**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**

# Controladores difusos para la postura y marcha de un robot bípedo de 12 GDL

**TESIS**

Que para obtener el título de

**Ingeniero Mecatrónico**

**P R E S E N T A**

Allen Eduardo Sánchez Ortega

**DIRECTOR DE TESIS**

Dr. Edmundo Gabriel Rocha Cózatl

**Ciudad Universitaria, Cd. Mx., 2017**

*Dedicado a mis abuelitos*
*Memo y Jóse quienes han sido*
*mis mejores maestros*

# Agradecimientos

A a mis padres **Edgar** y **Virginia**, por todo el amor, atención y apoyo incondicional brindado durante mi vida.

A a mis cuatro abuelos, por la sabiduría, amor y apoyo brindado.

A a mi novia **Angie** ♡ por darme su amor y cariño incondicional.

A a mis amigos, por compartir inolvidables momentos y por sus enseñanzas, **Alan, José, Felipe, Hector, Alejandro Oviedo, Alejandro Avendaño, Jhovan, Pablo y Abner.**

Al **Dr. Edmundo** y a la **Ing. Fernanda** por su orientación y dedicación para hacer posible este trabajo.

Al apoyo otorgado por el programa PAPIIT IN1139: ”Planeación de trayectorias y control de marcha de un robot bípedo”.

# Índice general

# Índice de figuras

# Índice de tablas

# Abstract

In this work a method for posture and gait control using fuzzy logic controllers in a biped robot type Scout is presented. For these purposes an embeded system has been developed for the robot in order to implement control laws.

The present work results from previous thesis in which, they have developed several analisys for the biped robot, as: instrumentation, cinematics, dynamics, and gait trajectories.

The studies and analysis gathered around locomotion stability lead to two branches, dynamical stablilization using Zero Moment Point (ZMP) as first criteria for the gait, and dynamical stabilizatiom using the center of mass (COM). In this thesis COM stabilization will be used, and using ZMP criteria the gait of the robot will be prove as stable during a long gait cycle.

For posture control, there are two different controllers, a Self Tuning Fuzzy Logic PID (STFL-PID) controller, and a Fuzzy Logic Posture (FLP) controller. In STFL-PID control the COM will be stabilize at the horizontal reference of zero, in the sagittal plane (pitch direction) and coronal plane (roll direction). The Self Tuning task of the PID control is done by a Mamdani-type fuzzy controller, it is based on knowledge rules using error, sum error, and error change behavior to obtain desired Kp, Ki, Kd constants. The FLP control takes place in the coronal plane (roll direction), and it command the distance between the hip and the ankle of each leg. These distances are reduced, or extend in pitch direction given the error behavior towards roll direction.

For gait controller only STFL-PID control in pitch and roll direction is used. the additional controller for the distance between the hip and the ankle is discard because during the trajectorie gait application all angles from the joints are modified, and it would be deform if used.

In the 6th chapter an analisys between STFL-PID controller and a traditional PID controller is presented. The prevoius test highlights the performance in settle time of the STFL-PID versus PID. Besides STFL-PID showed up a robust behavior under stress and perturbations.

As future work, a new scheme of control is proposed, by adding ZMP criterium and magnetometer measure to control laws.

# Resumen

En este trabajo se propone el control de postura y marcha de un robot bípedo tipo Scout basados en lógica difusa. Para estos propósitos, se ha desarrollado un sistema embebido para el robot con el fin de implementar las leyes de control de dichos controladores.

Esta tesis es el resultado de trabajos anteriores en los se han efectuado varios análisis para el robot bípedo, como: instrumentación, cinemática, dinámica y trayectorias de marcha.

Los estudios y análisis reunidos alrededor de la estabilidad de la locomoción bípeda conducen a dos ramas, la estabilización dinámica utilizando el punto de momento cero (ZMP, por sus siglas en inglés) como primer criterio para la marcha, y la estabilización dinámica utilizando el centro de masa (COM, por sus siglas en inglés). En esta tesis se utilizará la estabilización del COM y usando el ZMP como criterio de estabilidad, la marcha del robot se demostrará estable en un ciclo de marcha largo.

