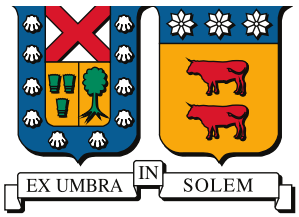

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE

"Sensorización de robot cuadrúpedo ArgoV2 para monitoreo de variables durante caminatas"

MEMORIA DE TITULACIÓN PARA OPTAR AL TITULO INGENIERO CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES

Francisco Miqueles Varela

Profesor Supervisor
Dra. María José Escobar

Co-Referente
Dr. Gonzalo Carvajal

Marzo de 2023

AGRADECIMIENTOS

QUISIERA agradecer a todas los alumnos, profesores y amigos que me acompañaron durante este proceso universitario brindandome apoyo, pues fueron un gran pilar para optar el título a ingeniero. Además, quiero mencionar a mi familia quienes incondicionalmente apoyaron mis trabajos y decisiones, en especial a mi hermano.

Me gustaría agradecer especialmente a un par de personas que cooperaron grandemente en este trabajo de memoria. Primero a Rodrigo Lanas, trabajador del AC3E quién brindó las herramientas necesarias para el desarrollo del proyecto y aportó con ideas para la soluciones de problemas de este mismo. A Pablo Reyes, alumno de magister quién fue un gran tutor y guía, permitiendo que el proyecto avanzara a pasos agigantados. Finalmente, a la profesora María José Escobar, con quién estoy sumamente agradecido por darme la oportunidad de trabajar en un proyecto desafiante y entretenido como lo fue este.

Francisco Miqueles Varela

RESUMEN

LOS robots con extremidades móviles han sido un tema recurrente de investigación debido a la importancia en la locomoción que pueden brindar, pues tienen la capacidad de moverse en terrenos accidentados y muy desestructurados. A diferencia de a un robot con ruedas cuyo movimiento está limitado por obstáculos a través de un trayecto continuo, la movilidad de un robot con extremidades solo está limitada por la disponibilidad de puntos de apoyo discontinuos (pisadas) que pueden distribuirse en todo el entorno. Esto les permite que sus sistemas sean capaces de superar obstáculos como ranuras, huecos, escalones, terreno resbaladizo, etc. Actualmente, son utilizados para el transporte de objetos, reconocimiento de terreno e incluso misiones de rescate. Además, en temas de investigación relacionados a la robótica bioinspirada e inteligencia artificial son muy estudiados, teniendo como objetivo final poder otorgar autonomía en sus movimientos y acciones.

Este trabajo tiene como objetivo principal el rediseño del ArgoV2, robot de 4 extremidades móviles con 3 grados de libertad cada una. El movimiento de las extremidades del robot va ligado a una serie de servo-motores Dynamixel, los cuales se comunican y controlan mediante dos conexiones físicas hacia un PC, una para la alimentación y otra para transmitir datos. Se investigó, estudió e implementó sensores y controladores que entregaran y manipularan información sobre la detección de pisadas y movimiento del robot. En la detección de pisadas, se implementó un sistema mecánico con un sensor de luz de bajo costo y un LED, obteniedo una señal continua representando la fuerza ejercida en el pie de la extremidad. La detección de movimiento fue con un acelerómetro giroscopio, obteniendo la aceleración y velocidad angular del robot. Finalmente todas las señales y data fueron controladas por un microcontrolador, el cual interactúa con un PC convencional.

Palabras Claves

Robots con extremidades móviles, sensores, caminata, ArgoV2, microcontrolador, Dynamixel.

ABSTRACT

Mobile legged robots have been a recurring topic of research due to their importance in locomotion, as they have the ability to move in rugged and highly unstructured terrains. Unlike a wheeled robot whose movement is limited by obstacles along a *continuous* path, the mobility of a legged robot is only limited by the availability of *discontinuous* foothold points (footsteps) that can be distributed throughout the environment. This allows their systems to overcome obstacles such as gaps, holes, steps, slippery terrain, etc. They are currently used for object transportation, terrain recognition, and even rescue missions. Additionally, they are widely studied in research related to bio-inspired robotics and artificial intelligence, with the ultimate goal of providing autonomy in their movements and actions.

This work aims to redesign the ArgoV2, a robot with 4 mobile legs, each with 3 degrees of freedom. The movement of the robot's legs is linked to a series of Dynamixel servo motors, which are communicated and controlled through two physical connections to a PC, one for power and the other for data transmission. Sensors and controllers were investigated, studied, and implemented to provide and manipulate information about footstep detection and robot movement. For footstep detection, a mechanical system with a low-cost light sensor and an LED was implemented, obtaining a continuous signal representing the force exerted on the leg's foot. Movement detection was done using an accelerometer gyroscope, obtaining the robot's acceleration and angular velocity. Finally, all signals and data were controlled by a microcontroller, which interacts with a conventional PC, adapting the connection with the Dynamixel motors.

Keywords

Mobile legged robot, sensors, walking capabilities, ArgoV2, microcontroller, Dynamixel.

ÍNDICE

AGRADECIMIENTOS i

RESUMEN ii

ABSTRACT iv

ÍNDICE DE FIGURAS viii

ÍNDICE DE TABLAS x

ABREVIACIONES xi

1. INTRODUCCIÓN **1**
1.1. Motivación y contexto 1
1.2. Definición del Problema 2
1.3. Estado del Arte 2

2. EL ROBOT ARGOV2 **7**
2.1. Estructura del robot 7
2.2. Motores Dynamixel AX-12A 9
2.3. Incorporación de motores y comunicación con el ArgoV2 . 10
2.4. Tabla de Control 12
2.5. Protocolo Dynamixel 1.0 16

3. SENSORIZACIÓN DEL ROBOT ARGOV2 **19**
3.1. Detección de Movimiento 19
3.1.1. MPU-6050 19
3.1.2. Comunicación I^2C 20
3.1.3. Registros y parámetros 22
3.1.4. Movimiento y orientación 22

3.1.5. Filtro de Kalman 23
3.1.6. Velocidad y distancia recorrida 25
3.1.7. Ángulo de inclinación 26
3.2. Detección de Pisadas 28
3.2.1. Implementación del pie flexible al ArgoV2 32
3.2.2. Nuevo sistema mecánico 32
3.2.3. Rediseño de pierna del ArgoV2 34
3.2.4. Base o Pedestal 34
3.2.5. Resorte 35
3.2.6. Varilla de Empuje 35
3.2.7. Seguro 36
3.2.8. Bota Flexible 37

4. CONTROL Y COMUNICACIÓN CON EL ROBOT 39
4.1. BlackPill - STM32F401CB 39
4.1.1. STM32CubeIDE y HAL 40
4.1.2. Direct Memory Access (DMA) 41
4.1.3. Conversor Bidireccional 3.3 - 5 [V] 41
4.2. Comunicación con el PC 42
4.2.1. Chip FT232 42
4.2.2. Protocolo ArgoV2 43

5. IMPLEMENTACIÓN DEL DISEÑO 45
5.1. Impresión y Construcción del Robot 45
5.2. BlackPill con STM32CubeIDe 46
5.2.1. Configuración, programación y depuración . . . 47
5.2.2. Código y Funciones 47
5.3. Comunicación con el PC y motores 48
5.3.1. Configuración de comunicación UART full-duplex y half-duplex 48
5.3.2. Codificación de toma de decisiones del controlador . 49
5.4. Inicialización, configuración y comunicación con el MPU-6050 50
5.5. Circuito y medición de detección de pisadas 51
5.6. Placa Perforada 51

6. PRUEBAS Y RESULTADOS OBTENIDOS 54
6.1. Movimiento y calibración de motores Dynamixel 54
6.2. Medición de gravedad 58
6.3. Medición de velocidad angular 60
6.4. Implementación del Filtro de Kalman 62

6.5. Calibración del ADC 66
6.6. Tensión en el pie flexible 67
6.7. Tiempos de Respuesta 68

7. CONCLUSIONES Y TRABAJO A FUTURO 72
7.1. Conclusiones 72
7.2. Trabajo Futuro 73

BIBLIOGRAFÍA 75

Índice de figuras

1.1. Robot ArgoV2. 2

2.1. Cuerpo sólido del robot ArgoV2. 7
2.2. Tapa del cuerpo sólido del ArgoV2. 7
2.3. Marco 1 del ArgoV2. 8
2.4. Marco 2 del ArgoV2. 8
2.5. Pierna del ArgoV2. 8
2.6. Bota de la pierna del ArgoV2. 8
2.7. Motor Dynamixel AX-12A. 9
2.8. Ubicación de los motores como articulaciones. 11
2.9. USB2DYNAMIXEL 12
2.10. Conexión para el acceso a la tabla de control. Imagen sacada de [13] 12

3.1. Componente MPU-6050. 20
3.2. Ejemplo de data enviada por I^2C. 21
3.3. Condición de inicio (izquierda) y de parada (derecha) en comunicación I^2C. 21
3.4. Diagrama de bloques del filtro Kalman. 25
3.5. Representacion de vectores para la obtención de ángulos de inclinación. Imagen elaborada por [19]. 26
3.6. Sensor de ultrasonido. 29
3.7. Sensor táctil FSR. 30
3.8. Modelo simplficado (izquierda) y detallado (derecha) del pie flexible. Imagen elaborada por [21] 31
3.9. Sensor de Luz LDR. 31
3.10. Sistema mecánico incluyendo un LDR y un LED. 32
3.11. Sección transversal del nuevo sistema mecánico. 33
3.12. Perfil de la nueva extemidad de la pierna. 34

3.13. Comparación entre la pierna antigua (rosado) y la nueva (morado). 34
3.14. Dimensiones y perfil de la base. 35
3.15. Dimensiones y perfil de la varilla de empuje. 36
3.16. Dimensiones y perfil del Seguro. 37
3.17. Dimensiones y perfil de la nueva bota flexible. 38
3.18. Dimensiones y perfil del molde. 38

4.1. Microcontrolador STM32F401. 40
4.2. Conversor Bidireccional. 42
4.3. Chip FT32 de FTDI. 42

5.1. Una de las nuevas piernas del ArgoV2. 46
5.2. STLink-v2 mini. 47
5.3. Diagrama de bloques de la toma de decisiones del controlador. 49
5.4. Circuito del LDR. 52
5.5. Circuito del LED. 52
5.6. Placa perforada PCB con todos los componentes por si sola y ubicada en el robot. 52
5.7. Modelo nuevo del ArgoV2. 53

6.1. Experimento para medir ángulo de los motores para las piernas 1 y 2. 56
6.2. Experimento para medir ángulo de los motores para las piernas 3 y 4. 57
6.3. Gráfico de aceleración en [mg] para el primer experimento. . 58
6.4. Aceleración manteniendo los ejes x e y fijos. 60
6.5. Velocidad angular durante el experimento de la silla para el eje x y z. 62
6.6. Velocidad angular durante el experimento de la silla para el eje y, y para el experimento sin movimiento. 63
6.7. Gráfico de comparación de aceleración medida y estimada. . 64
6.8. Distancia recorrida y velocidad durante el experimento. . . 65
6.9. Gráfico de comparación el ángulo de inclinación medido y estimado. 66
6.10. Gráfico de los cuatro canales del ADC sin sensores conectados. 67
6.11. Voltaje medido en la pierna 1 y 2. 69
6.12. Voltaje medido en la pierna 3 y 4. 70

Índice de tablas

2.1. Tabla de especificaciones del motor 10
2.2. Tabla de campos para el área EEPROM 14
2.3. Tabla de campos para el área RAM 15
2.4. Tabla de posibles instrucciones 17
2.5. Tabla de posibles errores 18
2.6. Instruction Packet 18
2.7. Status Packet 18

3.1. Paquete de datos I^2C 22

4.1. Tabla de posibles valores entregados por los sensores . . . 44

6.1. Tabla de ángulos para los motores de cada extremidad. . . 55
6.2. Tabla de estadísiticas del acelerómetro. 59
6.3. Tabla de estadísticas del giroscopio. 61
6.4. Tabla de estadísticas de los canales del ADC. 66
6.5. Tabla de las variaciones de voltaje de cada pierna del ArgoV2. 68
6.6. Tabla de tiempos de respuesta para cada motor/sensor . . 71

ABREVIACIONES

PC	: Personal Computer
CPU	: Central Processing Unit
LED	: Light Emiting Diode
LDR	: Ligth Dependent Resistor
UART	: Universal Asynchronous Receiver Transmitter
I^2C	: Inter-Integrated Circuit
HAL	: Hardware Abstraction Layer
APIs	: Application Programming Interfaces
HyperNEAT	: Hypercube-based NeuroEvolution of Augmenting Topologies
PLA	: Ácido Poliláctico
TPU	: Poliuretano termoplástico
DC	: Direct Current
V	: Voltage
USB	: Universal Serial Bus
TTL	: Transistor to Transistor Logic
RAM	: Random Access Memory
EEPROM	: Electrically Erasable Programmable Read-Only Memory
FSR	: Force Sensing Resistor
SPI	: Serial Peripheral Interface
DMA	: Direct Memory Access
FTDI	: Future Technology Devices International
Tx	: Transmitter
Rx	: Receiver
SWD	: Serial Wire Debug
HSE	: High-speed External
ADC	: Analog to Digital conversion
LSB	: Least Significant Bit

Capítulo 1

INTRODUCCIÓN

1.1. Motivación y contexto

Este proyecto se basa en el rediseño y construcción del ArgoV2, robot cuadrúpedo de extremidades móviles. El robot ArgoV2 fue creado por ingenieros en diseño de productos durante el año 2013 y se conforma de una serie de modelos 3D junto a servomotores de la marca Dynamixel.

Este robot ha sido estudiado y utilizado previamente por exalumnos de la carrera de Ingeniería Civil Electrónica, siendo sometido a entrenamientos para el aprendizaje de su caminata mediante la implementación de algoritmos neuroevolutivos. En particular, dentro de los trabajos relacionados al tema a tratar, podemos encontrar las memorias de los exalumnos Oscar Silva y Pablo Reyes [1] [2] quienes estudiaron, implementaron y mejoraron el algoritmo Hipercubo basado en el Aumento de Topologías (HyperNEAT) para generar caminatas para el robot ArgoV2. El algoritmo HyperNEAT permite la evolución de la topología y los pesos de una red neuronal tomando como base un algoritmo genético.

Ambos trabajos se separaban en dos partes principales: simular el entrenamiento del algoritmo neuroevolutivo y luego llevarlo a terreno mediante el uso de la plataforma robótica ArgoV2.

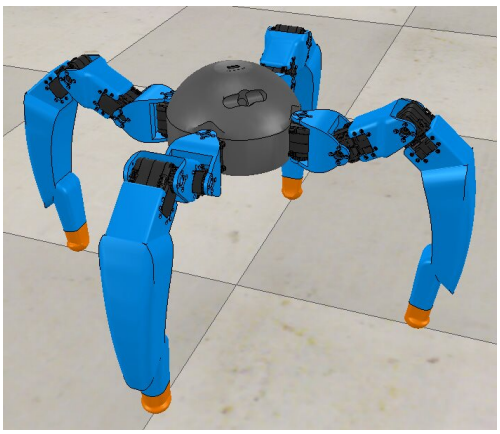

Figura 1.1: Robot ArgoV2.

1.2. Definición del Problema

Si bien ambos trabajos lograban la implementación y mejora del algoritmo en términos del aprendizaje de la caminata del robot, una vez las pruebas salieron de la simulación, los problemas ocurrían. Las imperfecciones del terreno, su rugosidad e inclinación entorpecían el movimiento de los motores, arruinando así el aprendizaje del robot y terminando en muchos casos en no lograr realizar una caminata. Por ende, el principal interés de este proyecto es sensorizar el ArgoV2 para que obtenga información del entorno y terreno en el que se encuentra. Así, se podría mejorar el aprendizaje del robot usando información extra en la entrada de la red o agregando lazos cerrados de control. Esto se realizó principalmente con una detección de pisadas mediante la inclusión de un sistema mecánico y una detección de movimiento mediante la obtención de la aceleración y rotación que experimenta el robot en todas direcciones. El sistema mecánico entrega una señal continua que representa la fuerza ejercida en la punta de cada extremidad del robot. Finalmente, se incluyó un controlador que incorporara la comunicación de los motores del robot y manejara la data sensada por la detección de pisadas y movimiento.

1.3. Estado del Arte

Respecto al estudio de sensores para su implementación, el autor Ahmed M. Almassri y su equipo [3], publicaron un trabajo en el que se habla del funcionamiento de los distintos métodos que se usan para medir la presión, es decir, obtener la distribución de una fuerza ejercida en un área específica, así como de tipos de sensores que transformen esta cantidad física en una señal

medible e interpretable por el usuario.

Celda de Carga: Las celda de carga son un tipo de transductor de presión muy utilizado en la industria debido a la gran variedad de posibles aplicaciones que poseen. Estos transforman una fuerza mecánica en una señal eléctrica, siendo el modelo más común el extensómetro. Existen muchos métodos distintos para la transducción de la fuerza en celdas de carga, así como presentarse en distintas formas y tamaños.

Lámina indicadora de presión: Este tipo de sensor se utiliza normalmente para medir la presión entre dos superficies diferentes. Su construcción consta de un material colorido cubierto en ambas caras por láminas de poliéster, además un grupo de microcápsulas diseñadas para romperse a distintos valores de presión. Cada una de estas microcápsulas contienen una pequeña cantidad de tinta, de modo que mientas más fuerte sea la fuerza aplicada al sensor se obtiene una imagen más oscura. La principal ventaja de este sensor radica en su simpleza ya que no usa componentes electrónicos.

Sensor táctil: Los sensores táctiles miden parámetros de un objeto mediante el contacto físico con este. Entre los parámetros que pueden medir se encuentran la presión, temperatura, vibración, torque, fuerza normal y de corte, etc.

Sensor piezoresistivo: Los sensores piezoresistivos miden la presión mediante la variación de resistencia eléctrica de un material piezoresistivo. Estos materiales tienen la característica de sufrir una deformación ante un estrés mecánico, lo que resulta en una variación de resistencia que puede ser detectado mediante una medición de voltaje. Este tipo de sensor destaca por su bajo costo, facilidad de construcción, durabilidad a largo plazo y efectividad en términos de sensado.

Sensor piezoeléctrico: Los sensores piezoeléctricos miden la presión ejercida mediante la diferencia de potencial producido en un material piezoeléctrico. Estos materiales (normalmente cristales) experimentan un fenómeno en el que, al ser sometidos por una tensión mecánica, su masa adquiere una polarización eléctrica, lo que resulta en una diferencia de potencial y cargas eléctricas en su superficie. Este tipo de sensor destaca en su facilidad de integración y se usan normalmente para mediciones dinámicas.

Sensor capacitivo: Los sensores capacitivos consisten en dos placas conductoras cuya distancia varía según la fuerza que se aplique al sensor, traduciendose en una variación de capacitancia. Son muy llamativos debido a su sensibilidad, robustez y poco consumo de energía.

Finalmente se realiza una comparación entre los sensores explicados exponiendo sus pros y contras para luego seleccionar el mejor para su implementación en una mano robótica mediante un algoritmo de control.

En esta misma línea, el investigador T. K. Maiti, de la Universidad de Hiroshima en Japón [4], desarrolló un sistema control para la caminata de un robot móvil humanoide mediante la detección de pisadas usando sensores piezoresistivos en sus extremidades. Primero, creó un modelo del sensor de presión en un entorno simulado para poder hacer simulaciones de la caminata del robot. Luego, comenzó con la integración de la información obtenida por el sensor piezoresistivo de cada extremidad a un micro-controlador Arduino Uno mediante un convertidor análogo-digital para finalmente lograr su caminata con un algoritmo de control.

Un enfoque diferente al problema es entregado por Csaba Kertész [5] quién creó y probó un algoritmo supervisado de aprendizaje de máquinas basado en *random forest* para el reconocimiento del suelo en un robot doméstico móvil AIBO, entrenandolo para seis distintas superficies. Los sensores usados son los siguientes.

Acelerómetro El acelerómetro entrega información sobre la aceleración que existe en un objeto a lo largo de coordenadas (x, y, z). Son muy importantes en los estudios de caminatas en robots móviles pues son cruciales para conseguir estabilidad.

Sensor infrarrojo El sensor infrarrojo emite, detecta y mide radiación infrarroja dentro de un área específica. Son utilizados para medir movimiento, proximidad y temperatura.

