_file, fieldnames = field-
names)

        info = {

            "time Gyro": tg,

            "time Euler Angles": t_e,

            ƶ3": y3,

            ƶ3s": y3s,

            "tc": tc,

            ü": u,

            }
```

```
        csv_writer.writerow(info)
def Rasp_Config():
    s_left = 16
    s_right = 12
    {
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(s_left, GPIO.OUT)
    GPIO.setup(s_right, GPIO.OUT)
    }
    global res1, res2
    res1 = GPIO.PWM (s_left, 500)
    res1.start (100)
    res2 = GPIO.PWM (s_right, 500)
    res2.start (100)
def s_data(i):
    data = pd.read_csv (/home/pi/Documents/Data.csv")
```

```
    t_e = data ["time Euler Angles"]

    tg = data ["time Gyro"]

    y3 = data [ƶ3"]

    y3s = data [ƶ3s"]

    u = data [ü"]

    tc = data ["tc"]

    {

    plt.cla()

    plt.plot (t_e, y3, label = Roll")

    plt.plot (tg, y3s, label = Roll Speed")

    plt.plot (tc, u, label = Çontrol Signal")

    plt.legend (loc=üpper left")

    plt.grid(True)

    }

def euler_angles():

    sock.send(":1\n")

    val =
```

```
while 1:
    data = sock.recv(1).decode()
    t2 = time.time()
    global t_e
    t_e = round(t2 - t1, 6)
    val = val + data
    if data == "\r":
        break
{
l = len(val)
y = val [0:l]
cf = y.find (",")
y1 = float (y [0:cf])
yk = y [cf+1:l]
cf2 = yk.find(",")
y2 = float (yk [0:cf2])
}
```

```
    yf = yk [cf2+1:l]
    cf3 = yf.find (",")
    global y3
    y3 = float (yf [0:cf3])
def gyro():
    sock.send(":38\n")
    sock.send(":48\n")
    val =
    while 1:
        data = sock.recv(1).decode()
        t2 = time.time()
        global tg
        tg = round (t2 - t1, 6)
        val = val + data
        if data == "\r":
            break
    {
```

```
    l = len(val)
    y = val [0:l]
    cf = y.find(",")
    y1s = float (y [0:cf])
    yk = y [cf+1:l]
    cf2 = yk.find (",")
    y2s = float (yk [0:cf2])
    }
    yf = yk [cf2+1:l]
    cf3 = yf.find ("\r")
    global y3s
    y3s = float (yf [0:cf3])
def data_PID (err_acum, Kp, Ki, Kd):
    t2 = time.time()
    global tc
    tc = round (t2 - t1, 6)
    q = 0.015
```

```
    err = q - y3
    P = err*Kp
    err_acum += (err*tc)
    I = err_acum*Ki
    D = y3s*Kd
    global u
    u = P+I+D
    u = min (max (u, -1),  -1)
    CSV_Write()
def act_control():
    if u>0:
        dt1 = (1 - (abs(u)/10)) * 100
        dt2 = 100
        res1.ChangeDutyCycle(dt1)
        res2.ChangeDutyCycle(dt2)
    else:
        dt1 = 100
```

```
        dt2 = (1 - (abs(u)/10)) * 100
        res1.ChangeDutyCycle(dt1)
        res2.ChangeDutyCycle(dt2)
{
    BluetoothConfig()
    CSV_Config()
    Rasp_Config()
}
data =
t1 = time.time()
try:
    print("Loading Data...")
    while True:
        euler_angles()
        gyro()
        data_PID(err_acum, Kp, Ki, Kd)
        act_control()
```

```
except KeyboardInterrupt:
    res1.ChangeDutyCycle(100)
    res2.ChangeDutyCycle(100)
    pass
{
    ani = FuncAnimation (plt.gcf(), s_data, interval = 1)
    plt.show()
    sock.close()
}
```

Anexo B. Tabla actualizada del código para caracteres ASCII

Caracteres ASCII de control

00	NULL	Carácter nulo
01	SOH	Inicio encabezado
02	STX	Inicio texto
03	ETX	Fin de texto
04	EOT	Fin transmisión
05	ENQ	Consulta
06	ACK	Reconocimiento
07	BEL	Timbre
08	BS	Retroceso
09	HT	Tab horizontal
10	LF	Nueva línea
11	VT	Tab horizontal
12	FF	Nueva página
13	CR	Retorno de carro
14	SO	Desplaza afuera
15	SI	Desplaza adentro
16	DLE	Esc. vínculo datos
17	DC1	Control disp. 1
18	DC2	Control disp. 2
19	DC3	Control disp. 3

20	DC4	Control disp. 4
21	NAK	Conf. negativa
22	SYN	Inactividad sínc
23	ETB	Fin bloque trans
24	CAN	Cancelar
25	EM	Fin del medio
26	SUB	Sustitución
27	ESC	Escape
28	FS	Sep. archivos
29	GS	Sep. grupos
30	RS	Sep. registros
31	US	Sep. unidades
127	DEL	Suprimir

Caracteres ASCII Básicos

32	ESP	48	0	64	@	80	P	96	‘
33	!	49	1	65	A	81	Q	97	a
34	"	50	2	66	B	82	R	98	b
35	#	51	3	67	C	83	S	99	c
36	$	52	4	68	D	84	T	100	d
37	%	53	5	69	E	85	U	101	e
38	&	54	6	70	F	86	V	102	f
39	’	55	7	71	G	87	W	103	g
40	(	56	8	72	H	88	X	104	h
41	)	57	9	73	I	89	Y	105	i
42	*	58	:	74	J	90	Z	106	j
43	+	59	;	75	K	91	[	107	k
44	,	60	<	76	L	92	\	108	l
45	-	61	=	77	M	93	]	109	m
46	.	62	>	78	N	94	^	110	n
47	/	63	?	79	O	95	_	111	o

112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~

Caracteres ASCII extendido

128	Ç	144	É	160	á	176	■	192	■	208	ð	224	Ó	240	-
129	ü	145	æ	161	í	177	■	193	■	209	Đ	225	ß	241	±
130	é	146	Æ	162	ó	178	■	194	■	210	Ê	226	Ô	242	=
131	â	147	ô	163	ú	179	■	195	■	211	Ë	227	Ò	243	$\frac{3}{4}$
132	ä	148	ö	164	ñ	180	■	196	■	212	È	228	õ	244	¶
133	à	149	ò	165	Ñ	181	Á	197	■	213	i	229	Õ	245	§
134	å	150	û	166	^{a}	182	Â	198	ã	214	Í	230	μ	246	÷
135	ç	151	ù	167	^{o}	183	À	199	Ã	215	Î	231	þ	247	¸
136	ê	152	ÿ	168	¿	184	©̲	200	■	216	Ï	232	Þ	248	°
137	ë	153	Ö	169	®̲	185	■	201	■	217	■	233	Ú	249	¨
138	è	154	Ü	170	¬	186	■	202	■	218	■	234	Û	250	·
139	ï	155	ø	171	$\frac{1}{2}$	187	■	203	■	219	■	235	Ù	251	^{1}
140	î	156	£	172	$\frac{1}{4}$	188	■	204	■	220	■	236	ý	252	^{3}
141	ì	157	Ø	173	¡	189	¢	205	■	221	|	237	Ý	253	^{2}
142	Ä	158	×	174	«	190	¥	206	■	222	Ì	238	¯	254	■
143	Å	159	■	175	»	191	■	207	■	223	■	239	´	255	nbsp