Para el control de la postura existen dos controladores diferentes, uno llamado control PID autosintonizable mediante lógica difusa (PID-ASLD), y el otro control difuso de postura (CDP). En el control PID-ASLD el COM se estabilizará en la referencia horizontal de cero, en el plano sagital (dirección Pitch) y coronal (dirección Roll). La autosintonización del control PID se hace utilizando un controlador difuso tipo Mamdani, este sintonizador tiene como base de conocimientos reglas basadas en el comportamiento del error, de la suma del error y el cambio del error para obtener las constantes, Kp, Ki y Kd deseadas. El CDP tiene lugar en el plano coronal(dirección Roll) y controla la distancia entre la cadera y el tobillo de cada pierna del robot. Estas distancias se reducen o se extienden en dirección Pitch según el error en la dirección Roll.

Para el control de marcha se utiliza el control PID-ALSD en la dirección Pitch y Roll. El CDP se descarta durante la caminata, porque durante ésta se modifican todos los ángulos de las articulaciones creando un corriemiento (offset) no deseado.

En el Capítulo 6 se presenta un análisis entre el controlador PID-ALSD y un controlador PID tradicional usados para el control de postura. El experimento realizado pone de manifiesto que el desempeño en el tiempo de asentamiento del control PID-ASLD en comparación con un control PID tradicional es mejor. Además el control PID-ASLD mostró un comportamiento robusto bajo perturbaciones (cambios de pendiente).

Finalmente, como trabajo futuro, se propone un esquema de control añadiendo el criterio del ZMP y un magnetómetro, para que con las lecturas generadas se añada información a la ley de control.

# Estructura del trabajo

**Capítulo 1. Introducción**

Se da una breve reseña del estado del arte de los robots bípedos, de la lógica difusa y controladores PID autosintonizables mediante lógica difusa. Posteriormente, se hace el planteamiento del problema, la hipótesis y justificación del presente trabajo. Enseguida, se presentan los objetivos, alcances y limitaciones.

**Capitulo 2. Preliminares**

Se exponen los conceptos teóricos utilizados en esta tesis. Las primeras dos secciones están dirigidas a la locomoción bípeda y su estabilidad. En la tercera sección se da una breve descripción de la lógica difusa, se introduce el concepto de razonamiento difuso o aproximado y el modus ponens difuso. Finalmente se abordan los conceptos básicos de control, para después dar lugar, a la descripción de los controladores tipo Mamdani y PID autosintonizable.

**Capitulo 3. Descripción del robot bípedo Scout**

Se describen las características físicas y electrónicas del robot bípedo Scout. Se reporta la instrumentación actual del robot y se mencionan las mejoras que se requieren en el desarrollo del sistema de control del robot.

**Capitulo 4. Diseño de los controladores difusos**

Se establece el esquema de control propuesto, se muestra la configuración de los controladores difusos para la postura y marcha del robot bípedo Scout. En primera instancia, se presenta el control de postura en dirección pitch y roll, y su diseño mediante lógica difusa. Finalmente, basado en el control de postura se muestra el control de marcha, y su configuración.

**Capitulo 5. Sistema embebido**

Se aborda el diseño e implementación del sistema electrónico para el robot bípedo Scout. Se muestran las características de los microcontroladores y la microcomputadora utilizados, así como las particularidades de los protocolos de comunicación entre estos elementos. Se describe de manera breve el funcionamiento de la interfaz gráfica. Por último se desarrolla la implementación del esquema de control establecido en el Capitulo 4.

**Capitulo 6. Análisis de resultados**

Se presentan los resultados obtenidos al aplicar el esquema de control propuesto mediante dos experimentos. En el primer experimento se utiliza el cambio de pendiente de una plataforma de pruebas para medir el desempeño del control PID-ASLD versus un control PID tradicional. En el segundo experimento se utiliza el control de marcha para observar el comportamiento de las ganancias sintonizadas (Kp, Ki, y Kd), así como la alineación del centro de masa con la referencia horizontal. Finalmente se grafican las coordenadas medidas durante la marcha del ZMP en ambos pies.