Junto a estos se tienen sensores de presión tanto en las articulaciones y pies del robot para detección de pisadas y ángulos de movimiento. Luego, usando la información de los sensores, construyeron un vector de características dentro de una ventana de tiempo arbitraria para entrenar nueve clasificadores. Además, agregaron información adicional para cada tipo de sensor, calculando y agregando al vector la transformada rápida de Fourier (FFT) para cada uno de ellos, tomando en la mayoría de casos los primeros armónicos. El resultado

fue un algoritmo capaz de detectar seis tipos de suelos distintos, mostrando una validación cruzada de un $96,2\,\%$ y una precisión del $94\,\%$.

Un trabajo similar es presentado por Douglas Vail et al [6] quienes, usando el mismo robot AIBO, tomaron solo los valores estadísticos de las mediciones de su acelerómetro (media, varianza, correlación entre x, y, z) y lograron clasificar tres superficies distintas con una precisión de $92,3\,\%$. También presentaron una estimación de la velocidad basada en **K-Nearest Neighbors** (KNN) resolviendo así la ineficiencia que presenta integrar la medición de aceleración (acumulación de ruido). Otro gran aporte fue entregado en [7] quienes usaron acelerómetros de bajo costo para ayudar la percepción y manipulación de una extremidad robótica. Esto fue realizado mediante la integración de tres acelerómetros simples en una extremidad de 6 grados de libertad, logrando estimar casi perfectamente los ángulos de cada articulación mediante solo la información que estos entregaban.

Mark A. Hoepflinger et al [8] presentaron un método para estimar las propiedades del terreno para mejorar la estabilidad y caminata de robot móviles mediante un algoritmo de aprendizaje de máquinas Adaboost. Aquí clasifican el terreno usando cuatro clasificadores débiles de distintos tipos de formas concavas y convexas de varias profundidades para construir un clasificador fuerte. Para ello tomaron una sola pierna de un robot cuadrúpedo y le integran 3 sensores de presión simples en su articulación para poder obtener la fuerza normal y tangencial del punto de contacto de la extremidad. La información entregada por los sensores fue posteriormete filtrada en un filtro pasa bajos y digitalizada para luego ser enviada a un microcontrolador (Microchip DSPIC33FJ) mediante una interfaz UART a un PC convencional. La medición de los tres sensores y la corriente medida en el motor de la extremidad se construye un *vector de características* para entrenar *clasificadores*. Finalmente el algoritmo clasificó con un gran porcentaje de precisión (sobre $90\,\%$) las cuatro superficies y estructuras del terreno.

Joshua Christie et al [9], tomaron un camino diferente en este ámbito, considerando todo aspecto de audio involucrado en caminatas de robots móviles. Estos afirman que se pueden extraer importantes datos de clasificación con el sonido producido por el trayecto de los pies a través del piso, del aire y de sus pisadas. Esta información es capturada con un micrófono integrado en un robot cuadrúpedo y procesada en un algoritmo de aprendizaje **Support Vector Machine** (SVM) para obtener un clasificador de terreno en tiempo real. El robot utiliza 18 motores Dynamixel (3 por extremidad) y una cámara

Point Grey Chameleon; esta última solo para corroborar que el funcionamiento del algoritmo sea correcto. La captura de audio es realizada en el software abierto HARK-ROS y probado offline en MATLAB. El audio fue enventanado con una ventana Hanning y procesado con la FFT para luego pasar por una eliminación de ruido. Finalmente, se toman más de 20 características de la FFT para la construcción del *vector de características*. El resultado fue un algoritmo capaz de clasificar con un 92 % de precisión un total de 7 distintas superficies en tiempo real.

Si bien existen muchos métodos y sensores distintos para poder caracterizar el terreno en el que se encuentra un robot, Yingtian Xu [10] construyó un sensor multimodal para generalizar esta tarea. Este particular sensor es construído mediante una combinación de un sensor piezoresistivo y uno piezoeléctrico, además de utilizar un efecto triboeléctrico para la adquisición de información. Luego, esta información es procesada por sistemas de control, siendo capaz de caracterizar completamente el terreno entregando información sobre la textura, dureza y rugosidad del suelo. Además, probaron el sensor multimodal en un experimento real con un robot móvil humanoide, obteniendo buenos resultados respecto al sensado.

Capítulo 2

EL ROBOT ARGOV2

2.1. Estructura del robot

El ArgoV2 es un robot cuadrúpedo de extremidades móviles el cual fue diseñado por ingenieros en diseño de productos durante el año 2013. La estructura del robot esta diseñada en una serie de modelos 3D, separándose en dos partes principales. La primera es el cuerpo o centro del robot, es la parte más pesada de este y es la pieza que conecta las cuatro extremidades móviles. Consta de dos partes, un cuerpo sólido y una tapa tipo cúpula, como se puede observar en las figuras 2.1 y 2.2. Estas piezas se utilizan para ubicar circuitería usada para la comunicación con el robot, las cuales se entrarán más a detalles en la siguiente sección.

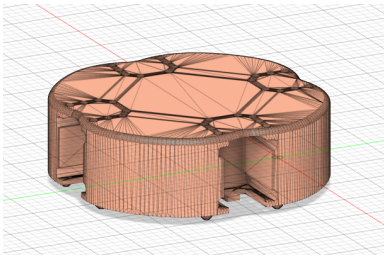

Figura 2.1: Cuerpo sólido del robot ArgoV2.

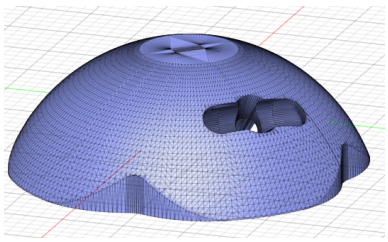

Figura 2.2: Tapa del cuerpo sólido del ArgoV2.

La segunda parte del robot sería el modelo de las extremidades. Cada extremidad se conforma por cuatro piezas distintas: una pierna de estructura robusta (Figura 2.5), dos marcos que interconectan la pierna con el cuerpo

del robot (Figura 2.3 y 2.4) y finalmente una bota flexible que se encarga del contacto con el piso (Figura 2.6). La pierna y los marcos se conectan entre sí hacia el cuerpo del robot mediante el posicionamiento de tres motores Dynamixel. Estos motores otorgan toda la movilidad al robot, entregando tres grados de libertad a cada pierna, moviéndose dos verticalmente y uno horizontalmente.

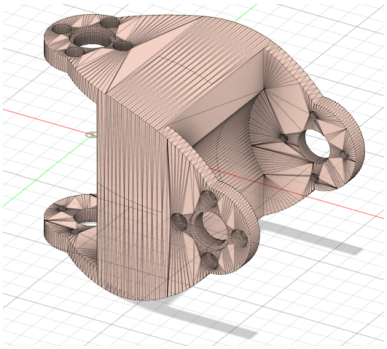

Figura 2.3: Marco 1 del ArgoV2.

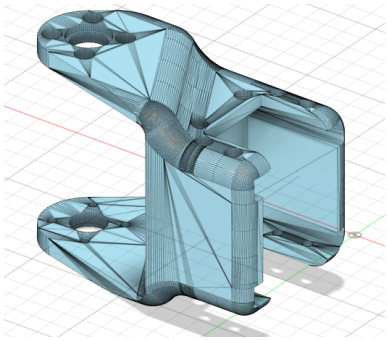

Figura 2.4: Marco 2 del ArgoV2.

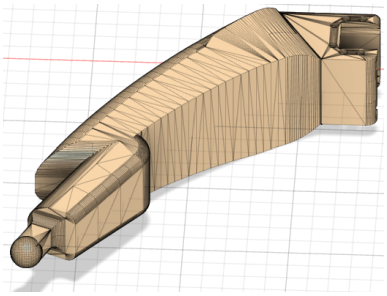

Figura 2.5: Pierna del ArgoV2.

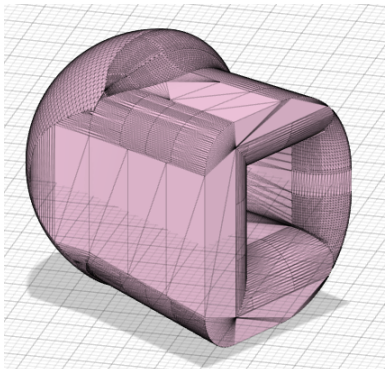

Figura 2.6: Bota de la pierna del ArgoV2.

Las piezas son impresas en filamento PLA (ácido poliláctico) en su gran mayoría, exceptuando por la bota que es impresa en filamento TPU (poliuretano termoplástico) debido a que este tipo de material presenta mayor flexibilidad. Para el ensamble de las piezas y motores se usan pernos y tornillos, exceptuando la bota que va a presión en la extremidad del robot.

2.2. Motores Dynamixel AX-12A

El movimiento principal del robot va ligado a una serie de servo-motores inteligentes de la marca Dynamixel, en específico, los Dynamixel AX-12A [11]. Estos motores tienen una dimensión de 32 x 50 x 40 [mm], un peso de 54.6 [g] y funcionan con una alimentación de entre 9 - 12 [V] DC. Cuentan con una conexión de tres pines, uno para transmitir data y dos para la alimentación. Además, cuentan con una variada opción de baudrates para su comunicación

Figura 2.7: Motor Dynamixel AX-12A.

hacia el controlador, la cual será explicada a más detalle en la siguiente sección. Más información de las especificaciones del motor se pueden ver en la tabla 2.1. Este modelo de motor funciona mediante el acceso, ya sea en escritura o lectura, de una tabla de control de registros. Aquí es donde se puede tanto obtener como editar distintas caracterísiticas del motor como pueden ser el torque, limitar ángulos, cambiar la velocidad de conexión, etc.

Tabla 2.1: Tabla de especificaciones del motor

Item	Specifications
Baud Rate	7,843[bps] 1[Mbps]
Weight	AX-12 (53.5 [g]), AX-12+ (53.5 [g]), AX-12A (54.6 [g])
Dimensions	32 X 50 X 40 [mm] 1.26 X 1.97 X 1.57 [inch]
Resolution	0.23[°]
Running Degree	0 300[°] Endless Turn
Motor	Cored
Gear Ratio	254:1
Stall Torque	1.5[N.m](at 12[V], 1.5[A])
No Load Speed	59[rev/min](at 12[V])
Operating Temperature	-5 +70[°C]
Input Voltage	9.0 12.0 [V] **(Recommended: 11.1V)**
Command Signal Physical Connection	Digital Packet TTL Level Multi Drop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	254 ID (0 253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Gear Material	Engineering Plastic (Full)
Case Material	Engineering Plastic (Front, Middle, Back)

2.3. Incorporación de motores y comunicación con el ArgoV2

Los motores Dynamixel se ubican entre las pieza 3D que conforman cada extremidad. Específicamente, se ubican en las extremidades de las piezas pierna, marco 1 y marco 2, funcionando así como las articulaciones del robot. Cada una de estas piezas posee espacio para la ubicación de tornillos que mantienen a los motores firmes para el movimiento de las articulaciones. Además, para evitar que la pierna haga movimientos imposibles (debido a que las piezas se bloqueen entre sí) los 3 motores de cada extremidad son limitados a ciertos ángulos. En específico, el motor más cercano al centro del robot se mueve entre 105 y 195 grados, el motor entremedio entre 80 y 220 grados y el motor más cercano a la bota entre 70 y 230 grados. Estos motores son alimentados mediante una fuente externa de 12[V], la cual define la primera

conexión física para que el robot pueda caminar. Esta fuente es conectada a un SMPS2Dynamixel [12], componente que alimenta de forma estable los motores.

La comunicación del robot se realiza mediante un adaptador USB2DYNAMIXEL

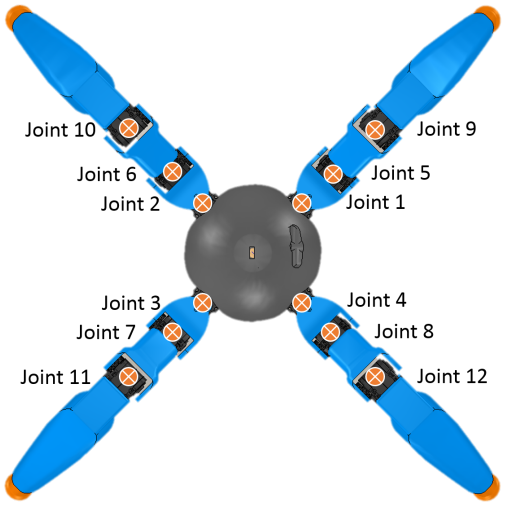

Figura 2.8: Ubicación de los motores como articulaciones.

[13]. El adaptador permite la comunicación entre motor y PC transformando señales analógicas a señales digitales usando una conexión USB. El proceso transforma los datos seriales enviados por TTL (Transistor to Transistor Logic) a protocolo USB y viceversa. Esto establece la segunda conexión física necesaria para el funcionamiento del robot, por donde se modifican y mueven los motores Dynamixel para empezar su caminata. El controlador que maneje la data de los sensores diseñados para la detección de movimiento y pisadas tendrá que reemplazar esta comunicación con los motores, de modo que el funcionamiento actual del robot se mantenga intacto. La data enviada por el protocolo hacia los motores debe estar al menos a una tensión de 5[V]. Típicamente los controladores usados en la industria se alimentan con rangos bajos de tensión (aproximadamente 2-3.6 [V]) y por ende, se requerirá de un circuito que eleve la tensión de las señales para el funcionamiento correcto de los motores.

Figura 2.9: USB2DYNAMIXEL

2.4. Tabla de Control

La tabla de control es una estructura que consiste en varios campos donde se encuentran registros con data, ya sea para guardar el estado de alguna característica del motor o realizar control de este mismo. El usuario es capaz de manipular estos campos para chequear el estado de los motores mediante la **lectura** de data enviando paquetes de instrucciones. Análogamente, mediante la **escritura** de data el usuario es capaz de controlar el motor. En ambos casos, la **dirección** del registro a acceder definirá que característica que se quiere manipular.

La tabla de control se divide en dos áreas específicas, RAM (Random Ac-

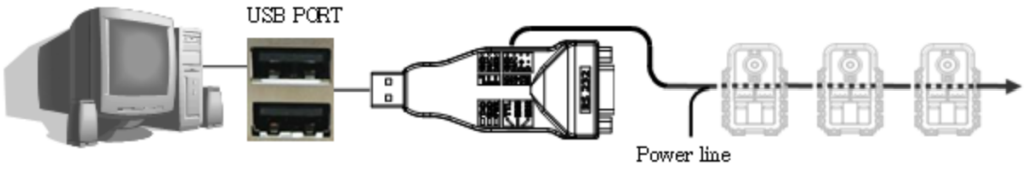

Figura 2.10: Conexión para el acceso a la tabla de control. Imagen sacada de [13]

cess Memory) y EEPROM (Electrically Erasable Programmable Read-Only Memory). El área de la RAM es reseteada a sus valores iniciales cada vez que el motor es encendido, es decir, es volátil. En cambio, el área de EEPROM mantiene sus valores aunque el dispositivo sea apagado (no-volátil).

Cada campo de la tabla de control consta de distintas propiedades que hay que tener en consideración al momento de acceder al registro. El primero es el **tamaño** del dato, que normalmente varía entre 1 - 2 bytes. En caso de que el

largo sea mayor a 2, los bytes son guardados usando Little Endian. La segunda propiedad es el **acceso**. Aquí pueden haber dos opciones, RW y R, definiendo si el registro es de lectura y escritura o de solo lectura. El primero de estos es para el control del motor y el segundo se usa para monitoreo. Finalmente, la última propiedad es el **valor inicial**, definiendo los valores nominales de fábrica para cada campo. Si esta propiedad es modificada en el área EEPROM los valores modificados se mantendrán posterior al reseteo. En cambio, los valores iniciales del área RAM son restaurados posterior al reseteo. En las tablas 2.2 y 2.3 se pueden observar el nombre de todos los campos para ambas áreas junto a sus propiedades.

Si bien todos los campos de ambas áreas son importantes para el manejo y control del motor, nuestro interés va ligado a construir una comunicación entre este y un controlador. Por ende, nuestro enfoque es entender los campos de **ID**, **Baud Rate** y **Return Delay Time**. Respecto al ID, este corresponde a la identificación de cada motor. Este debe ser editado mediante el acceso al registro, pues los motores vienen de fabrica con un ID = 1. Como se verá en la siguiente sección, los paquetes de instrucciones deben especificar el ID del motor que se quiera controlar, permitiendo que exista una conexión en paralelo entre motores, así entregando solo un paquete en común para todos los dispositivos. El Baud Rate definirá cual es la velocidad de transmisión entre la conexión con el controlador. De fábrica los motores se comunican a una velocidad de 1 [Mbps]. Finalmente, el Return Delay Time especifica el tiempo que tomará para que el motor envíe un paquete de status al controlador. Este valor es importante de conocer, ya que así podemos saber cual será el tiempo que habrá entre el envío y recepción de un paquete.

Tabla 2.2: Tabla de campos para el área EEPROM

Address	Size(Byte)	Data Name	Description	Access	Initial Value
0	2	Model Number	Model Number	R	12
2	1	Firmware Version	Firmware Version	R	-
3	1	ID	Dynamixel ID	RW	1
4	1	Baud Rate	Comunication Speed	RW	1
5	1	Return Delay Time	Response Delay Time	RW	250
6	2	CW Angle Limit	Clockwise Angle Limit	RW	0
8	2	CCW Angle Limit	Counter-Clockwise Angle Limit	RW	1023
11	1	Temperature Limit	Maximum Internal Temperature Limit	RW	70
12	1	Min Voltage Limit	Minimum Input Voltage Limit	RW	60
13	1	Max Voltage Limit	Maximum Input Voltage Limit	RW	140
14	2	Max Torque	Maximum Torque	RW	1023
16	1	Status Return Level	Select Types of Status Return	RW	2
17	1	Alarm LED	LED for alarm	RW	36
18	1	Shutdown	Shutdown Error Information	RW	36

Tabla 2.3: Tabla de campos para el área RAM

Address	Size(Byte)	Data Name	Description	Access	Initial Value
24	1	Torque Enable	Motor Toque On/Off	RW	0
25	1	LED	Status LED On/Off	RW	0
26	1	CW Compliance Margin	CW Compliance Margin	RW	1
27	1	CCW Compliance Margin	CCW Compliance Margin	RW	1
28	1	CW Compliance Slope	CW Compliance Slope	RW	32
29	1	CCW Compliance Slope	CCW Compliance Slope	RW	32
30	2	Goal Position	Target Postion	RW	-
32	2	Moving Speed	Moving Speed	RW	-
34	2	Toque Limit	Toque Limit	RW	Max Torque
36	2	Present Position	Present Position	R	-
38	2	Present Speed	Present Speed	R	-
40	2	Present Load	Present Load	R	-
42	1	Present Voltage	Present Voltage	R	-
43	1	Present Temperature	Present Temperature	R	-
44	1	Registered	If Instruction is registered	R	0
46	1	Moving	Movement Status	R	0
47	1	Lock	Locking EEPROM	RW	0
48	2	Punch	Minimun Current Threshhold	RW	32

2.5. Protocolo Dynamixel 1.0

Como fue mencionado antes, los servo-motores inteligentes Dynamixel usan un protocolo para su comunicación y funcionamiento. Estos motores usan el Dynamixel Protocol 1.0, protocolo asincrónico serial de 8 bit, 1 bit de stop y ninguno de paridad [14]. Esta comunicación es del tipo **Half-duplex UART**, es decir, el transmisor (Tx) y el receptor (Rx) utilizan la misma conexión física y no pueden enviar información al mismo tiempo. La comunicación se basa en la transmisión de data en paquetes de bytes, de modo que se envía un paquete de instrucciones al motor y este una vez lo recibe transmite un paquete de status de vuelta. El paquete de bytes viene formado por los siguientes campos

Header El header es el campo que indica el inicio del paquete. Es formado por dos bytes distintos con el valor de 255 (0xFF).

ID Aquí se señala el ID del dispositivo que debe recibir la instrucción. Es conformado por un byte, donde la dirección 254 (0xFE) es para el broadcast.

Length Este campo indica el largo en bytes de las instrucciones/error, los parámetros y del *checksum*.

Instruction or Error Si el paquete es de instrucción, aquí se indica la instrucción que debe ejecutar el dispositivo. En caso de ser un paquete de status, se indica si hubo algun error en cualquiera de los campos.

Parameters Los parámetros son el data auxiliar de las instrucciones. Su uso depende de la instrucción dada.

Checksum Es usado para revisar si el paquete es dañado durante la comunicación. Se calcula como la negación de la suma de todos los campos anteriores.