**Capitulo 7. Conclusiones y trabajo a futuro**

Se dará cuenta de las conclusiones obtenidas y el cumplimiento de la hipótesis establecida. Se tratará el potencial desarrollo de la inteligencia artifical para la generación de trayectorias del robot bípedo Scout. Además se propone un nuevo esquema de control añadiendo el criterio del ZMP y un magnetómetro como complemento al esquema de control propuesto en el Capítulo 4

**Apéndice A. Operación entre conjuntos difusos**

Se da un breve descripción de las operaciones más importante en los conjuntos difusos y su contraste con la lógica clásica.

**Apéndice B. Relación entre conjuntos difusos**

Se definen las relaciones entre conjuntos difusos y su diferencia con las relaciones con conjuntos clásicos. Se define el producto cartesiano difuso.

**Apéndice C. Operadores difusos generalizados**

Con el fin de hacer el proceso de diseño de un controlador de una forma general, se introducen los operadores generalizados, los cuales proporcionan un compendio de operadores, que pueden variar el comportamiento del sistema según lo deseado.

**Apéndice D. Diseño de un controlador difuso tipo Mamdani**

Se aborda una descripción general a través de un ejemplo, del controlador difuso. Se describen sus cuatro etapas: fusificación, base de conocimientos, inferencia lógica, y defusificación. En cada una de las etapas se muestra la metodología seguida.

**Apéndice E. Estabilidad de controladores difusos**

En éste apéndice se describe la estabilidad de los sistemas difusos, y como el comportamiento asintóticamente estable, puede ser otorgado a través de las reglas de control, interpretando al error y al cambio del error. Se define al punto de equilibrio, y se sientan las bases para elaborar un controlador difuso con un comportamiento asintóticamente estable.

**Apéndice F, Códigos**

Se muestran los códigos: Control de marcha (Python), Tarjeta de adquisición(C++) y Tarjeta driver de servomotores (C++).

**Referencias**

Se enlista cada uno de los soportes acádemicos, multimedia o web para la realización de éste trabajo.

# Capítulo 1

# Introducción

## 1.1 Estado del arte

### 1.1.1 Robots bípedos

La caminata bípeda ha sido un tema sumamente atractivo para la robótica móvil en las ultimas décadas, ya que su movimiento no se limita a terrenos planos, como es el caso de los robots móviles con llantas, esto incrementa el numero de tareas que un robot bípedo puede realizar, aumentando su versatilidad al momento de aplicarlas.

Los primeros indicios del análisis de la caminata bípeda fueron en el renacimiento con *De Motu Animalium*, hecho por Geovanni Borelli en el siglo XVII ( año 1680) [5], en su trabajo comparó las distintas especies de animales estudiando su locomoción, su geometría, su anatomía, entre aquellas especies que eran bípedas, como los pájaros y los humanos. Su principal enfoque era la descripción de los movimientos de manera geométrica, tomando en cuenta los vértices y las posiciones que estos tomaban en el momento del movimiento, como podemos apreciar en la Figura1.1.

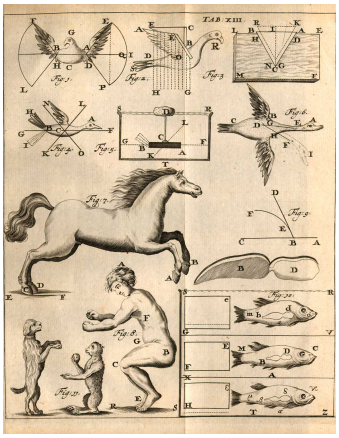

Figura 1.1: *De Motu Animalium*, Giovanni Borelli [5]

Entre los siglos XVII y XIX se destacó un periodo creativo en el desarrollo de máquinas denominadas autómatas (una figura mecánica que imita los movimientos de un ser animado). En esta época los autómatas realizaban distintas tareas entre ellas había quienes imitaban la caminata de animales como leones o perros, la misma caminata humana también fue imitada de manera minimalista, sin embargo había escasos ejemplares, sus movimientos eran limitados y sus tareas restringidas.

Uno de los autómatas mas sobresalientes de esta época fue el que hizo Leonardo da Vinci [8], Éste mantenía una estética humanoide de proporciones antropomórficas, es decir las piernas y los brazos

estaban escalados de la misma forma que un humano. Tenía un diseño complejo de mecanismos que hacían que se moviera de manera suave, por medio de engranes y poleas. El autómata era capaz de sentarse, mover los brazos, y girar la cabeza; por otro lado, no era capaz de caminar a pesar de ser bípedo.

Figura 1.2: Automata Humanoide Leonardo da Vinci [8]

Fue hasta el siglo XX, donde por primera vez se acuño el termino *robot*, en la obra teatral R.U.R. (Rossum Universal Robots), por el escritor checo Karel Capek, en 1921. Como tal la palabra robot se deriva de la palabra checa *robota* que significa *labor o trabajo forzado*, siendo su principal significado el de esclavo [9].

El concepto de robótica fue madurando hasta el punto en el que en el año 1947, en su colección de historias *I, Robot*, Isaac Asimov estableció las leyes de la robótica. Las historias describían la aplicación de las leyes a robots que al menos eran capaces de razonar como lo hace un niño; es decir, que tuvieran la suficiente maduración cognitiva para tomar decisiones morales [6].

Con la leyes de la robótica se sentó la base actual de como ver a los robots. Ya no sólo se incluye la suavidad del movimiento en la construcción ni la complejidad del sistema para llevarlos acabo, sino el raciocinio y la toma de decisiones, a fin de cuentas la inteligencia artificial. Ésto impulso a la investigación e implementación de nuevas metodologías y tecnologías para los años posteriores en la construcción de robots.

En el mundo industrial, en particular en el área de la automatización, en la década de los 60, comenzaron a permear los primeros brazos robóticos o manipuladores [10], sin embargo estos manipuladores comenzaron a alejarse de la estética humanoide, y se enfocaron de lleno al concepto funcional, habiendo variedades y estilos diferentes con el paso del tiempo, por ejemplo los brazos robóticos seriales, los robots paralelos, los robots cartesianos o sus versiones híbridas.

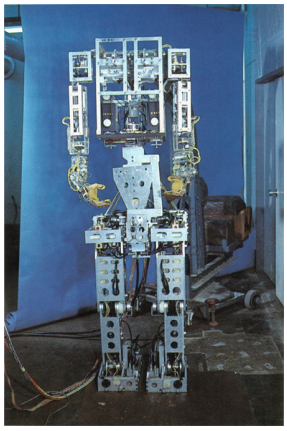

Figura 1.3: Robot Humanoide, WABOT I, WASEDA University, Japón 1973 [8]

Por otro lado se resolvieron los problemas cinemáticos y dinámicos inherentes a los tipos de robots previamente dichos, creando así controladores específicos y nuevos diseños que permitieran automatizar las lineas de producción de una manera más sofisticada, barata, y rápida.

A principios de los años 70 ya se tenia consolidado el concepto de robótica y se habían hecho variedad de manipuladores. En esta década se comenzó con los robots humanoides; cabe destacar, que hubo un enfoque distante anatómicamente hablando del humano, ya que en lugar de una configuración con músculos y tendones, se opto por los servomotores y eslabones rígidos como podemos ver en la Figura 1.4.

En especifico, entre los años 1970-1973 se creo WABOT I (WAseda roBOT) [8],

diseñado por Ichiro Kato, líder del equipo en WASEDA University. Este ejemplar fue el primer robot antropomórfico del mundo a escala humana.

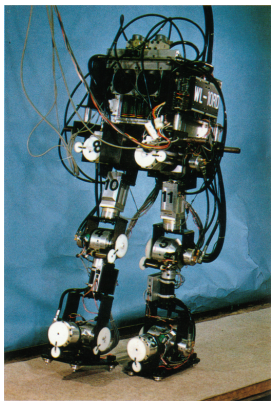

Figura 1.4: Robot Bípedo, WL-10R, WASEDA University, Japón 1983 [8]

En 1983 se construyó WL-10R (Waseda Legged), el cual constaba de 10 articulaciones, 5 en cada pierna actuadas por servomotores y poseía una nueva creación de eslabones y protecciones a base de plástico. En algunas partes reforzado por fibra de carbono [8].

WASEDA University intentó también con robots construidos con músculos artificiales neumáticos, o hidráulicos, sin embargo no tuvieron el efecto deseado ya que los materiales y la dificultad de la construcción ascendía vertiginosamente. Actualmente se han desarrollado prototipos robóticos con músculos y extensiones artificiales bio-inspirados, los cuales permiten una movilidad mucho mas uniforme y suave, estos músculos están basados en multi-filamentos [20].