Con el uso de este protocolo se puede describir y manipular el movimiento de los motores a gusto, donde la instrucción define si queremos obtener o cambiar información de este (como pueden ser el torque ejercido, velocidad de movimiento, etc) como también realizar una acción (moverse a cierto ángulo, etc). Además, el envío de estos paquetes permite una conexión en paralelo de múltiples dispositivos, de modo que un único paquete es envíado a todos los motores donde solo el del ID indicado ejecutará la instrucción mandada. La serie de instrucciones posibles se puede ver en la tabla 2.4, donde se muestra que valor de bytes debe tomar el campo de instrucciones así como una breve descripción de la instrucción a realizar.

Tabla 2.4: Tabla de posibles instrucciones

Value	Instructions	Description
0x01	Ping	Instruction that checks whether the Packet has arrived to a device with the same ID as Packet ID
0x02	Read	Instruction to read data from the Device
0x03	Write	Instruction to write data from the Device
0x04	Reg Write	Instruction that registers the Instruction Packet to a standby status; Packet is later executed through the Action instruction
0x05	Action	Instruction that executes the Packet that was registered beforehand using Reg Write
0x06	Factory Reset	Instruction that resets the Control Table to its initial factory default settings
0x08	Reboot	Instruction that reboots DYNAMIXEL (See supported products in the description)
0x83	Sync Write	For multiple devices, Instruction to write data on the same Address with the same length at once
0x92	Bulk Read	For multiple devices, Instruction to write data on different Addresses with different lengths at once (See supported products in the description)

Análogamente existe una tabla de errores posibles, que será la información que nos entregará cada motor una vez haya recibido un paquete de instrucción. A diferencia de este último, el byte del campo de error del paquete de status setea cada bit en uno o cero indicando los posibles errores que hayan ocurrido. Por tanto, se debe desglozar los bytes de este campo para identificar si algún error ocurrió. Los posibles errores por bit se pueden ver en la figura 2.5.
El protocolo Dynamixel 1.0 cuenta con librerías pre-hechas en Github [15], la que presenta funciones para la comunicación con los motores a través de lenguajes de programación como C, C++, Python, etc. En las figuras 2.6 y 2.7 se puede observar un ejemplo de la estructura de los paquetes de bytes que comunican los motores con el controlador.

Entonces, para comunicarse con los servo-motores es necesario construir un paquete de bytes y trasmitirlo con el protocolo UART mediante una conexión serial. Esto impone condiciones de diseño al proyecto, ya que el microcontrolador seleccionado debe ser por lo menos capaz de transmitir Half-duplex UART por alguno de sus periféricos.

Tabla 2.5: Tabla de posibles errores

Bit	Error	Description
Bit 7	0	
Bit 6	Instruction Error	In case of sending an undefined instruction or delivering the action instruction without the Reg Write instruction, it is set as 1
Bit 5	Overload Error	When the current load cannot be controlled by the set Torque, it is set as 1
Bit 4	Checksum Error	When the Checksum of the transmitted Instruction Packet is incorrect, it is set as 1
Bit 3	Range Error	When an instruction is out of the range for use, it is set as 1
Bit 2	Overheating Error	When internal temperature of DYNAMIXEL is out of the range of operating temperature set in the Control table, it is set as 1
Bit 1	Angle Limit Error	When Goal Position is written out of the range from CW Angle Limit to CCW Angle Limit , it is set as 1
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control table, it is as 1

Tabla 2.6: Instruction Packet

Header1	Header2	PacketID	Length	Instruction	Params	Checksum
0xFF	0xFF	ID	Length	Instruction	Params	CHKSUM

Tabla 2.7: Status Packet

Header1	Header2	PacketID	Length	Error	Params	Checksum
0xFF	0xFF	ID	Length	Error	Params	CHKSUM

Capítulo 3

SENSORIZACIÓN DEL ROBOT ARGOV2

3.1. Detección de Movimiento

Gran parte de sensorizar el robot ArgoV2 es poder detectar cuando , como y hacia donde se esta moviendo durante su caminata. De esta tarea estará encargado un **acelerómetro**, componente que permite medir la aceleración de los ejes x, y, z. Con esta información se puede mantener la estabilidad de la caminata del robot, evitando que este tambalee o que incluso se caiga. Un acelerómetro puede detectar aceleraciones no deseadas y proporcionar datos que se pueden utilizar para corregir la posición y la velocidad del robot. Hay muchas razones que pueden producir estas aceleraciones no deseadas, algunas serían la inclinación del terreno, la fricción y la presencia obstáculos. Otra característica que presenta el acelerómetro es detectar la magnitud y dirección de la fuerza de gravedad a la que esta sometido el robot. Además, como se menciono en el estado de arte, la detección de la gravedad es de gran utilidad para hacer control con la caminata del robot, obtener el centro de gravedad e incluso detectar la superficie del terreno.

3.1.1. MPU-6050

Para este proyecto se usará el acelerómetro giroscopio MPU-6050 [16]. El MPU-6050 es un sensor de movimiento que posee un acelerómetro y un giroscopio en los tres ejes (x, y, z) con una alta precisión y tiene dimensiones de

4 x 4 x 0.9 [mm]. El acelerómetro mide la aceleración lineal a lo largo de los ejes en g (gravedad). En cambio, el giroscopio mide la velocidad de angular en cada eje en grados por segundo [°/s]. La combinación de ambas datas permiten describir precisamente el movimiento y orientación del robot. Además, el componente tiene un sensor de temperatura. Debido a todas estas características este componente es muy usado en la industria, pues es de poco consumo, pequeño tamaño y muy accequible monetariamente.

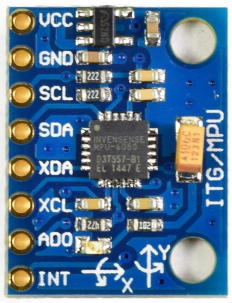

Figura 3.1: Componente MPU-6050.

3.1.2. Comunicación I^2C

El dispositivo presenta una resolución de 16 bits para sus ADC (Analog to Digital Conversion) internos tanto para el acelerómetro como para el giroscopio y se comunica a través del protocolo de comunicación serial I^2C (Inter-Integrated Circuit) a una velocidad de 400 [kHz]. Esta comunicación usa la arquitectura **maestro-esclavo**, donde el primero de estos es el que coordina y inicia la comunicación y el segundo está a la espera de ordenes para enviar o recibir data. Para el uso de este protocolo se debe disponer de dos conexiones físicas para la comunicación, una para transmitir data (SDA) y otra para sincronización serial (SCL). El componente trabaja como slave, por lo que típicamente se usa un microcontrolador que trabaja como master. La alimentación varía entre $2{,}375V - 3{,}46V$.

La manera de coordinar la data enviada o recibida por el maestro es tarea de la conexión SCL, la cual define cuando debe ser leído un bit de la conexión SDA mediante pulsos cuadrados. Es decir, cuando la conexión SCL marca un nivel lógico de 1 el dispositivo debe leer la data. En la siguiente figura se puede ver un ejemplo de esto, en donde particularmente se esta enviando un valor binario 0b01001101 equivalente a un 77 en decimal.

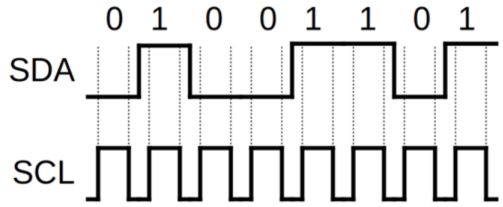

Figura 3.2: Ejemplo de data enviada por I^2C.

El maestro estará encargado de construir y enviar un paquete de datos para la comunicación con los esclavos. Dentro de este paquete se debe indicar el inicio y final (parada) de la conexión. Para el inicio, la conexión del pin SCL debe mantenerse en alto y dejar el pin SDA en bajo. Análogamente, la condición de parada se ponen ambos pin en alto, comenzando por el pin SCL y luego el pin SDA. El paquete de datos enviado por el maestro se compone

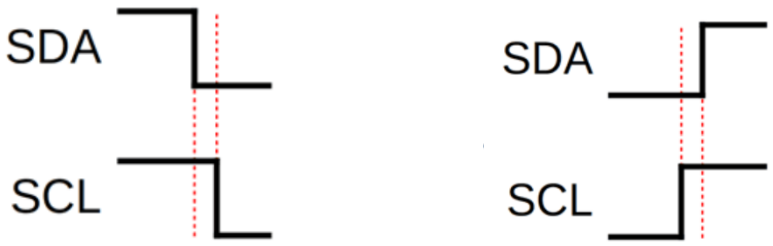

Figura 3.3: Condición de inicio (izquierda) y de parada (derecha) en comunicación I^2C.

de dos tramas generales: la de dirección (address) y la de data. La trama de dirección se compone por un byte con el address del esclavo con el que se abre la comunicación, por un bit de lectura/escritura (R/W) y un bit de reconocimiento (ACK o acknowledgement). La trama de data se compone por un byte de datos que se quieran escribir/entregar del registro y un bit de recono-

cimiento. Tomando todo esto en cuenta se construye el paquete de datos, cuya estructura se puede observar en la tabla 3.1.

Tabla 3.1: Paquete de datos I^2C

Start	Address	R/W	ACK	Data	ACK	Stop

3.1.3. Registros y parámetros

El MPU-6050 cuenta con un número de registros internos que controlan el funcionamiento y configuraciones del acelerómetro y giroscopio. Estos registros guardan información de la data sensada, configuraciones de los modos de operación y sensibilidad, además de los parámetros necesarios para su uso. Estos son accedidos por el master mediante la comunicación I^2C para leer o escribir sobre ellos, manipulando así el funcionamiento del componente.

La información de cada registro puede encontrarse en el mapa de registros del MPU-6050 [17]. En este documento se detalla la dirección de cada registro así como una breve descripción de cual función cumple y la manera de manipularlo, con un número de 32 registros distintos para acceder. Cada registro contiene una dirección, modo (escritura/lectura) y un byte de data. En particular tendremos interés en los registros que contienen la información de aceleración y velocidad ángular de todos los ejes. Además, debemos resetear y calibrar el componenete mediante los registros que definen los parámetros de este, como podrían ser: velocidad del reloj, la escala del acelerómetro y giroscopio, habilitación de la comunicación y la configuración de pines SDA y SCL.

3.1.4. Movimiento y orientación

Los valores de aceleración y velocidad angular nos permiten describir y comprender el movimiento y orientación del robot. Particularmente, podemos detectar **cuando** el robot comienza y finaliza su caminata, ya que existe una variación de velocidad al cambiar de estado de reposo a moverse en alguna dirección y viceversa. Esta variación de velocidad implica una aceleración en los ejes donde se efectúa el movimiento, lo cual es detectado por el acelerómetro. Análogamente, podemos detectar cuando el robot gire durante su caminata mediante la velocidad angular entregada por el giroscopio.

Si bien podemos detectar cuando ocurre un cambio de velocidad y orientación en el robot, nos interesa obtener valores continuos de velocidad, distancia y ángulos de inclinación a los que esta sometido durante su movimiento, esto para cada eje. Para ello podemos usar las relaciones físicas entre estas cantidades y los valores de aceleración y velocidad angular entregados por el componente. Específicamente, integrando estos dos valores podemos conseguir la velocidad y ángulo de inclinación del robot. Integrando una vez más la aceleración se obtiene la distancia recorrida por el robot. Las ecuaciones matemáticas que describen estas relaciones son

$$v_k = v_{k-1} + a_k \cdot \Delta t \tag{3.1}$$

$$d_k = d_{k-1} + v_k \cdot \Delta t + \frac{1}{2} \cdot a_k \cdot \Delta t^2 \tag{3.2}$$

$$\theta_k = \theta_{k-1} + \omega_k \cdot \Delta t \tag{3.3}$$

Donde v_k corresponde a la velocidad, d_k la distancia, a_k la aceleración, θ_k el ángulo de inclinación y ω_k la velocidad angular, todos para el instante k. Note que los valores descritos en las tres ecuaciones dependen del valor en el instante de tiempo $k-1$, es decir, de las condiciones iniciales del movimiento. Esto presenta un gran problema, pues como se verá en la sección de resultados, la aceleración y velocidad angular medidos por el componente presentan pequeños errores y variaciones a lo largo del tiempo. Estos errores son constantemente integrados por las ecuaciones de arriba, provocando que en un poco periodo de tiempo las mediciones de velocidad, distancia y ángulo de inclinación presenten un error acumulado (mientras más tiempo pase peor será la medición).

3.1.5. Filtro de Kalman

El error acumulado debido a la integración de pequeños errores de los valores sensados pueden ser procesados y eliminados mediante el uso de un filtro de Kalman [18]. El filtro de Kalman es un algoritmo utilizado para estimar el estado de un sistema dinámico que está sujeto a ruido e incertidumbre. Éste se basa en dos modelos matemáticos: un modelo sobre el sistema dinámico de interés y un modelo de medición. El modelo del sistema dinámico describe cómo evoluciona el estado del sistema a lo largo del tiempo. Este modelo se describe matemáticamente mediante un conjunto de ecuaciones que relacionan el estado actual del sistema con el estado anterior sus entradas. En este contexto, las ecuaciones del sistema dinámico corresponden a la relación física

entre descritas previamente en (3.1), (3.2) y (3.3), donde las variables corresponden a la velocidad y distancia. Por su parte, el modelo de medida describe cómo las observaciones se relacionan con el estado del sistema. En este caso, correspondería a los valores sensados por el acelerómetro giroscopio. Estos dos modelos son combinados linealmente y ponderados mediante la ganancia de Kalman, la cual minimiza el error cuadrático medio entre el estado y el valor medido.

El filtro de Kalman utiliza estos modelos para calcular una estimación óptima del estado del sistema en cada instante de tiempo a través de dos pasos recursivos: la fase de predicción y la fase de actualización. La fase de predicción usa el modelo dinámico junto a una matriz de covarianza para estimar el estado y la incertidumbre del sistema en el siguiente instante de tiempo. Las ecuaciones generales de la predicción del estado y la incertidumbre son

$$x[k] = A[k-1] \cdot x[k-1] + B[k-1] \cdot u[k-1] \tag{3.4}$$

$$P[k] = A[k-1] \cdot P[k-1] \cdot A[k-1]^T + Q[k-1] \tag{3.5}$$

Donde $x[k]$ representa la predicción del estado del sistema, $u[k]$ es la entrada del sistema, $A[k]$ es una matriz de trasición representando el modelo dinámico, $B[k]$ es la relación entre la entrada y la predicción del estado, P[k] es la predicción de la incertidumbre y $Q[k]$ es la matriz de covarianza representando la incertidumbre real del sistema. Usando estos valores es posible calcular la ganancia de Kalman $K[k]$ mediante las siguientes ecuaciones

$$L[k] = H \cdot P[k] \cdot H^T + R \tag{3.6}$$

$$K[k] = P[k] \cdot H^T \cdot L[k]^{-1} \tag{3.7}$$

Donde H indica cual estado del sistema será observado, R es la incertidumbre del valor observado y $L[k]$ una matriz de intermediaria. Luego, se procede con la fase de actualización. Aquí se actualiza la predicción del estado e incertidumbre del sistema mediante el uso de la ganancia de Kalman, el valor observado y el estado real medido $M[k]$.

$$x[k] = x[k] + K[k] \cdot (M[k] - H \cdot x[k]) \tag{3.8}$$

$$P[k] = (I - K[k] \cdot A[k]) \cdot P[k] \tag{3.9}$$

Donde I es la matriz identidad. Finalmente, se toman el nuevo estado e incertidumbre del sistema y se itera N veces por el algoritmo.

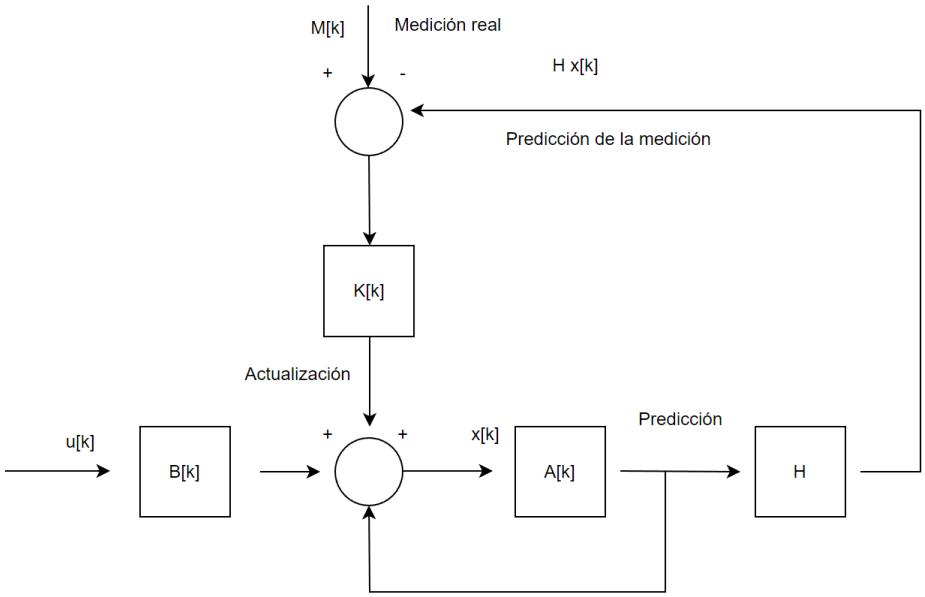

Figura 3.4: Diagrama de bloques del filtro Kalman.

3.1.6. Velocidad y distancia recorrida

La aplicación del filtro de Kalman para el uso del acelerómetro permite la obtención de la velocidad y distancia evitando el error acumulado por la integración de la aceleración. Por tanto, debemos aterrizar la teoría mediante la definición de las variables usadas en el algoritmo. Primero, definimos un vector de estados con los valores de interés. Es decir, tendremos un vector de estado de tres dimensiones, definido por

$$x[k] = \begin{bmatrix} d_k \\ v_k \\ a_k \end{bmatrix}$$

Los valores iniciales de este vector corresponde al vector nulo, ya que al inicio del movimiento no existe distancia recorrida, velocidad ni aceleración. Luego, definimos las ecuaciones dinamicas de forma matricial

$$A[k] = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

Como el acelerómetro nos entrega mediciones reales de la aceleración experimentada en cada eje, la matriz de observación viene definida por

$$H = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Respecto a las matrices de incertidumbre $P[k]$, R y $Q[k]$ deben ser encontrados o estimados con los datos reales. Análogamente, la variación de tiempo dependerá de la frecuencia de toma de datos. Finalmente, el valor medido $M[k]$ corresponde a la medición real de aceleración. Usando estas matrices e iterando en el algoritmo se consigue la distancia, velocidad y aceleración sin ruido.

3.1.7. Ángulo de inclinación

La adquisición del ángulo de inclinación mediante la integración de la velocidad angular entregada por el giroscópio presenta los mismos problemas de acumulación de error. Sin embargo, a diferencia del acelerómetro, no tenemos un valor real de ángulo de inclinación para poder corregir los valores predichos por el algoritmo de Kalman ni tampoco podemos saber las condiciones iniciales del problema. Por ende, debemos obtener los ángulos mediante otro método.

Una solución posible es presentada por CarbonAeronautics [19], en donde mediante relaciones trigonométricas con los valores vectoriales de aceleración es posible obtener los ángulos de inclinación del componente. Para ejemplificar esto supongamos que el MPU-6050 se encuentra en reposo con sus ejes alineados de forma que la aceleración de gravedad es detectada en el eje z. Luego, el componente sufre una rotación en el eje x, como se ve en la figura 3.5. Usando la definición de la tangente de un ángulo, la inclinación del eje y

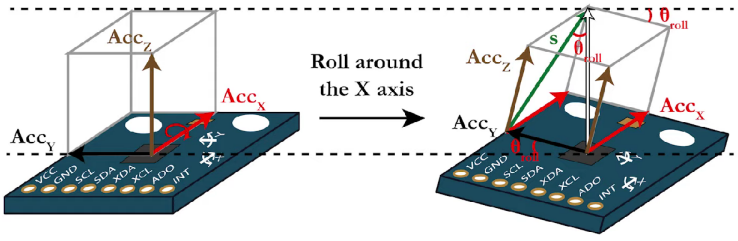

Figura 3.5: Representacion de vectores para la obtención de ángulos de inclinación. Imagen elaborada por [19].

mantiene una relación con las cantidades vectoriales de aceleración de la forma

$$tan(\theta_{roll}) = \frac{Acc_Y}{s} \tag{3.10}$$

Donde, por el teorema de Pitágoras

$$s = Acc_X^2 + Acc_X^2 \tag{3.11}$$

Por tanto, despejando el ángulo de inclinación se tiene

$$\theta_{roll} = \theta_Y = atan\left(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right) \tag{3.12}$$

Análogamente para la inclinación del eje z

$$\theta_Z = atan\left(\frac{Acc_Z}{\sqrt{Acc_X^2 + Acc_Y^2}}\right) \tag{3.13}$$

La obtención de los ángulos de inclinación nos permite aplicar el filtro de Kalman nuevamente. Definimos el vector de estado como

$$x[k] = \left[\theta_k\right]$$

En este caso, el valor inicial de este vector dependerá del ángulo de inclinación que exista previamente al uso del algoritmo. La entrada al sistema dinámico va definida como

$$u[k] = [\omega_k]$$

Las matrices del sistema dinámico y de relación entre entrada y estado son simples, definidas por

$$A[k] = [1]$$

$$B[k] = [\Delta t]$$

Como existe solo un valor de interés, la matriz de observación es

$$H = [1]$$

Respecto a las matrices de incertidumbre P[k], R y Q[k] deben ser encontrados o estimados con los datos reales. Finalmente, el valor medido $M[k]$ es igual a los ángulos medidos en (3.12) y (3.13) dependiendo que eje estemos analizando.