Japón sentó las bases de la robótica bípeda, sin embargo en otras localidades en la década de los 80, se propició un estudio generalizado de la robótica bípeda, tal es el caso del MIT (Massachusetts Institute of Technology), que en esta década realizó estudios con diversos robots bípedos y su principal enfoque fue el desarrollo de caminatas estables [8].

Por otro lado, en la década de los 90 en Francia surgieron laboratorios especializados en robots bípedos, que tenían como principal objetivo el estudio de una marcha estable probando distintas configuraciones en los pies y en las caderas [8].

En la primera década de los 2000, se comenzó una tendencia hacia los robots de servicio, creando una nueva rama de robots, que auxilian a la humanidad de una manera mas directa, dejando en paralelo la construcción de robots humanoides y bípedos. Un ejemplo claro de los robots de servicio son los exoesqueletos robóticos, los cuales están diseñados para ser un complemento de la fuerza humana en las piernas o en los brazos [8].

Principalmente en Japón se crearon programas para la creación de robots de servicio en favor de los ancianos, estos tipos de robots casualmente eran llamados vehículos con piernas, ya que eran robots bípedos pero con la finalidad de carga; la persona iba encima como en una silla de ruedas [8].

En [23] elaborado por Jung-Yup Kim, Ill-Woo Park y Jun-Ho Oh se establece el esquema de control para una caminata estable sobre piso no regular, se puede resaltar que dicho trabajo se basó en el robot KHR-2, un robot desarrollado en Corea del sur. Su algoritmo de control consiste en mantener un patrón de caminata y realizar una modificación al CoM para poder generar una marcha estable. Se plantea los controles predictivos que hacen que el robot humanoide KHR-2 responda ante una posible caída y el control de vibraciones, diseñado para reducir cualquier tipo de perturbación de éste estilo.

Los análisis hechos en [22], elaborados por Jung-Yup Kim, Ill-Woo Park y Jun-Ho Oh. Dan un primer acercamiento al control de postura. En dichos artículos se solucionó el problema de postura sintonizando un control PID, y un control basado en heurística para la distancia entre la cadera y los pies (longitud de piernas).

Uno de los ejemplos modernos de robots autónomos humanoides, es el robot SAFFiR (ver Figura 1.5), este robot está siendo construido con el objetivo de servir como bombero. Su diseño fue establecido para ser robusto a pesar de las condiciones de trabajo, además su autonomía esta sostenida por un sistema de comunicación CAN (Controller Area Network) de interfaz abierta, lo que le permite la reducción de cables al máximo y una carga de procesamiento compleja [26] [31], esto a su vez permitirá la programacion de diversos algoritmos que ayuden a sus tareas como bombero.

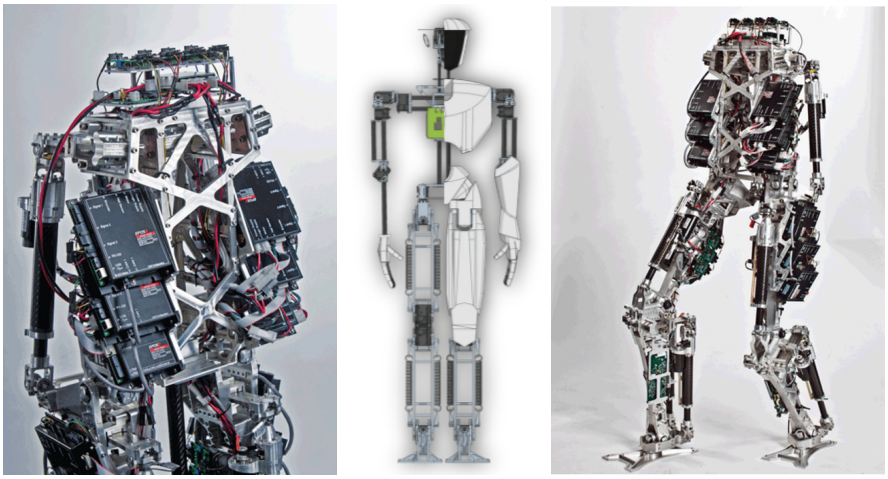

Figura 1.5: Robot Autónomo, Shipboard Autonomous Fire Fighting Robot, Virginia Tech RoMeLa laboratory