3.2. Detección de Pisadas

La detección de pisadas en el ArgoV2 es una tarea no trivial debido a las complicaciones que presenta el modelo de las piernas del robot. Como se mostró previamente (Figura 2.5) la estructura de la pierna del robot es asimétrica, cambia su grosor a medida que se acerca al pie y finaliza como un tipo de esfera en la que se puede encajar la bota flexible. Debido a que la estructura esférica no llena el interior de la pieza flexible esta presenta una deformación con el contacto con el piso. El primer acercamiento a una detección de pisadas es sensar esta pequeña deformación con algún sensor táctil o de presión, pudiendo así discriminar si la extremidad pisa el suelo o no.

Existen muchas formas de realizar una detección de pisadas en robots móviles, así como una gran variedad de sensores que permiten el sensado de información útil para distinguir si la extremidad del robot tuvo un contacto con el piso. Dentro del diseño desarrollado en esta memoria se contemplaron tres principales sensores que permitieran detectar el contacto con el piso. En estas posibilidades esta un sensor de ultrasonido HC-SR04 [20], el cual entregaría información sobre la distancia que hay entre el piso y la extremidad mediante el tiempo que demora el sonido en volver al sensor y la velocidad en la que esta viaja. Este componente trabaja a 5V y presenta una medición entre 2 a 400 [cm] a un angulo entre 0 y 15 grados. Por tanto, este componente nos permite presindir de la deformación de la bota del robot. Si bien la precisión de este sensor es apropiada para este trabajo de memoria, las dimensiones de este complican grandemente el diseño de una nueva pierna, pues la integración de este podría obstruir o complejizar la caminata del robot.

Figura 3.6: Sensor de ultrasonido.

La segunda opción fue el uso de un sensor táctil FSR (Force Sensing Resistor) [21]. Este sensor es una resistencia en forma de lámina fina formada por materiales piezo-resistivos. Esta resistencia es de forma circular con un diámetro de 12,5$[mm]$, un grosor de 0,3$[mm]$ y es flexible en gran parte de su composición. Respecto a su integración al robot, hay que tomar en cuenta que su resistencia tiene un comportamiento no-lineal. En particular, al no ejercer presión se comporta como un circuito abierto y luego su resistencia decrece exponencialmente en un rango de $[100K - 200][\Omega]$ a una presión de $[0 - 100][N]$. Para aprovechar este funcionamiento, una fácil integración del componenete es la medición de voltaje en la resistencia a través de un divisor de voltaje. A diferencia del componente anterior, las dimensiones del FSR son suficientemente pequeñas para integrarlos a la pierna del robot. Sin embargo, el peso del robot no presenta suficiente fuerza para deformar el componente y poder obtener un rango aceptable para detectar si hubo o no contacto con el piso.

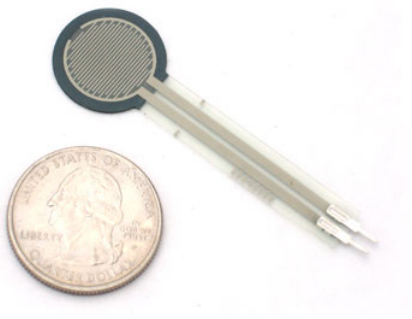

Figura 3.7: Sensor táctil FSR.

Teniendo en cuenta la problemática de que la deformación de la bota flexible no es suficiente para obtener un rango plausible de detección de pisadas, se optó por una tercera opción. Lei Zhang et al [22] presentan un diseño de un pie flexible para robots de extremidades múltiples. En este trabajo se desarrolla una pierna para detectar pisadas bio-insparada en el funcionamiento de la extremidad de un insecto. Como mencionan, el funcionanmiento de la extremidad de un insecto se conforma por dos partes: la almohadilla de la pierna del insecto, la cual al contacto presenta una deformación que se adapta a la complejidad del piso amortiguando la fuerza ejercida hacia el resto de la extremidad y una membrana flexible que se conecta a la almohadilla y se encarga de terminar de amortiguar el impacto para proteger las articulaciones del insecto. Así, ellos diseñaron un mecanismo de pie robótico con dos grados de libertad y formado por seis partes: una suela flexible (Flexible sole), una base (Pedestal), una varilla de empuje (Push rod), un resorte de compresión, un sensor de presión y una placa inferior (Bottom Plate). Este mecanismo seguiría el funcionamiento de la extremidad del insecto: el contacto con el piso deformaría la suela flexible ubicada en la base empujando la varilla de empuje, esta a su vez comprime el resorte quién entra en contacto con el sensor de presión ubicado en la placa inferior, resultando en una medición de voltaje distinta a cuando la extremidad no esta en contacto con el piso. El diseño utiliza un sensor de fuerza de bajo perfil Honeywell FSS [23], el cual tiene un rango de detección de 0-20[N] y es de pequeño tamaño.

La implementación de este pie robótico encaja muy bien en el modelo del ArgoV2, ya que este presenta una estructura tipo varilla en donde se encaja la bota flexible hecha en filamento TPU. Esto permitiría poder aprovechar la deformación de la bota mediante el diseño de un sistema mecánico dentro de la pierna del robot, de modo que se haga contacto con un sensor dentro de

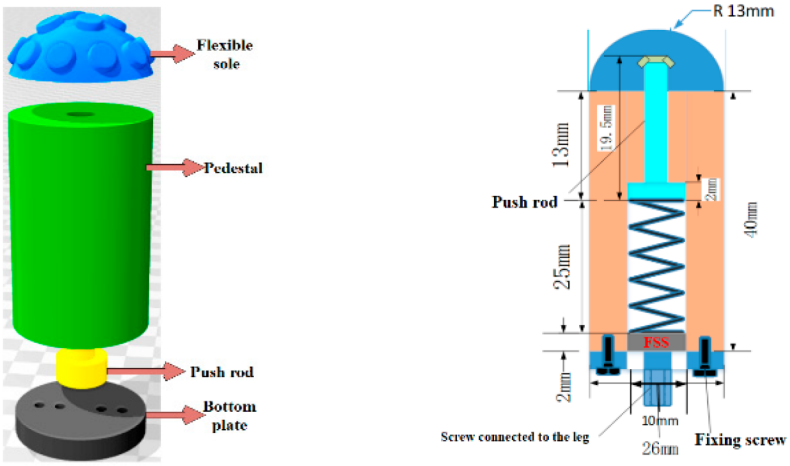

Figura 3.8: Modelo simplficado (izquierda) y detallado (derecha) del pie flexible. Imagen elaborada por [21]

este mismo. Pero ocurre un nuevo problema, las dimensiones de la extremidad de la pierna corresponden a un elipsoide de 2,5[cm] x 2 [cm] de ancho, lo que complejiza grandemente la inclusión del sensor táctil FSR. Por ende, se consideró la inclusión de un nuevo sensor que tuviera las dimensiones necesarias para poder ubicarse dentro de la pierna del robot. El componente elegido es un sensor de luz LDR (Light Dependen Resistor). Este componente es una resistencia cuyo valor disminuye a medida que se presenta mayor luz en su superficie. Tiene dimensiones de 6 x 7 [mm] y varía su resisitividad entre 2 y 10 [kΩ]. Similarmente al FSR, este sensor puede ser incluido circuitalmente mediante un divisor de tensión y una resistencia de medición arbitraria.

Figura 3.9: Sensor de Luz LDR.

3.2.1. Implementación del pie flexible al ArgoV2

Considerando el diseño presentado en [22], se eliminó el sensor de presión para poder adaptar el sistema mecánico a la extremidad del ArgoV2. Siguiendo la estructura del pie robótico, se diseño de manera que se ubicara un LED (Light Emitting Diode) en la placa inferior (donde previamente estaba el sensor de presión) y el sensor LDR en la extremidad de la varilla de empuje que se encuentra en contacto con el resorte. De esta forma cuando la extremidad del robot entra en contacto con el piso la suela flexible empuja la varilla de empuje comprimiendo el resorte y disminuyendo así la distancia que existe entre el LED y el sensor LDR. Esta variación de distancia se traduce en una mayor cantidad de luz recibida por el sensor LDR, permitiendo poder detectar mayores valores de voltaje cuando hay contacto con el piso y viceversa.

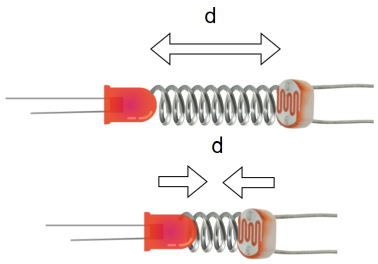

Figura 3.10: Sistema mecánico incluyendo un LDR y un LED.

Una vez el nuevo funcionamiento del sistema mecánico fue diseñado, se entró a diseñar el modelo 3D del robot. Para ello se debió adaptar las seis piezas del sistema mecánico original, teniendo en consideración la inclusión del LED y el LDR. Este re-diseño se enfoca principalmente en la pierna de robot, en donde mediante la adición de nuevos modelos 3D, diseñados en el software Fusion360, se logra adaptar el sistema al ArgoV2.

3.2.2. Nuevo sistema mecánico

El nuevo sistema mecánico sigue la estructura del pie flexible con unas pequeñas modificaciones que permiten incluir los sensores **dentro** de la pierna del robot. Esto implica la eliminación de la placa inferior, pues el modelo 3D de la pierna cumple el funcionamiento de posicionar las demás piezas del sistema mecánico. Se modificaron las piezas correspondientes a la base o pedestal, la

varilla de empuje y el resorte de compresión, para la ubicación del LED y el LDR. Además, se creo la nueva pieza **Seguro**, la cual se encarga de mantener el sistema a presión dentro de la pierna. En la figura se puede observar la sección transversal del nuevo sistema mecánico, en donde se indica como va ubicado cada modelo 3D.

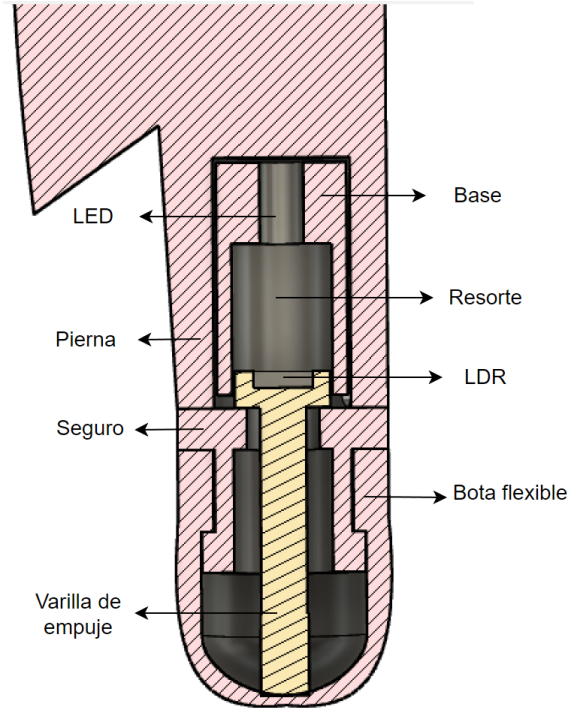

Figura 3.11: Sección transversal del nuevo sistema mecánico.

3.2.3. Rediseño de pierna del ArgoV2

El primer paso para la inclusión del sistema mecánico es la edición de la pierna del robot. Comparando con el sistema de [22], la pierna del robot cumpliría la función de la pieza placa inferior. Es decir, esta pieza funciona como base para que todas las piezas posteriores puedan ubicarse correctamente. Para ello se eliminó la estructura en forma de varilla al final de la extremidad dejando así una superficie plana hacia la bota. Luego, se perforó esta superficie plana con cilindro de 30 [mm] de largo y un diámetro de 16[mm]. En la siguientes secciones se detallará que para mantener este nuevo sistema mecánico firme se necesitan de tornillos. Además, como el LED y el LDR se ubican dentro de la pierna, debe haber un riel para los cables de sus pines. Por ende también se perforó la superficie con un cilindro de 3 [mm] de diámetro y 12 [mm] de largo para los tornillos. Para los cables, se perforaron 4 cilíndros de 2[mm] de diametro a lo largo de toda la extremidad, dos comenzando desde el interior de la pierna (LED) y dos desde la extremidad de esta (LDR). El perfil de la nueva superficie plana del extremo de la pierna y una comparación entre el modelo antiguo y nuevo se puede observar en la Figuras 3.12 y 3.13.

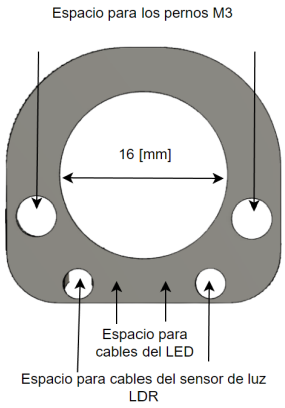

Figura 3.12: Perfil de la nueva extemidad de la pierna.

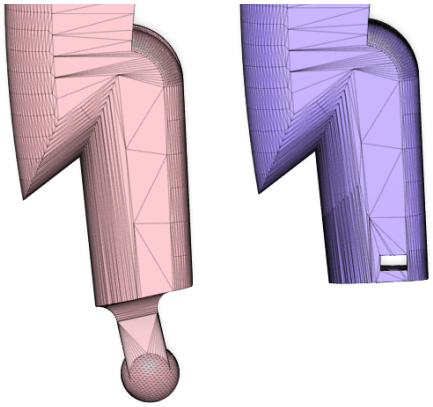

Figura 3.13: Comparación entre la pierna antigua (rosado) y la nueva (morado).

3.2.4. Base o Pedestal

La siguiente pieza a diseñar corresponde a la base o pedestal. Análogamente al sistema de [22], la base cumple la función sostener el resorte y la varilla de empuje. Además, para el nuevo sistema mecánico, esta pieza se encargará de

ubicar el LED. Se creó la pieza como un cilindro hueco en uno de sus extremos y cerrado en el otro, con dimensiones de 30 [mm] de largo, 14 [mm] de diametro interno y 16 [mm] de diametro externo. El extremo cerrado tiene un grosor 10 [mm], en donde se creo una hoyo circular de 4,9[mm] de diametro para ubicar el LED. Finalmente se crearon dos rieles para el paso de los cables y se redujeron todas las dimensiones de la base en 1[mm] para que esta encajara perfectamente en la pierna.

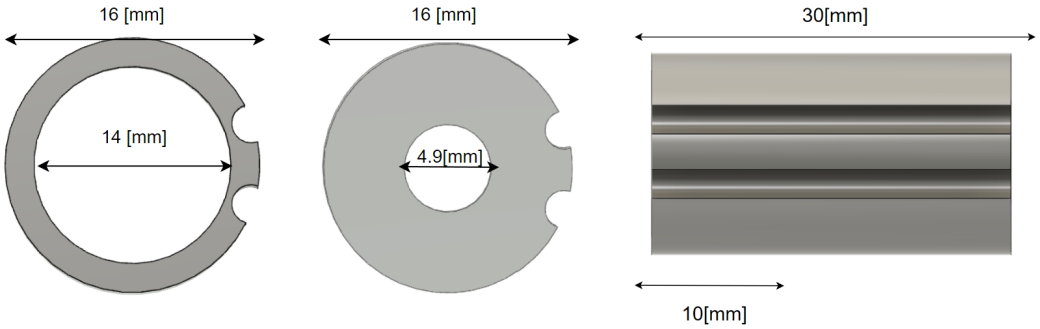

Figura 3.14: Dimensiones y perfil de la base.

3.2.5. Resorte

El resorte se ubica dentro de la base, dando espacio para el LED en un extremo y la varilla de empuje en el otro. Las dimensiones de la base limitan el largo del resorte elegido, ya que con el espacio creado para el LED solo deja 20[mm] libres. Por ende, simplemete se tomó un resorte de compresión de 15[mm] de largo y conformado por cuatro vueltas. Así, quedan 5 [mm] dentro de la base para la varilla de empuje.

3.2.6. Varilla de Empuje

La varilla de empuje es la que se encarga de transformar la deformación de la bota del robot en una fuerza que comprime el resorte. También, para el nuevo sistema, es la pieza encargada de ubicar el sensor de luz LDR. Por ello se diseño la pieza con dos partes principales. La primera es un cilindro de 12[mm] de diámetro y 5 [mm] de ancho. Esta parte cuenta con un espacio en forma de rectangular de 8[mm] x 7[mm] x 2[mm] en su centro, con la funcionalidad de ubicar el LDR. La segunda parte corresponde a la varilla, un cilindro de 5 [mm] de diámetro y 35[mm] de largo. Esta pieza tambien cuenta con orificios para el paso de los pines del componente y rieles para el paso de los cables.

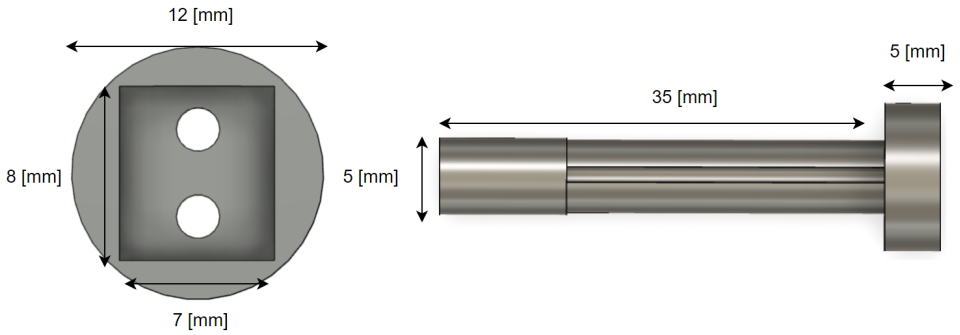

Figura 3.15: Dimensiones y perfil de la varilla de empuje.

3.2.7. Seguro

Para la adaptación del sistema mecánico a la pierna del ArgoV2 se agregó una nueva pieza/modelo 3D. Como las piezas mencionadas anteriormente se ubican dentro de la extremidad del robot, necesitamos que estas se mantengan fijas y no muestren movimiento durante la caminata del este. Por tanto, se creó la pieza Seguro, la cual se conforma de una estrutura que sigue la forma de la pierna por 5 [mm] con un orificio de 10 [mm] de diámetro. Este orificio esta hecho de forma que que la varilla pueda moverse hacia arriba y abajo hasta que su movimiento sea limitado por el cilindro donde esta ubicado el sensor LDR (parte superior de la varilla de empuje). Luego, se amplía el diametro del orificio a 12 [mm] continuando con una estructura cilindrica de 10[mm] de largo y 18[mm] de diámetro exterior, para finalizar con otro aumento a su diámetro exterior a 20[mm] durante 5[mm] de largo, esto para que la bota flexible pueda entrar a presión a ella. Para asegurar que esta pieza logre mantener todas las demás fijas se realizó una perforación para el paso de un tornillo M3 a lo largo de todas las piezas juntas, el cual mediante la ubicación de una tuerca en la pierna del robot mantiene todo el sistema fijo a presión. También se agregaron espacios para los cables del sensor LDR ubicado en la varilla de empuje, dándole a este espacio una forma ovalada para que exista un poco más de libertad a los cables debido a que la varilla estará en movimiento.

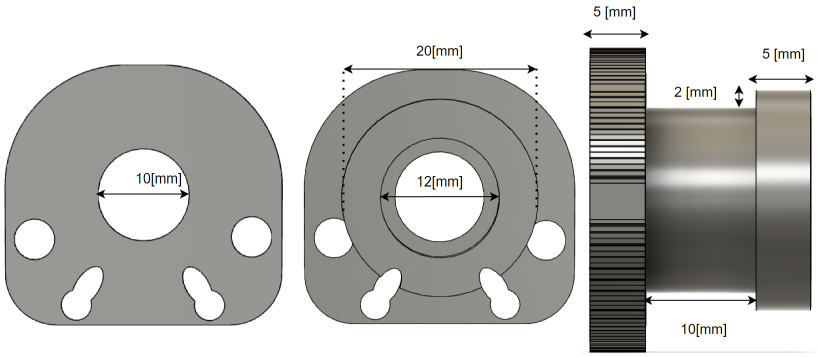

Figura 3.16: Dimensiones y perfil del Seguro.

3.2.8. Bota Flexible

El re-diseño del sistema mecánico para el ArgoV2 elimina la varilla que posee la pierna original del robot. Por ende, el diseño de la bota flexible actual no coincide con el sistema nuevo, principalmente porque la bota ya no se puede encajar en ninguna parte. Esto nos invita a re-diseñar desde cero la estructura de la bota del robot, adaptandola al sistema del LED y LDR, además de hacerla encajar a presión en la pieza seguro. Debido a estas razones se creo una nueva bota flexible. Esta nueva pieza mantiene la estructura elíptica del final de la pierna durante 15[mm] con un hueco en forma de circunferencia de 18 [mm] de diámetro para encajar en el seguro. Luego, la pieza termina en un sólido de revolución generado por una elipsoide de dimensiones 10[mm] x 8 [mm]. Este elipsoide crea una estructura hueca, lo que permite que haya espacio para los cables del sensor LDR y exista posibilidad de que la pieza se deforme al contacto con el piso. Finalmente, se hicieron espacios para los cables del LDR y los tornillos M3.

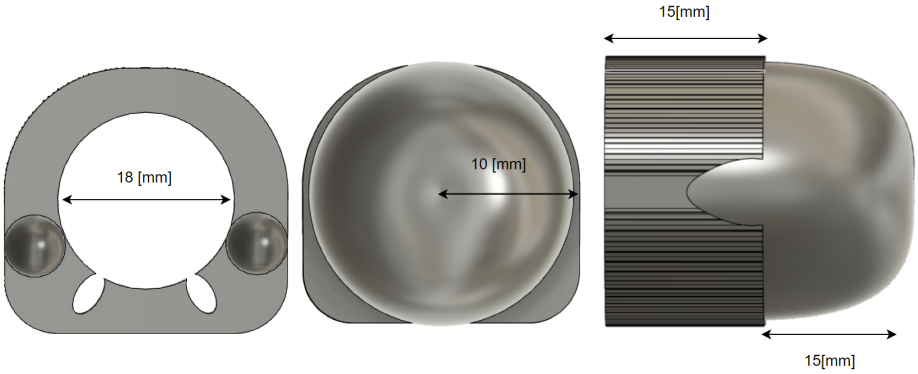

Figura 3.17: Dimensiones y perfil de la nueva bota flexible.

Recordemos que la bota flexible actual esta hecha en filamento TPU, característico por su flexibilidad y deformación. Sin embargo, en [22] la suela flexible esta hecha con silicona. Para mantener coherencia con su pie flexible, se crearon una serie de moldes que permitieran construir la nueva bota flexible de silicona. El molde es una caja de dimensiones 50 x 50 x 50 [mm], moldeando la forma de la nueva bota flexible. Se forma de 3 partes principales. Las primeras dos partes se crean de un corte transversal en la caja a una altura de 30[mm] desde el piso. La tercera pieza corresponde a una estructura compleja que se ubica entre las dos piezas anteriores y cumple la función de moldar el interior de la bota flexible dandole así el grosor y formas indicados para que este pueda entrar a presión en el seguro.

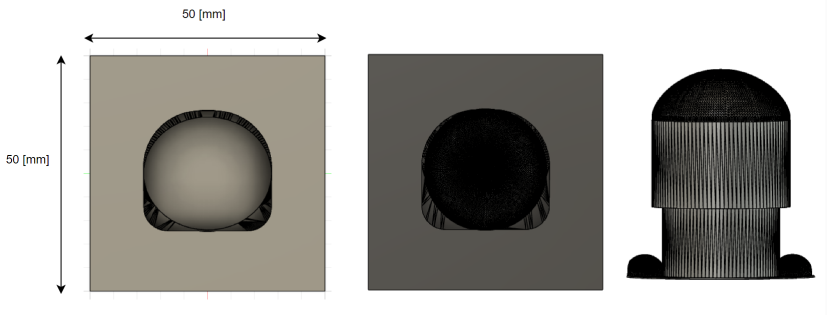

Figura 3.18: Dimensiones y perfil del molde.

Una vez todas las piezas del nuevo sistema mecánico fueron diseñadas, se procedió a ubicarlas dentro del software Fusion360. Aquí se comprobaron que las dimensiones de todos los modelos encajaran entre sí, reduciendo todas ellas en 1 [mm] de grosor para la impresión.

Capítulo 4

CONTROL Y COMUNICACIÓN CON EL ROBOT

El paso final para el diseño del ArgoV2 es poder establecer una comunicación que entregue la información de la data sensada a un PC común. La idea es poder conservar la comunicación serial entre PC y motores que usa actualemente el robot, por lo que necesitamos un componente que obtenga y manipule la data sensada enviando paquetes de datos a un PC sin que el usuario interactúe directamente con él. Para ello, se tiene que disponder de un **controlador** y de un **protocolo** de comunicación. El controlador debe cumplir con las condiciones de diseño impuestas por los sensores y componentes antes descritos. Es decir, debe ser capaz de poder transmitir comunicaciones seriales Half-duplex UART para el control de motores, I^2C para el control del acelerómetro giroscopio, y además de contar con al menos cuatro periféricos para la medición de divisores de tensión para la detección de pisadas. El protocolo debe ser capaz de integrar la comunicación usada por los motores Dynamixel para el uso de sus librerías y entregar la información del acelerómetro giroscopio y los sensores de luz en una sola conexión serial.

4.1. BlackPill - STM32F401CB

La elección del controlador que manejará la información es el microcontrolador STM32F401 de la marca ST [24]. El STM32F401 o más conocido como BlackPill abastece las necesidades que el proyecto impone. Su CPU corresponde a un 32-bit ARM Cortex-M4 operando a una frecuencia de hasta 84 MHz,

es decir, sus registros e instrucciones son de 32-bit, permitiendo así el uso de tipos de datos de largo almacenamiento. Cuenta con una rápida respuesta a sus memorias incorporadas (memoria Flash de 256 Kbyte y memoria SRAM de 64 Kbyte) además de 48 pines configurables. El dispositivo ofrece también un ADC de 12-bit con hasta 16 canales, un reloj de tiempo real de poco consumo y permite comunicaciones estándar como podrían ser: UART, I^2C, SPI, USB, y más.

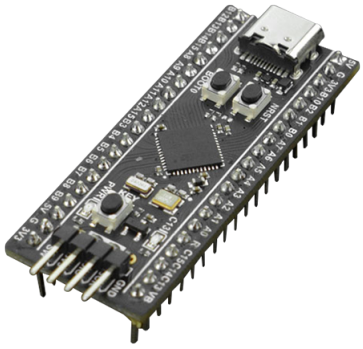

Figura 4.1: Microcontrolador STM32F401.

4.1.1. STM32CubeIDE y HAL

La marca ST presenta el software STM32CubeIDE para la programación y depuración de sus micro-controladores. Este software facilita la interacción con el dispositivo mediante el uso de librerías HAL (Hardware Abstraction Layer) [25], las cuales permiten la abstracción de hardware de bajo nivel mediante APIs (Application Programming Interfaces) de alto nivel. Esto se traduce en una mayor simpleza para la implementación de código para el manejo y control de registros del componente. Las HAL cuentan con una gran variedad de APIs para el control y configuración de periféricos que nos serán de interés como USART, ADC, I^2C, etc. En particular, permite el control de características de bajo nivel como DMA (Direct Memory Access) o interrupciones, las cuales serán claves para la implementación del diseño de las comunicaciones de la detección de pisadas y movimiento. El lenguaje de programación del software puede ser en C o C++, siendo usada esta última para este proyecto.

Lo más importante de las librerías HAL es que permiten fácilmente transformar una comunicación UART de dos conexiones (full-duplex) a una de

una conexión (half-duplex), transmitir datos a un slave por I^2C y manejar múltiples canales del ADC por los periféricos. Con estas especificaciones el controlador es capaz de manejar toda la información sensada en la detección de pisadas y movimiento, además de controlar los motores Dynamixel para la caminata del robot.

4.1.2. Direct Memory Access (DMA)

Las librerías HAL permiten manejar de manera muy simple el DMA o Direct Memory Access. Como su nombre lo indica, es una técnica utilizada en microcontroladores STM32 para permitir que ciertos periféricos de entrada/-salida accedan directamente a la memoria principal del sistema, sin tener que pasar por la CPU. Esto presenta muchas ventajas, pues se pueden recibir datos de manera rápida sin que la CPU gaste tiempo y recursos en la transferencia de datos.

Cuando se inicia una transferencia usando DMA se asigna temporalmente un bloque de memoria para almacenar los datos recibidos. Estos datos pasan directamente a la memoria Flash hasta finalizar la transmisión, en donde se avisa a la CPU que los datos han sido transferidos mediante interrupciones.

El DMA es usado posteriormente en el código del BlackPill, en donde se recibe data tanto de UART como del ADC en segundo plano. La data es recibida hasta que ocurre una interrupción, dando paso al controlador a tomar decisiones sobre a quién y que datos enviar.

4.1.3. Conversor Bidireccional 3.3 - 5 [V]

Como fue mencionado en el capítulo del robot ArgoV2, los niveles lógicos de los paquetes de datos enviados a los motores Dynamixel son de 5[V]. Esto complejiza el diseño del control del robot, pues la gran mayoría de los microcontroladores de la industria emiten señales de voltajes bajos (1.7-3.6[V]). La solución es usar un conversor bidireccional que convierta 3.3[V] a 5[V]. Su funcionamiento es simple, se debe disponer de 5 conexiones, 4 de entrada y 1 de salida. Las conexiones de entrada corresponden ambos niveles lógicos (3.3 y 5 V]), la data que se quiere nivelar y tierra. La conexión de salida corresponde a la data nivelada a 5[V]. Entonces, necesitamos disponer de una fuente de 5[V], tarea que cumple el componente que comunica los datos hacia el PC.

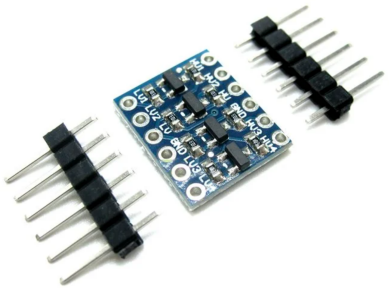

Figura 4.2: Conversor Bidireccional.

4.2. Comunicación con el PC

Finalmente se debe estandarizar una comunicación entre el controlador y el usuario vía conexión serial. Como se mencionó antes, el controlador será "invisible" para el usuario quién solo interacutará con los sensores y motores mediante el envio de data. Por ende, el controlador solo tomará decisiones de que, cuando y como envía data hacía el usuario según este la pida.

4.2.1. Chip FT232

La comunicación entre el controlador y el PC será manejada por el chip FT232 de la compañia FTDI (Future Technology Devices International) [26]. Este componente convierte las señales de una conexión serial en señales de protocolo USB y viceversa. Se alimenta desde una conexión Mini-USB a USB, por donde mismo se reciben y envían datos desde un PC convencional. Buscando no complejizar más el diseño del robot se decidió que la conexión serial entre el controlador y el PC será mediante UART full-duplex, es decir, dos conexiones físicas (Tx, Rx). La velocidad de comunicación es definida por un Baud Rate de 115200 [bps]. El componente presenta 6 pines para la

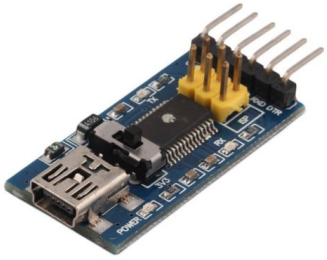

Figura 4.3: Chip FT32 de FTDI.

manipulación de data. Dos de estos pines corresponde a la alimentación, donde el pin de tierra (GND) debe ser conectado a la misma tierra que el controlador para conservar la referencia. Existe un segundo pin el cual entrega un voltaje de 5[V] siempre y cuando se este energizando con el conector Mini-USB. Este pin será clave para entregar el nivel alto para el conversor lógico usado en los motores Dynamixel (recordemos que estos funcionan a 5[V]). Los otros dos pines a utilizar son el Tx y Rx, los cuales deben conectarse a los pines Rx y Tx del controlador en ese orden, quién trasmitirá paquetes de datos mediante UART.

4.2.2. Protocolo ArgoV2

El protocolo de comunicación entre controlador y PC tiene la importante tarea de poder incluir el Protocolo Dynamixel 1.0 dentro de su funcionamiento agregando la información sensada por los componentes en la detección de pisadas y movimento. También buscamos que este protocolo sea capaz de comunicarse directamente con los motores y sensores sin tener que interactuar directamente con el controlador, si no entregarle a este la información de que dato se quiere obtener o manipular para que tome decisiones de que paquete de datos envia al PC.

Por ende se creo el **Protocolo ArgoV2**, protocolo asincrónico serial de 8 bit, 1 bit de stop y ninguno de paridad. Este protocolo cumple con las condiciones descritas previamente tomando como base los paquetes de datos enviados por el Protocolo Dynamixel 1.0, agregando un nuevo formato de paquete usado solo para la data sensada. Este nuevo formato sigue la estructura de los paquetes de datos del protocolo Dynamixel, realizando cambio menores de modo que el controlador pueda discriminar si el paquete que recibió se dirige hacia los motores o sensores. Es importante mencionar que este nuevo formato solo tendrá como objetivo obtener los datos sensados y no modificar ni manipular la configuración de los sensores. Por tanto, similarmente al protocolo de los motores, se enviará un paquete de instrucción indicando que dato se quiere recibir y el controlador enviara un paquete de data con el dato pedido. Los campos del nuevo formato son:

Header Al igual que en el protocolo Dynamixel, el header es el campo que indica el inicio del paquete. Es formado por dos bytes distintos con el valor de 240 (0xF0). De esta forma el controlador será capaz de discriminar si el paquete de datos enviado quiere comunicarse con los motores (header 0xFF) o recibir la data por los sensores (header 0xF0).

ID o Data Si el paquete es de instrucción, en el campo del ID se indicará cual es el sensor que se quiere obtener información. Si el paquete es de data, aquí se podrá obtener la aceleración y rotación de los 3 ejes además de la medición de los 4 canales del ADC.

Length Este campo indica el largo en bytes de los datos entregados por el sensor y del *checksum*.

Checksum Es usado para revisar si el paquete es dañado durante la comunicación. Se calcula como la negación de la suma de todos los campos anteriores.

Con el diseño de este nuevo formato para los sensores del robot es posible comunicarse con él mediante una sola conexión serial, solo se debe programar el controlador para que tome decisiones de a quién enviar datos mediante la detección del Header del paquete. Los valores a usar en el campo de ID/Data se pueden observar en la tabla 6.6, donde se entrega el valor en bytes que se debe usar, el sensor que entregará la data y una pequeña descripción de que es enviado.

Tabla 4.1: Tabla de posibles valores entregados por los sensores

Valor	Sensor	Description
0x01	LDR	Entrega el voltaje medido en el canal 1 del ADC
0x02	LDR	Entrega el voltaje medido en el canal 2 del ADC
0x03	LDR	Entrega el voltaje medido en el canal 2 del ADC
0x04	LDR	Entrega el voltaje medido en el canal 4 del ADC
0x05	Acelerómetro	Entrega la aceleración del eje x
0x06	Acelerómetro	Entrega la aceleración del eje y
0x07	Acelerómetro	Entrega la aceleración del eje z
0x08	Giroscopio	Entrega la rotación del eje x
0x09	Giroscopio	Entrega la rotación del eje y
0x10	Giroscopio	Entrega la rotación del eje z

Capítulo 5

IMPLEMENTACIÓN DEL DISEÑO

5.1. Impresión y Construcción del Robot

El primer paso para implementar el nuevo diseño del robot ArgoV2 es la impresión y construcción de este mismo. Los modelos 3D diseñados para el sistema mecánico de la detección de pisadas fueron impresos en una impresora 3D Ender-3 v2 [18], usando filamento PLA de color negro. Además, las piernas, los marcos y el centro del robot también fueron impresos en este filamento. Respecto a la bota flexible, se usó el molde diseñado con una solución de silicona FDA (Food and Drug Administration) de color verde. La silicona FDA es un elastómero de silicona de alta calidad que se caracteriza por su alta resistencia a la temperatura, flexibilidad, estabilidad química y biocompatibilidad. Estas características evidencian que el material cumpliría los estándares para la detección de pisadas, que sería la necesidad de un material resistente, flexible y que se deforme al contacto con el piso. Cada bota fue curada durante cuatro horas y 20 minutos antes de su uso en el robot.

Posteriormente, se procedió a armar el sistema mecánico de cada pierna del robot integrando el LED y LDR a las piezas base y varilla de empuje. Se usó un adhesivo instántaneo para asegurar los compenentes en los modelos 3D y cables de silicona flexible para asegurar que el movimiento de la varilla de empuje no arruinara el funcionamiento del sistema mecánico. Se ubicaron las piezas dentro de la pierna siguiendo la figura 3.11, fijando el sistema con tor-

nillos y pernos M3 y colocando la bota flexible a presión en el seguro. Cada pierna fue construída usando cables de un color específico, de modo que durante las pruebas y resultados se pudiera identificar facilmente cada una de ellas. En particular, para las piernas uno, dos, tres y cuatro se usaron cables de color rojo, blanco, azul y amarillo respectivamente.

Una vez el sistema mecánico fue integrado a cada una de las piernas, se procedió a ubicar los tres motores Dynamixel como articulaciones. Los motores van asegurados con pernos y tornillos M1, colocandose entre las piernas y los marcos 1 y 2. Cada motor fue calibrado entre los ángulos límite explicados anteriormente, cambiando sus ID entre 1-12 a lo largo de las cuatro piernas.

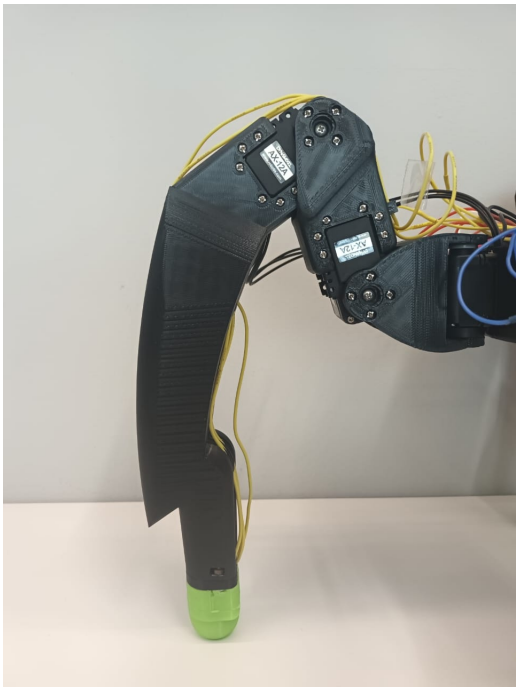

Figura 5.1: Una de las nuevas piernas del ArgoV2.

5.2. BlackPill con STM32CubeIDe

El segundo paso para la implementación del diseño es la configuración y conexión de los componentes para la detección de pisadas y movimiento además de la programación y depuración del controlador que se comunicará con el PC.

5.2.1. Configuración, programación y depuración

La programación y depuración del microcontrolador STM32F401 o BlackPill se realiza mediante el software STM32CubeIDE, el cual presenta muchas facilidades para iniciar el componente, configurar sus relojes internos y periféricos. Para la depuración y comunicación con el PC se usa un STLINK-V2 mini [27]. Este componente usa el protocolo SWD (Serial Wire Debug) para su funcionamiento, protocolo que usa 4 conexiones físicas para su comunicación. Dos de estas conexiones físicas corresponden a sincronización serial (SWCLK) y envío de data (SWDIO). Las otras dos conexiones son para entregar alimentación. Por tanto, se configuraron los periféricos PA13 y PA14 como

Figura 5.2: STLink-v2 mini.

entradas para la conexiones SWCLK y SWDIO. Además, se configuró el reloj de alta velocidad (HSE) a una frecuencia de 84[kHz] en los pines PH0 y PH1. Este reloj es de suma importancia, pues las comunicaciones a los sensores toman particiones de este para su sincronización. Es importante notar que si bien el STLink alimenta el BlackPill con 3.3 [V] este solo se usa para la programación y depuración del componente, la alimentación, una vez todas las conexiones son configuradas y probadas, se realiza mediante el dispositivo FTDI.

5.2.2. Código y Funciones

Se programó un código en C++ para el control y manejo de las señales sensadas, así como para el control de motores. El código se compone por distintas **funciones**, principalmente usadas en **interrupciones** producidas por las comunicaciones seriales y ADC. El main del código del BlackPill presenta las siguientes funciones

sendServoByte: Esta función se encarga de habilitar la transmisión de la conexión UART half-duplex y de enviar un byte de data a través de ella.

getServoByte: Esta función habilita la recepción de la comunicación UART half-duplex y deja al dispositivo a la espera de data mediante DMA.

HAL UARTEx ReceiveToIdle DMA: Función que recibe una cierta cantidad de data mediante DMA. Esto hasta que se recibe toda la data esperada o ocurre un timeout.

HAL UARTEx RxEventCallback: Interrupción de ambas conexiones UART, gatillada cuando finaliza la recepción de data. Aquí se maneja el envío y recepción de data hacia/desde el PC o motores.

HAL ADC Start DMA: Habilita el periférico del ADC y permite las peticiones de data por parte del DMA.

HAL ADC ConvCpltCallback: Interrupción del ADC. Es gatillada una vez se ha finalizado una conversión de algún canal.

5.3. Comunicación con el PC y motores

El controlador se comunica principalmente con el PC y los motores, siendo el primero de estos el que envía paquetes de instrucciones en búsqueda de obtener data de los sensores o monitorear/controlar los motores. Si los motores reciben una orden estos envían un paquete de status, el cual también debe ser manejado por el controlador para ser re-enviado al PC. Por tanto, se usaron las funciones antes descritas para establecer una toma de decisiones cíclica para el controlador, configurando ambas conexiones seriales.

5.3.1. Configuración de comunicación UART full-duplex y half-duplex

Lo primero es implementar la comunicación entre PC y controlador. Para ello se utilizaron las librerías HAL del software STM32CubeIDE, configurando una conexión UART full-duplex (USART2) y UART half-duplex (USART1). Ambas conexiones se establecieron a una velocidad de 115200 [bps], largo de dato de 8 bit , 1 bit de stop y sin paridad. Además, se habilitaron las interrupciones y el DMA. Notar que la velocidad de conexión nominal de los motores fue reducida, por lo que se configuró motor a motor reduciendo sus velocidades de comunicación.

5.3.2. Codificación de toma de decisiones del controlador

Mediante el uso del software STM32CubeIDE se codificó un loop mediante DMA e interrupciones para la toma de decisiones del controlador. El código creado comienza usando la función **HAL UARTEx ReceiveToIdle DMA**, la cual se mantiene a la espera de data mediante DMA hasta que la data recibida complete su transmisión u ocurra un timeout. Cuando termina la recepción de data, se gatilla la interrupción **HAL UARTEx RxEventCallback**, en donde el controlador comienza su toma de decisiones. El controlador debe ver quién envió la data que produjo la interrupción, si el PC o los motores. En caso de ser el PC, chequea el header del paquete de datos recibido, tomando una nueva decisión. Si el header es 0xF0, desglosa el paquete y envía un paquete de vuelta al PC con los valores del sensor pedido. En caso contrario, si el usuario envía un paquete con header 0xFF, el controlador re-envía el paquete hacia los motores. Es importante mencionar que la conexión UART half-duplex no puede enviar y recibir data simultáneamente y por ende, una vez el controlador re-envía el paquete de instrucciones este se queda a la espera del paquete de status enviado por el motor. Una vez se recibe el paquete nuevamente se gatilla la interrupción de la función **HAL UARTEx RxEventCallback**, donde se detecta que el motor envío un paquete de status, el cual se re-envía hacia el PC. Finalmente, el controlador continúa a la espera de datos. En la figura 5.3 se puede ver un diagrama de bloque que explica el comportamiento cíclico de la toma de decisiones del controlador.

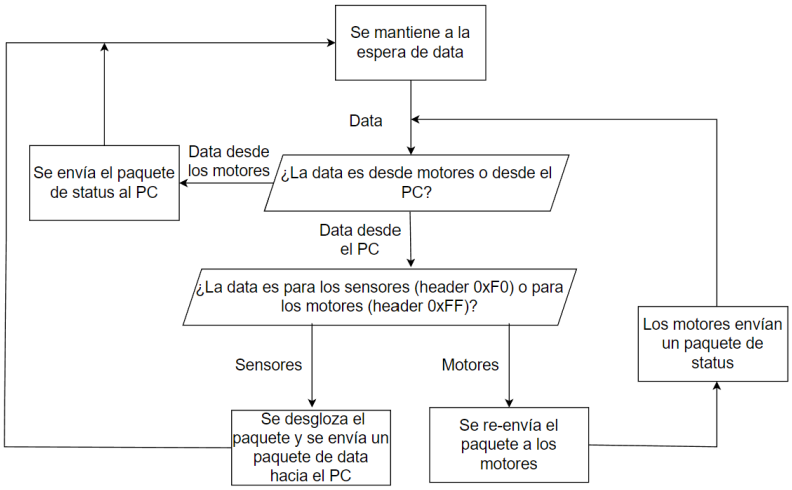

Figura 5.3: Diagrama de bloques de la toma de decisiones del controlador.

5.4. Inicialización, configuración y comunicación con el MPU-6050

La implementación de la detección de movimiento del robot es mediante la conexión serial I^2C con el acelerómetro giroscopio. Se usaron las librerías HAL del software STM32CubeIDE para el manejo de la conexión, siendo las APIs de alto nivel las que se encargan de construir el paquete de datos así como iniciar y finalizar la comunicación con los esclavos. Aquí se configuró la conexión a la velocidad de 400[kHz] y el largo de la dirección de registros a 7-bit, parámetros requeridos para el MPU-6050. Luego, se usaron las librerías de Bulanov Konstantin [28] para la escritura/lectura de los registros del componente.

Lo primero que hace la librería es inicializar el acelerómetro giroscopio, definir la velocidad y escala en la que serán tomados los datos. Para inicializar el dispositivo se lee el registro "WHO AM I"(0x75), en donde el esclavo envía un 0x68 indicando que el componente esta listo para iniciar. Posterior a esto se escribe un 0x00 al registro "PWR MGMT 1"(0x6B), iniciando así el reloj interno del componente. Para definir la velocidad de la toma de datos se escribe 0x07 en el registro "SMPLRT DIV"(0x19), seteando la velocidad a 1[kHZ]. La escala se define escribiendo en los registros "GYRO CONFIG"(0x1B) and "ACCEL CONFIG"(0x1C). Escribiendo un 0x00 se setea una escala de $\pm$ 4g para el acelerómetro y $\pm$ 500 °/s para el giroscopio.

Una vez el componente se ha inicializado y configurado se procede a leer los datos brutos de aceleración y velocidad angular. La información de los todos los datos mencionados se guardan en dos registros por separado, cada uno guardando el bit más y menos significativo. Tomando la aceleración en el eje x como ejemplo, el bit más significativo se guarda en el registro "ACCEL XOUT H" (0x3B) y el menos significativo en el registro "ACCEL XOUT L" (0x3C). La lectura de estos dos registros se combinan en una variable de 16-bits. Este proceso se repite para los registros de aceleración para el eje y,z, además de los registros de velocidad angular para los ejes x,y,z.

Finalmente, se hace una conversión de los valores brutos de aceleración y velocidad angular a gravedad y grados por segundos respectivamente. Para ello se debe obtener la sensibilidad de la medición, que para los parámetros usados son 16384 LSB/g y 131 LSB /°/s.

5.5. Circuito y medición de detección de pisadas

Como fue mencionado en el capítulo 5, la detección de pisadas es realizada por un sensor de luz y un LED. El sensor de luz se comporta como una resistencia cuya resistividad varía según la luz que incida en su superficie. Una simple manera de medir esta variación de resistividad es mediante un divisor de tensión y una resistencia de medición arbitraria. El circuito propuesto se puede observar en la figura 5.4. La medición de tensión vendría dada entonces por

$$V_{out} = V_{cc} \cdot \frac{R_m}{R_{LDR} + R_m}$$

donde V_{out} es la tensión medida, V_{cc} la tensión de alimentación, R_{LDR} y R_m el sensor de luz y la resistencia de medición respectivamente. Este circuito es alimentado por el BlackPill ($V_{cc} = 3{,}3[V]$), la resistencia de medición se eligió arbitrariamente como $R_m = 10[k\Omega]$ y la resisitividad del sensor de luz varía entre $10 - 2[k\Omega]$. Por tanto, el rango posible de tensión medida es de $1{,}65 - 2{,}75[V]$. Análogamente, se creo un circuito simple para el LED, usando una resistencia de 10[kΩ] para conectarlo a la fuente.

La medición de tensión es realizada por el ADC de 12-bit del BlackPill. Similarmente al acelerómetro giroscopio, se usan las librerías HAL para la configuración y manejo del periférico. Aquí, se configuran cuatro canales del ADC a un cuarto del reloj del microcontrolador (21 [kHz]), tomando un tiempo de sampleo de tres ciclos para cada uno. Además, se habilitaron las interrupciones.

Posterior a la medición del ADC se deben convertir los valores de bit a uno de tensión. Como el ADC es de 12-bit, este representa la tensión en 4095 valores posibles, por lo que la conversión será

$$V_{ADC} = ADC_{value} \cdot \frac{3{,}3}{4095}$$

5.6. Placa Perforada

Finalmente, una vez se implementó todo el diseño de las comunicaciones UART, I^2C y las mediciones del ADC se procedió a soldar todos los componentes en una placa perforada PCB. Aquí se conectó todo mediante cables de cobre y se soldaron mediante estaño para asegurar las conexiones. Posteriormente, se midió continuidad entre cables cerciorando que no hubiera conexiones

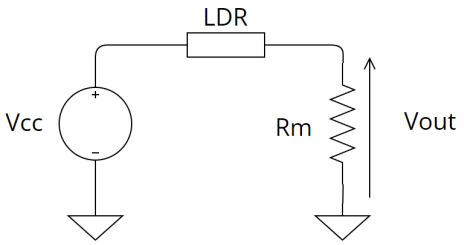

Figura 5.4: Circuito del LDR.

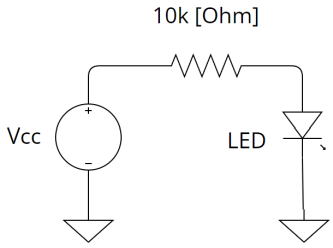

Figura 5.5: Circuito del LED.

indeseadas que pudieran dañar la circuitería.

En los cables de la placa se siguió un código de colores para la visualiza-

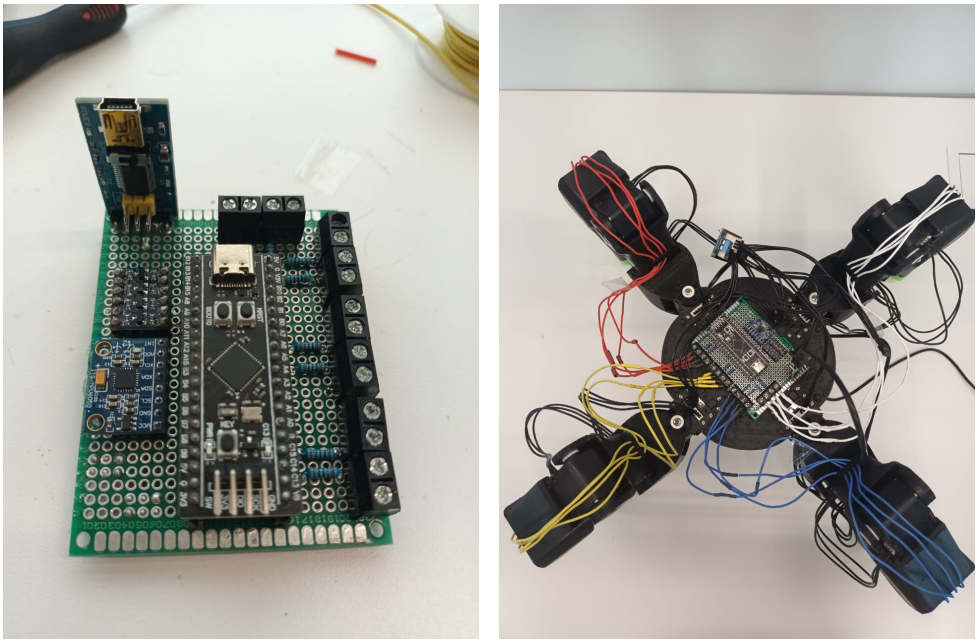

Figura 5.6: Placa perforada PCB con todos los componentes por si sola y ubicada en el robot.

ción de periféricos y fuentes. En particular, la fuente de 3.3[V] fue conectada con cables **rojos**, la fuente de 5[V] con cables **amarillos**, la tierra con cables **azules** y los periféricos con cables **verdes**. Se ubicó la placa perforada en el centro o cuerpo del robot, conectando las cuatro conexiones correspondientes al LED y LDR de cada pierna a ella. También, se conectó un cable USB-mini

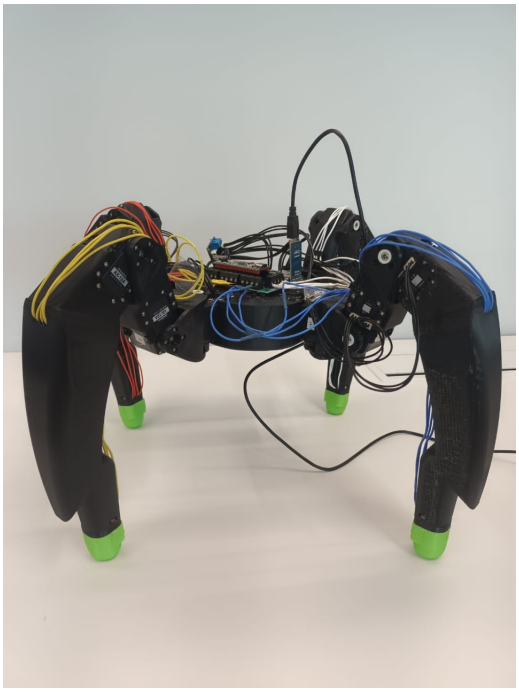

Figura 5.7: Modelo nuevo del ArgoV2.

a USB en el dispositivo FTDI para la conexión con el PC. Además, se interconectaron los motores de cada pierna en paralelo al SMPS2Dynamixel, de modo que solo exista una conexión hacia la alimentación de estos.

Capítulo 6

PRUEBAS Y RESULTADOS OBTENIDOS

6.1. Movimiento y calibración de motores Dynamixel

La primera prueba realizada a la nueva plataforma del ArgoV2 es la calibración y movimiento de motores Dynamixel. Como fue mencionado previamente, las piernas del robot no permiten que los motores de cada extremidad efectúen un movimiento completo de 360 grados, es decir, cada motor tiene un límite angular. Para poder evidenciar esto se realizó un experimento de 45 segundos para cada extremidad, eliminando la limitación angular de cada motor.

El experimento consiste en ubicar los motores en una posición limitada por el modelo 3D físico del robot, haciendo un barrido por todos los ángulos posibles hasta que ocurra otra limitación, obteniendo así el ángulo máximo y mínimo de cada motor para cada extremidad. Entonces, los primeros cinco segundos se mantuvieron los tres motores de la extremidad sin movimiento. Luego, se mueve el motor más cercano al centro en un barrido de ida y vuelta por 10 segundos. Se repite este barrido para los dos siguientes motores con un periodo de cinco segundos sin movimiento entre cada barrido, obteniendo una gráfica para cada extremidad.

De este experimento se desarrolló una tabla con los valores máximos y mínimos de cada motor. Es importante mencionar que cada tres motores hay un cambio de extremidad, información omitida en la tabla. Como se puede observar, los valores máximos de los primeros tres motores de cada extremidad superan

Tabla 6.1: Tabla de ángulos para los motores de cada extremidad.

Motor	Máximo [°]	Mínimo [°]
1	205.278	96.187
2	221.994	77.712
3	240.469	70.087
4	202.932	96.480
5	223.167	75.366
6	240.469	69.208
7	204.105	97.653
8	224.633	78.005
9	241.348	70.381
10	204.692	101.173
11	225.806	77.126
12	241.935	70.087

levemente los límites superiores impuestos en la sección de diseño (195, 220 y 230 grados respectivamente). En cambio, los valores mínimos se mantienen levemente menores o iguales a los limites inferiores (105, 80 y 70 respectivamente). Por tanto, se puede concluir que los límites previamente fijados en el ArgoV2 acotan correctamente los ángulos límite que impone el modelo físico del robot y son mantenenidos para el funcionamiento de éste.

Una vez se obtuvo los ángulos límite de cada motor se procedió a hacer pruebas para el movimiento de las extremidades mediante código. Se hicieron pruebas simples, en donde se cambiaba levemente la posición de los motores consiguiendo un control correcto para cada uno de ellos.

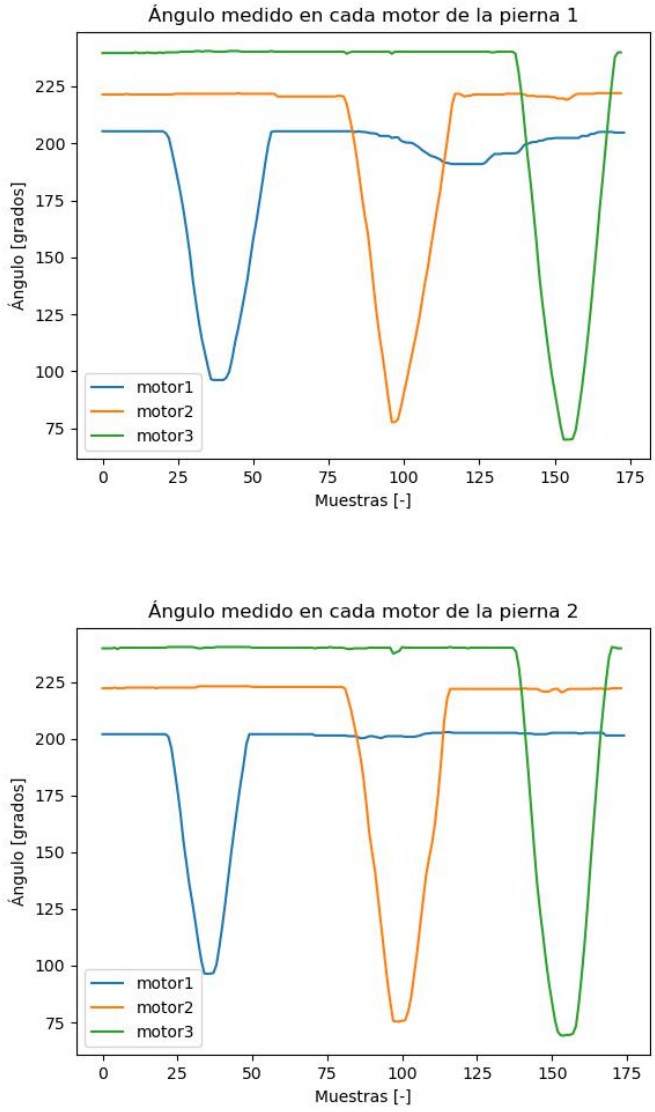

Figura 6.1: Experimento para medir ángulo de los motores para las piernas 1 y 2.

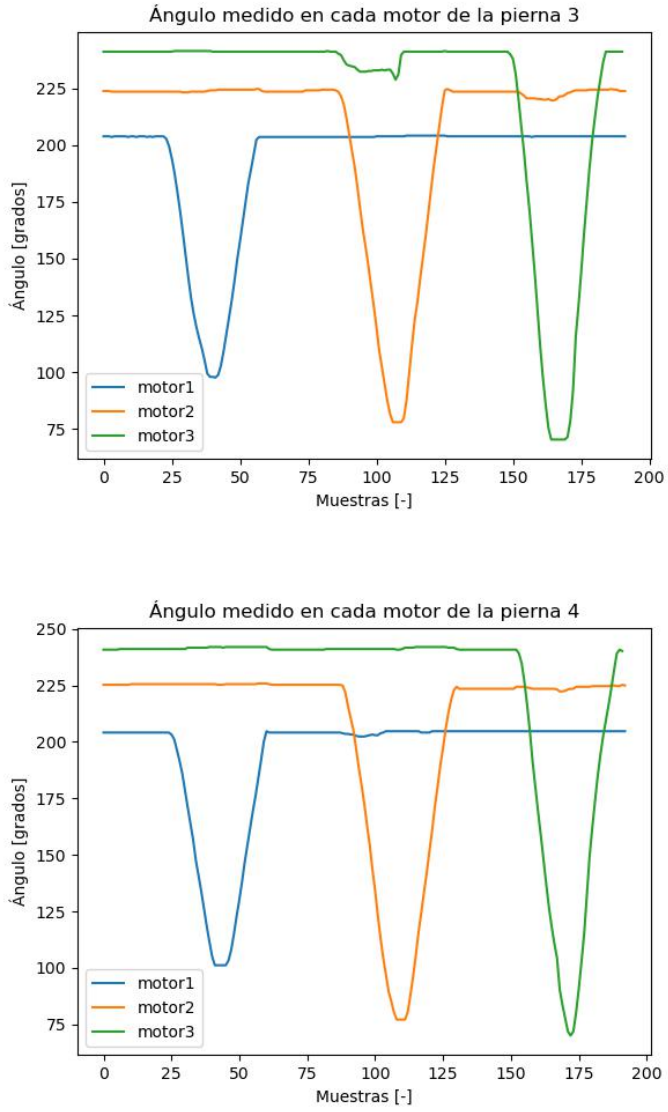

Figura 6.2: Experimento para medir ángulo de los motores para las piernas 3 y 4.

6.2. Medición de gravedad

Una forma simple de comprobar si un acelerómetro funciona correctamente es medir la aceleración de gravedad. El experimento más simple para comprobar esto es orientar la placa recostada en una superficie plana durante la toma de datos, donde se debería medir la gravedad en el eje z. De esta manera también se puede calibrar el acelerómetro, tomando los valores medidos y comparandolos con los valores teóricos de aceleración que debería tener cada eje. Por lo tanto, se hizo un experimento de 25 segundos midiendo la aceleración de todos los ejes con la placa recostada y sin moverse.

De este experimento se obtuvo la media y la varianza de aceleración en cada

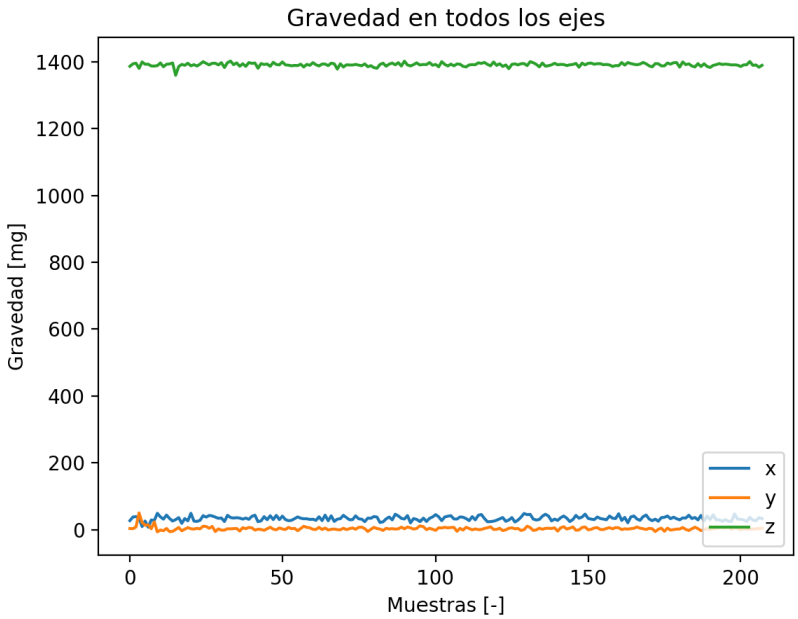

Figura 6.3: Gráfico de aceleración en [mg] para el primer experimento.

eje durante los 25 segundos, así como el valor máximo, mínimo y la máxima variación de [mg]. De aquí se puede concluir que la medición de cada eje es consistente a lo largo del tiempo, pues todos presentan varianzas bajas.

Como se puede ver, los valores experimentales de cada eje difieren de los valores teóricos que debería medir el acelerómetro (0, 0, 1000 respectivamente), existe un offset. Es de interés conocer el offset que presenta cada eje para extraer los valores reales de aceleración en cualquier contexto, calibrando así el

acelerómetro. Entonces, se probó la variación de aceleración en todos los ejes. Para ello se llevaron a cabo dos experimentos de toma de datos de 50 segundos cada uno, manteniendo los ejes x e y fijos. Se giró la placa perforada en 90 grados cada 10 segundos, dando una vuelta completa para observar como cambiaba la gravedad entre ejes. Se pudo observar que la medición de aceleración de gravedad era consistente con los movimientos realizados. Además, se evidenció que los valores de cada eje presentan offset, siendo 33,44, 3.180 y 391.368 para los ejes x, y, z respectivamente.

Tabla 6.2: Tabla de estadísiticas del acelerómetro.

Eje	Media [mg]	Varianza [mg^2]	Máximo [mg]	Mínimo [mg]	Máxima Variación [mg]
X	33.440	6.945	48.950	7.202	41.748
Y	3.180	5.246	50.170	-6.713	56.884
Z	1391.368	5.022	1402.221	1359.619	42.602

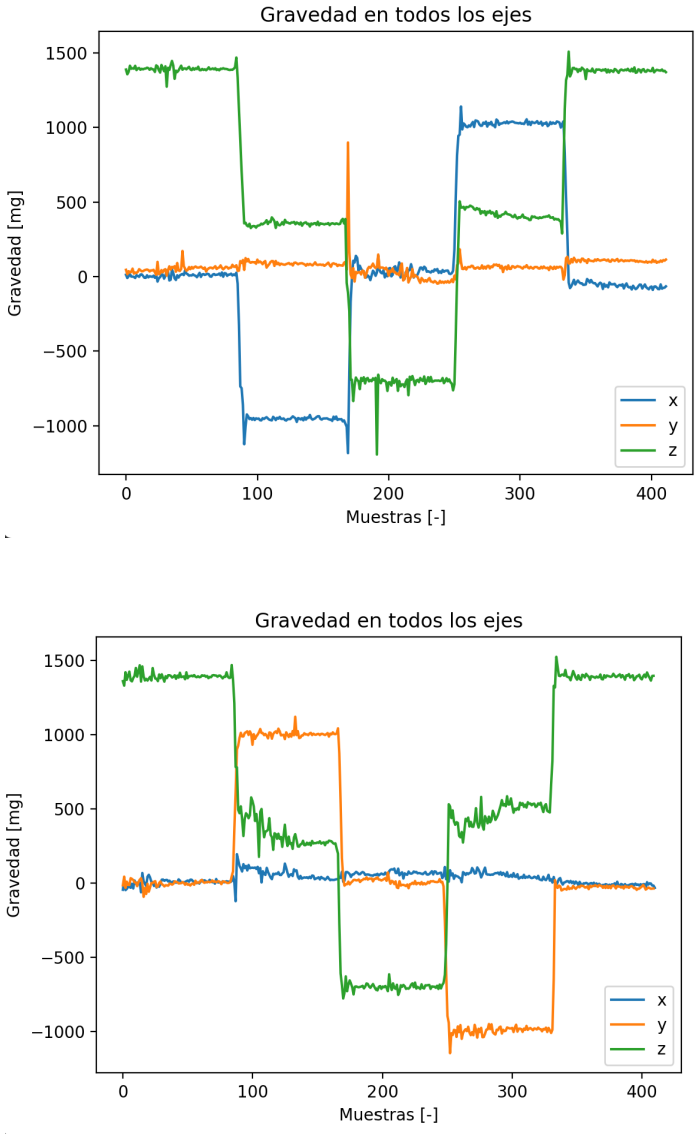

Figura 6.4: Aceleración manteniendo los ejes x e y fijos.

6.3. Medición de velocidad angular

Se experimetó con la placa perforada para comprobar que el giroscopio midiera correctamente la velocidad angular. Se ubicó la placa en una silla de oficina común, girando esta última sobre su propio eje para que el giroscopio experimentara una variación de velocidad angular. En particular, el

experimento consistía de cuatro etapas. La primera etapa fueron 10 segundos sin movimiento alguno en la placa. Pasados los primeros diez segundos se giró la silla en 360 grados dos veces, en sentido horario y antihorario. Luego se repite la primera etapa, para terminar con dos giros de 360 grados en sentido antihorario y horario. Esto se repitió cambiando la orientación de la placa tres veces, de modo que esta velocidad angular producida por la silla fuera percibida por un eje distinto cada vez. Luego, se realizó el otro experimento de 25 segundos , de donde no se sometió al giroscopio a ningún movimiento.

En el primer experimento se puede ver que el componente funciona perfectamente, ya que tanto para los giros horarios y antihorarios hay una variación de velocidad angular coherente con los movimientos, esto para los tres ejes. Es importante notar que hay mínimas variaciones de velocidad en los ejes que se mantuvieron fijos, esto debido a errores de medición. La presencia de un pequeño ángulo en la orientación de la placa produce que los otros ejes sufran del giro de la silla. Durante el segundo experimento se calcularon las mismas estadísticas que para el acelerómetro, las cuales se pueden ver en la tabla 6.4. De aquí se puede concluir que los valores medidos son consistentes a lo largo del tiempo, presentando varianzas muy bajas. Análogamente al acelerómetro, cada eje tiene la presencia de offset. Este offset es usado para la calibración del giroscopio, siendo los valores -13.725, 2.728 y -17.676 para los ejes x, y, z respectivamente.

Tabla 6.3: Tabla de estadísticas del giroscopio.

Eje	Media [°/s]	Varianza [°/s^2]	Máximo [°/s]	Mínimo [°/s]	Máxima Variación [°/s]
X	-13.725	0.055	-13.458	-13.870	0.411
Y	2.728	0.052	2.899	2.593	0.305
Z	-17.676	0.046	-17.532	-17.822	0.289

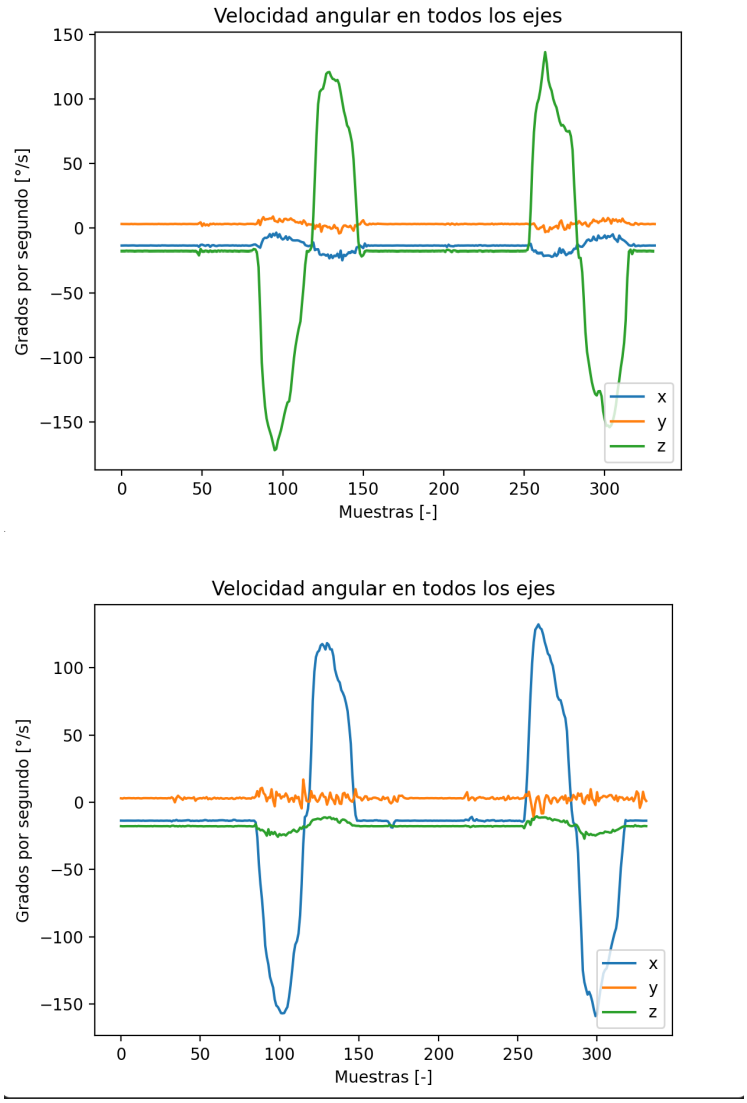

Figura 6.5: Velocidad angular durante el experimento de la silla para el eje x y z.

6.4. Implementación del Filtro de Kalman

En la sección de sensorización del ArgoV2 se diseñaron las ecuaciones generales para aplicar el filtro de Kalman a los valores de aceleración y velocidad angular para la adquisición de la velocidad, distancia y ángulos de inclinación del robot. Se programó el algoritmo en un código de Python definiendo todas

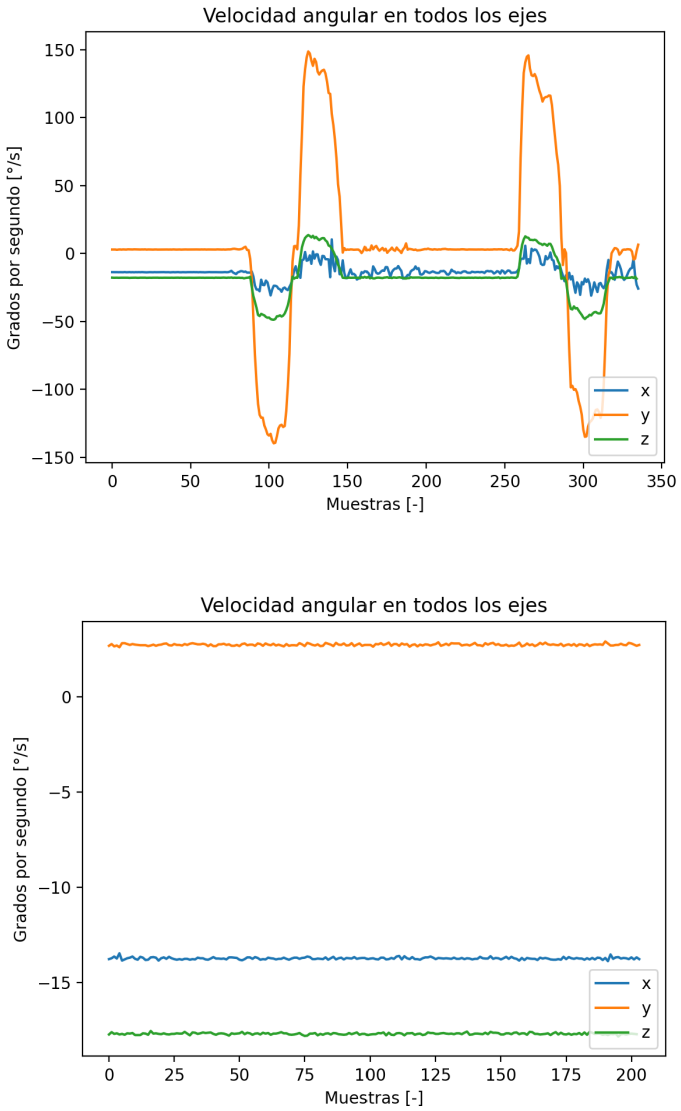

Figura 6.6: Velocidad angular durante el experimento de la silla para el eje y, y para el experimento sin movimiento.

las matrices necesarias para la predicción y ajuste del estado del sistema. Se definió la covarianza del ruido como 0.1, la covarianza de la medición como 1 y la covarianza del error inicial como 1. El tiempo de la toma de datos es de 0.125 segundos. Con estos valores se procedió a usar el algoritmo para el acelerómetro y giroscopio.

Para el acelerómetro, se tomó la data sensada para su calibración, tomando el eje z. Se calibró la data restando el offset calculado previamente y se transformaron los datos de [mg] a $[m/s^2]$. Este eje percibe la aceleración de gravedad, por lo que se supuso que esta aceleración correspondía a una caída libre sin roce y se usó el algoritmo de Kalman para graficar la velocidad y distancia recorrida durante los 25 segundos de experimento. Además, se realizó un gráfico de comparación entre la aceleración medida por el acelerómetro y la estimada por el algoritmo.

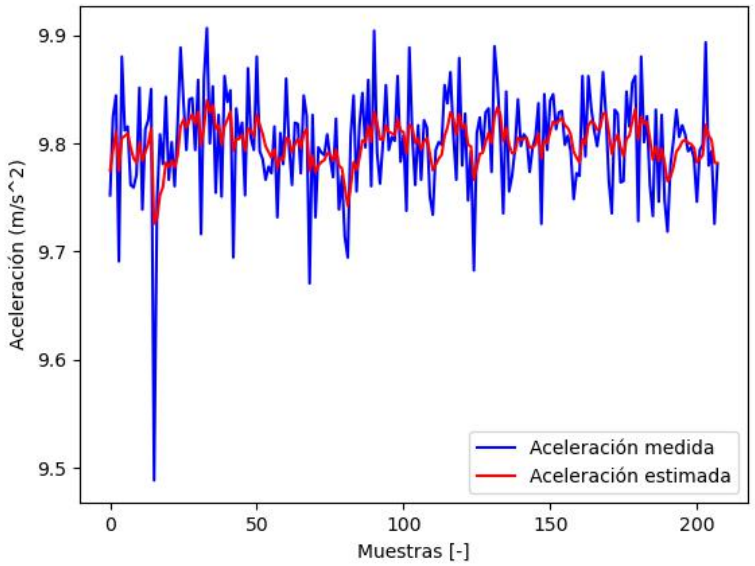

Figura 6.7: Gráfico de comparación de aceleración medida y estimada.

Como se puede ver, la comparación de aceleraciones evidencia que la aceleración estimada se mantiene más estable durante el tiempo y carece de grandes oscilaciones. También, la distancia y velocidad obtenidos por el algoritmo coinciden con los valores teóricos, además de mostrar una forma de onda correspondiente a una velocidad creciente y una distancia cuadrática durante el tiempo.

Para el giroscopio se hizo un nuevo experimento. Este consistía en mantener el MPU-6050 en reposo con el eje z detectando la gravedad durante cinco segundos. Luego, se rotaba este en 90 grados en sentido antihorario, manteniendo esta orientación por otros cinco segundos. Este proceso de repitió una vez, finalizando con cinco segundos en estado de reposo. Se midió la acelera-

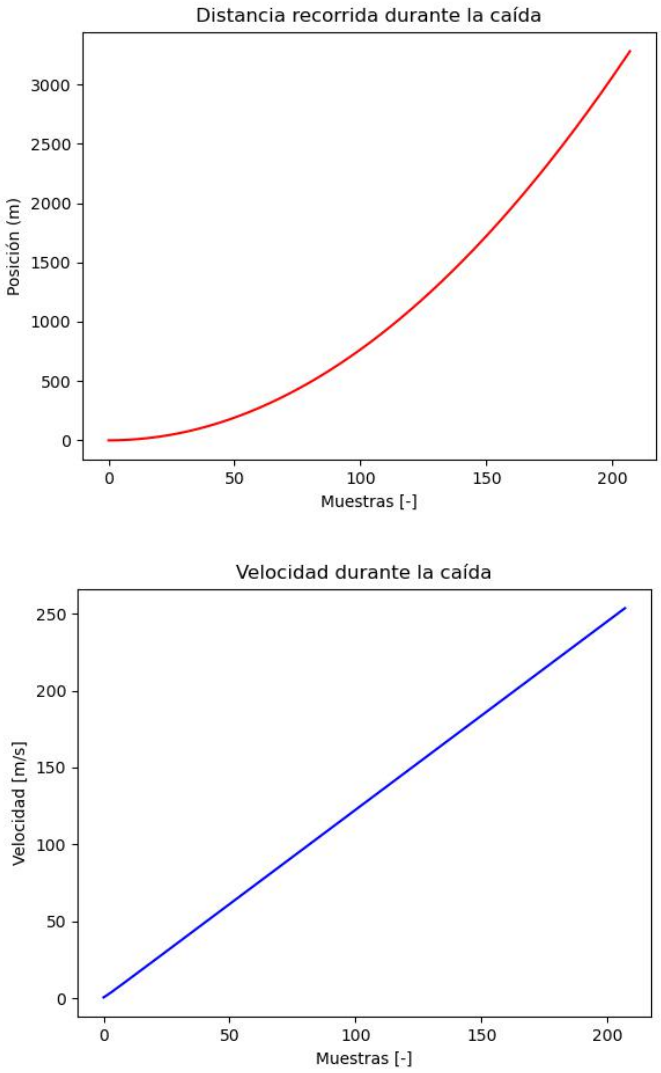

Figura 6.8: Distancia recorrida y velocidad durante el experimento.

ción en los tres ejes juntos calculando el ángulo de inclinación con relaciones trigonométricas. Además, se midió la velocidad angular del eje x. Con estos datos se implementó el filtro de Kalman. La comparación del valor medido y estimado se puede ver en la siguiente figura.

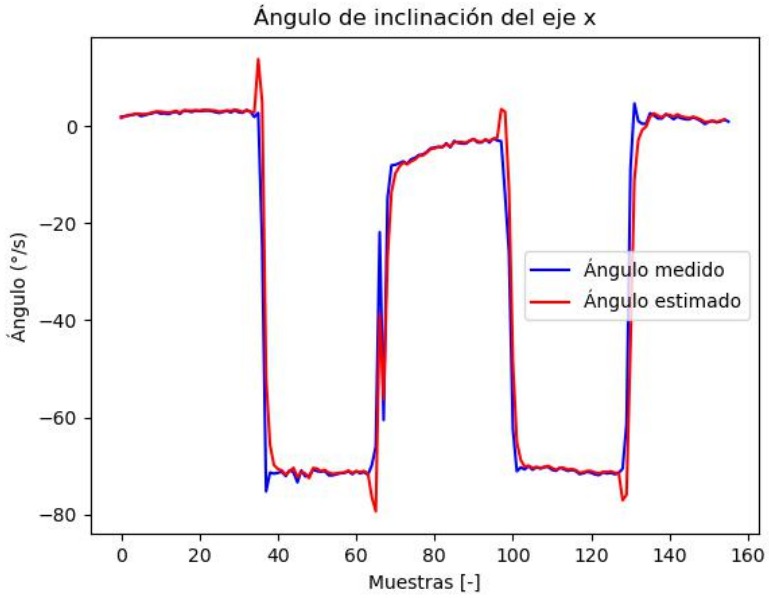

Figura 6.9: Gráfico de comparación el ángulo de inclinación medido y estimado.

6.5. Calibración del ADC

Antes de realizar alguna prueba con los sensores de la detección de pisadas es importante hacer una calibración de los cuatro canales del ADC. Para ello, se tomaron datos durante 25 segundos y se obtuvieron estadísticas de cada canal, sin conectar los sensores LDR al periféricos del ADC. De aquí podemos concluir que los canales presentan offset y que no presentan oscilaciones muy grandes debido a que presentan varianzas bajas. Similarmente a los casos anteriores, se obtuvo el offset es usado para la calibración.

Tabla 6.4: Tabla de estadísticas de los canales del ADC.

Canal	Media [V]	Varianza [V^2]	Máximo [V]	Mínimo [V]	Máxima Variación [V]
1	0.1909	0.001901	0.1950	0.1877	0.007252
2	0.495	0.003733	0.5044	0.4867	0.01772
3	0.1478	0.002699	0.1563	0.1418	0.01450
4	0.00204	0.002043	0.08703	0.07736	0.009670

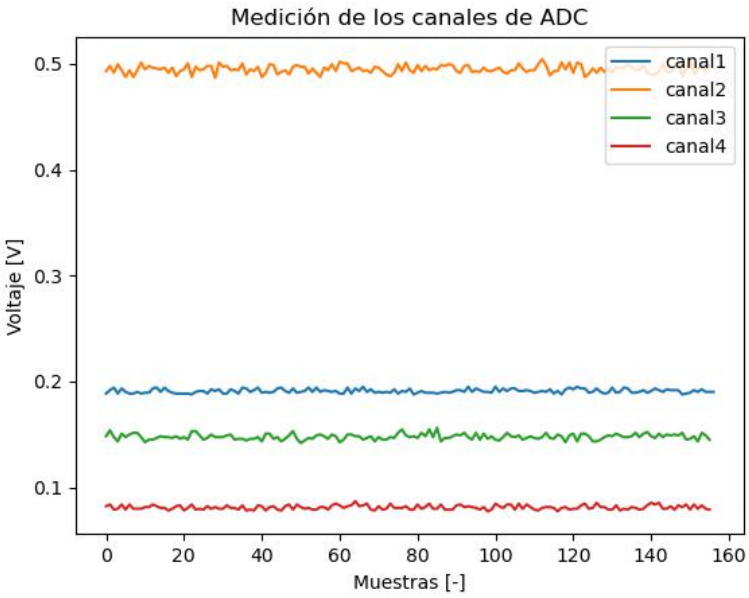

Figura 6.10: Gráfico de los cuatro canales del ADC sin sensores conectados.

6.6. Tensión en el pie flexible

Obtener el rango efectivo en el que varía la medición del LDR es crucial para determinar que tan bueno fue la implementación del diseño del pie flexible. Por tanto se midieron las variaciones de tensión máximas que presenta cada extremidad del robot. Se realizó un experimento manteniendo la extremidad suspendida en el aire sin ningún tipo de presión al pie flexible, en donde se inicio la toma de datos durante 60 segundos. Luego, se ejerció una fuerza al pie flexible que lo deformara comprimiendo el resorte al máximo posible, transicionando entre un estado sin y con presión cada 10 segundos. Este proceso se repitió con todas las extremidades. Los gráficos de tensión se pueden observar en las figuras 6.11 y 6.12. De estos datos se extrajeron la tensión máxima y mínima medida, calculando así la máxima variación de tensión para cada pie flexible de cada extremidad.

Tabla 6.5: Tabla de las variaciones de voltaje de cada pierna del ArgoV2.

Tensión [V]	Pierna 1	Pierna 2	Pierna 3	Pierna 4
Máximo [V]	2.5908	2.2475	1.9002	2.5118
Mínimo [V]	2.3144	1.9542	1.6141	2.2684
Variación [V]	0.2764	0.2933	0.2860	0.24336

6.7. Tiempos de Respuesta

El BlackPill se encarga del envío y recepción de data tanto de los sensores de las detecciones de pisadas y movimiento como de las comunicaciones con los motores y el PC. Dependiendo de que dato es pedido por el usuario el paquete de bytes hace un recorrido distinto por el controlador, siendo el más largo de estos cuando se envían datos al motor. Es importante poder medir cuanto es el tiempo que se demoran los datos en ser recibidos por el PC una vez es enviado un paquete de bytes al controlador, esto para todos los recorridos posibles. Por ende, se hizo un experimento de toma de datos para cada motor y sensor del ArgoV2, midiendo los tiempos de recepción de la data enviada por el controlador mediante un código en Python. Se tomaron 100 datos para cada caso, calculando el tiempo máximo, mínimo, la media y la varianza constuyendo la siguiente tabla. Como se puede ver, los tiempos de respuesta no varían significativamente entre motores/sensores, manteniendo valores similares en sus medias y presentando varianzas bajas.

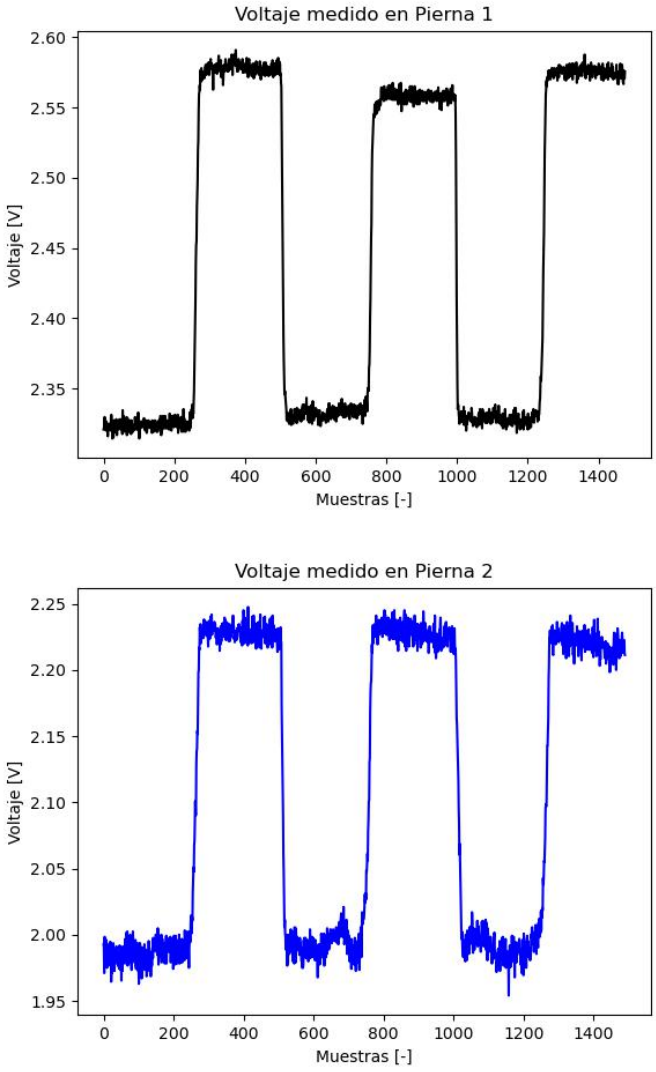

Figura 6.11: Voltaje medido en la pierna 1 y 2.

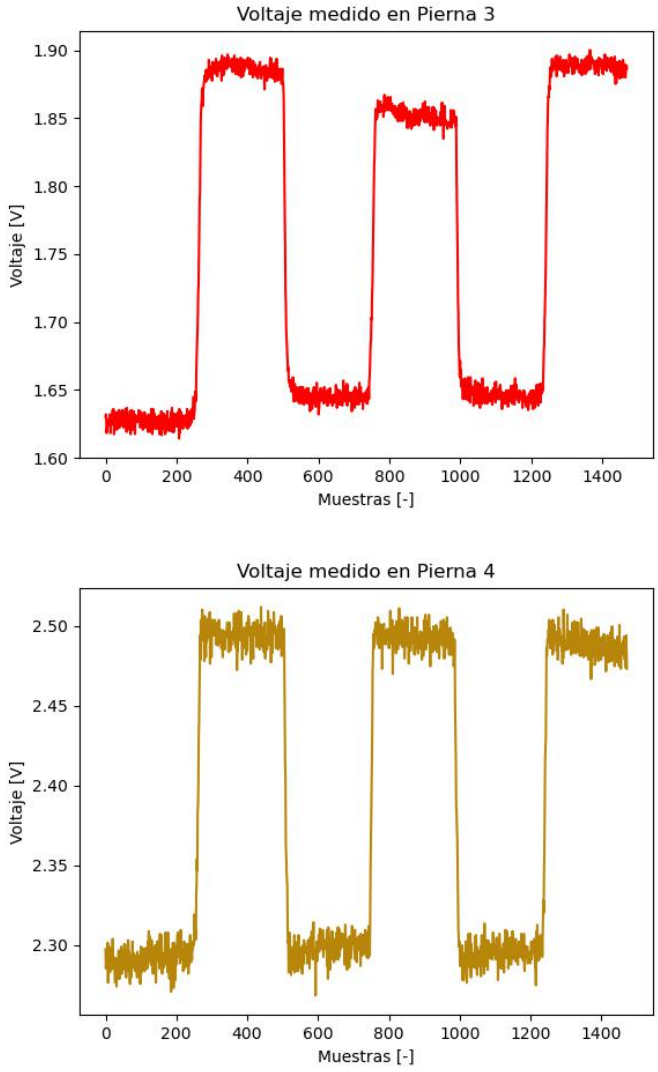

Figura 6.12: Voltaje medido en la pierna 3 y 4.

Tabla 6.6: Tabla de tiempos de respuesta para cada motor/sensor

Motor/Sensor	Máximo [s]	Mínimo [s]	Media [s]	Varianza [s^2]
Motor 1	0.04463	0.02571	0.03464	0.004450
Motor 2	0.04397	0.02987	0.03514	0.004282
Motor 3	0.04336	0.02518	0.03453	0.004607
Motor 4	0.04384	0.02888	0.03525	0.004331
Motor 5	0.04405	0.02607	0.03527	0.004175
Motor 6	0.04402	0.02636	0.03582	0.004795
Motor 7	0.04528	0.02576	0.03580	0.004838
Motor 8	0.04425	0.02636	0.03610	0.004787
Motor 9	0.04480	0.02552	0.03462	0.004488
Motor 10	0.04301	0.02580	0.03460	0.004275
Motor 11	0.04436	0.02588	0.03550	0.004536
Motor 12	0.04552	0.02543	0.03625	0.004746
Acelerómetro eje x	0.04588	0.02633	0.03448	0.004457
Acelerómetro eje y	0.04587	0.02500	0.03520	0.004879
Acelerómetro eje z	0.04636	0.02762	0.03539	0.004173
Giroscópio eje x	0.04424	0.02504	0.03488	0.004093
Giroscópio eje y	0.04386	0.02676	0.03582	0.004696
Giroscópio eje z	0.04422	0.02747	0.03487	0.004282

Capítulo 7

CONCLUSIONES Y TRABAJO A FUTURO

7.1. Conclusiones

En este trabajo de memoria se ha obtenido una plataforma robótica de extremidades móviles en donde se puede experimentar el aprendizaje de su caminata con nueva información del enterno y estado en el que se encuentra. Esto se logró gracias a la implementación de un sistema de detección de pisadas bio-inspirado en la extremidad de un insecto, un acelerómetro giroscopio encargado de la detección de movimiento, y un controlador STM32F4 para el manejo de la data sensada y la inclusión del sistema antiguo del robot. A partir de los resultados obtenidos se pueden determinar las siguientes conclusiones

1. La implementación de un pie flexible para la detección de pisadas fue exitosa, pues se obtuvieron cuatro señales continuas representando la presión ejercida en cada extremidad del robot en un rango de voltaje medible. Así se discriminó si la extremidad estaba en contacto con el piso y se cuantificó la fuerza efectuada hacia el robot debido a la pisada. Se calcularon estadísticas de los canales del ADC encargados de manejar esta señal continua, donde se calibró y probó el correcto funcionamiento del sistema mecánico.

2. En la detección de movimiento logró describir precisamente la aceleración y velocidad angular experimentadas por el robot. Se hicieron pruebas comprobando la veracidad de los datos sensados, obteniendo estadísticas para la calibración del componente. Se implementó y probó el filtro de

Kalman para la adquisición de la velocidad y distancia reduciendo el error acumulado que se presenta en la integración de la aceleración. Similarmente, se encontró una relación geométrica entre los valores vectoriales de aceleración y el ángulo de inclinación del robot, aplicando nuevamente el filtro de Kalman para obtener valores fiables.

3. El controlador unificó la data sensada junto con el control y manipulación de las extremidades del robot, condesando todo en un sólo protocolo serial. Se probarón y configuraron las conexiones seriales encargadas de la transmisión de datos del robot hacia el PC y motores donde se midieron los tiempos de envío de data para todos los sensores integrados.

7.2. Trabajo Futuro

Si bien este trabajo de memoria logró construir una plataforma robótica que incluyera la data sensada de dos sistemas de detección de pisadas y movimiento, existen una variedad de mejoras o trabajos a futuro para afinar el funcionamiento del robot. Dentro de estas posibilidades se pueden mencionar

1. El diseño circuital de los componentes del robot en una placa impresa PCB. Así se otorgaría mayor robustez al sistema implementado, reduciendo el porcentaje de error producido por el corte de cables o perturbaciones físicas al circuito.

2. Implementación de filtro pasa-bajo circuital para señales de detección de pisadas. Esto mejoraría la data recibida por el usuario de los sensores LDR, presentando menos oscilación de la señal recibida para su análisis.

3. Configuraciones al sistema mecánico de la extremidad, como puede ser un rodamiento lineal o cambio de sensores. Una mejora al sistema mecánico entregaría una señal continua más fiable y mayor presición en la medición de una fuerza ejercida en las extremidades.

4. Reemplazo de conexión con el PC. Otras opciones se presentan cuando se habla de una conexión serial entre PC y controlador. Aquí existen muchas variaciones, como usar el USBc del mismo microcontrolador, un modulo Bluetooth o un simple cambio de conexión USB-USBmini a USB-USBc.

5. Mejora de motores Dynamixel por alta-gamma. La empresa Dynamixel presenta una alta-gamma de servo-motores inteligentes, la series MX. Cambiar a un modelo más actual presentaría grandes opciones a la

movilidad del robot: podría soportar cargas más grandes, realizar movimientos de más de 360 grados, tener velocidades de movimiento mayores además de presentar mayor resolución de data.

6. Adición de más sensores. La detección de pisadas y movimiento no son los únicos métodos que nos entregan información relevante del entorno y estado del robot. Dentro de la información que podría ayudar al robot estaría: una piel táctil que permita detectar el contacto con obstáculos que perturben el movimiento, un GPS que permita entender la espacialidad del robot en el entorno, o cámaras para realizar procesamiento de imágenes.

BIBLIOGRAFÍA

[1] O. S. et al, "Time delays in a hyperneat network to improve gait learning for legged robots," 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7966390"

[2] P. R. et al, "Neuroevolutive algorithms for learning gaits in legged robots," 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8852635

[3] A. M. A. et al, "Pressure sensor: State of the art, design, and application for robotic hand," 2015. [Online]. Available: http://dx.doi.org/10.1155/2015/846487

[4] T. K. M. et al, "Walking robot movement on non-smooth surface controlled by pressure sensor," 2018. [Online]. Available: http://dx.doi.org/10.5185/amlett.2018.1878

[5] C. K. et al, "Rigidity-based surface recognition for a domestic legged robot," 2016. [Online]. Available: https://doi.org/10.1109/LRA.2016.2519949

[6] D. V. et al, "Learning from accelerometer data on a legged robot," 2004. [Online]. Available: https://doi.org/10.1016/S1474-6670(17)32082-7

[7] M. Q. et al, "Low-cost accelerometers for robotic manipulator perception," 2010. [Online]. Available: https://doi.org/10.1109/IROS.2010.5649804

[8] M. A. H. et al, "Haptic terrain classification for legged robots," 2010. [Online]. Available: https://doi.org/10.1109/ROBOT.2010.5509309

[9] J. C. et al, "Acoustics based terrain classification for legged robots," 2016. [Online]. Available: https://doi.org/10.1109/ICRA.2016.7487543

[10] Y. X. et al, "A flexible multimodal sole sensor for legged robot sensing complex ground information during locomotion," 2021. [Online]. Available: https://doi.org/10.3390/s21165359

[11] R. e manual, "Dynamixel ax-12a." [Online]. Available: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

[12] Dynamixel, "Smps2dynamixel overview." [Online]. Available: https://www.robotis.us/smps2dynamixel/

[13] ——, "Usb2dynamixel e-manual." [Online]. Available: https://emanual.robotis.com/docs/en/parts/interface/usb2dynamixel/

[14] ——, "Dynamixel protocol 1.0." [Online]. Available: https://emanual.robotis.com/docs/en/dxl/protocol1/

[15] D. SDK, "Dynamixel library," https://https://github.com/ROBOTIS-GIT/DynamixelSDK.

[16] I. Inc, "Mpu-6000 and mpu-6050 product specification," Revision 3.4. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[17] ——, "Mpu-6000 and mpu-6050 register map," Revision 4.2. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[18] Creality, "Ender-3 v2 3d printer user manual." [Online]. Available: https://manuals.plus/wp-content/uploads/2020/12/Creality-Ender-3d-Printer-User-Manual.pdf

[19] CarbonAeronautics., "Part xiv: Measure angles with the accelerometer." [Online]. Available: https://github.com/CarbonAeronautics?tab=repositories

[20] ElecFreaks, "Ultrasonic ranging module hc - sr04." [Online]. Available: https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf

[21] I. Electronics, "Force sensing resistor integration guide and evaluation part catalog," Version 1.0. [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/010/126/original/fsrguide.pdf

[22] L. Z. et al, "Design and research of a flexible foot for a multi-foot bionic robot," 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/17/3451

[23] Honeywell, "Fss-smt series - low profile force sensor." [Online]. Available: https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/siot/ja/products/sensors/force-sensors/fss-smt-series/documents/sps-siot-force-sensors-fss-smt-product-sheet-008181-3-en-ciid-44994.pdf

[24] ST, "Stm32f401xc stm32f401xe datasheet," Revision 7. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f411re.pdf

[25] ——, "Description of stm32f4 hal and low-layer drivers," 2020. [Online]. Available: https://www.st.com/resource/en/user$_manual/um1725 - description - of - stm32f4 - hal - and - lowlayer - drivers - stmicroelectronics.pdf$

[26] F. T. D. International, "Ft232r usb uart ic datasheet," 2020. [Online]. Available: https://ftdichip.com/wp-content/uploads/2020/08/DS$_FT232R.pdf$

[27] ST, "Um1075 user manual." [Online]. Available: https://www.st.com/resource/en/user$_manual/um1075-stlinkv2-incircuit-debuggerprogrammer - for - stm8 - and - stm32 - stmicroelectronics.pdf$

[28] B. Konstantin, "Stm32 hal library for gy-521 (mpu6050) with kalman filter." [Online]. Available: https://github.com/leech001/MPU6050