### 1.1.2 Lógica difusa

En 1965 se da la primera generalización y definición de los conjuntos difusos por **Lotfi Zadeh** [1] [51], durante esos años, no hubo una gran aceptación para los conjuntos difusos. Se comprendía como una teoria complementaria a la probabilidad, donde la lógica tradicional simplemente consideraba valores ponderados para ciertos elementos de los conjuntos clásicos, sin embargo era todo lo contrario, una teoría solida y general.[54]. Los controladores difusos ***FLC (Fuzzy Logic Controllers)***, han sido de las aplicaciones más importantes y exitosas desde que fuera introducida la teoría de los conjuntos difusos [51] [53].

Los controladores difusos nacen poco después cuando Zadeh muestra los algoritmos difusos en 1968 [52]. En éste trabajo se describe de manera general las aplicaciones que podrían tener los conjuntos difusos con base en algoritmos basados en reglas del estilo **IF-THEN**.

A principios de los años 70, se localizan los primeros controladores difusos basados en el esquema básico de Zadeh [52], los cuales fueron aplicados para procesos industriales o procesos mecánicos. Uno de los sistemas más comunes y probados fue el control del péndulo invertido, un sistema normalmente utilizado para el estudio del control en sistemas no lineales [15]. El Profesor **Ebrahim Mamdani** desarrolló los

[1] Zadeh fue influenciado por el físico y filósofo Max Black quien en 1937 propuso ***Vague sets*** (conocidos actualmente como: ***Fuzzy sets***)[15]

primeros algoritmos de control basado en el trabajo de Zadeh[52]. En 1975 presentó [28], dónde clarificó la metodología de los controladores difusos, una de sus características fue la facilidad con lo que el algoritmo podía ser extrapolado a diferentes sistemas.

La lógica difusa como concepto nace en 1975, justo 10 años después de la presentación de los conjuntos difusos por Zadeh [51]. En[53], Zadeh presenta la generalización (recopilación de trabajos previos) del concepto de la lógica difusa, así como su operabilidad para pasar de un contexto lingüístico a un contexto matemático, hizo énfasis en la aproximación del razonamiento humano al matemático en los sistemas difusos.

Uno de los análisis realizados más importantes por Bart Kosko, dice que los sistemas difusos pueden converger a cualquier función del estilo Entrada/Salida (con un número definido de reglas), probando entonces que estos sistemas pueden remplazar a comportamientos complejos[15].

En 1977 se presentó [34], dónde se elaboró un controlador para una intersección de autopista, en el sistema propuesto los semáforos son los actuadores (verde o rojo) y la variable a regular es el flujo de automóviles circulando, el objetivo es reducir el tiempo de espera de los vehículos al máximo. Algunos controladores previos solucionaron el problema, sin embargo al resolverlo con un controlador difuso la comparación entre controladores fue evidente, el controlador basado en lógica difusa mejoró el desempeño. El desempeño fue medido mediante el tiempo promedio de espera de los autos en la intersección.

El sistema difuso de **Mamdani** se caracteriza por constar de 4 partes principales (Fusificación, Base de conocimiento, Inferencia lógica, Defusificación), no depende de modelos matemáticos, sino de reglas preestablecidas del estilo ***IF-THEN (SI-ENTONCES)*** tal como lo establecio Zadeh en [52] y lo confirma Mamdani en [28]. Las reglas son las encargadas de transportar el conocimiento experto a terminos entendibles para una estrategia de control automático; es decir, basar el comportamiento en el error como entrada al sistema y obtener una salida de control acorde según el sistema diseñado[11]. Éste tipo de sistema difuso permitió que los controladores fueran desarrollados de manera más automatizada y simple [40].

Para 1984 los controladores difusos eran implementados en todo tipo de sistemas, por ejemplo en [36], se utilizó un controlador del tipo Mamdani para un generador de vapor, el controlador fue dividido en tres pequeños controlares SISO (Single-Imput Single-Output) que fungían como auxiliares para que las no linealidades del sistema no lo desestabilizaran.

A finales de los años 80 en Japón hubo una gran aceptación en el uso de sistemas difusos para el control. De hecho al año 1987, se le considera el año de mas éxito, debido al gran número de productos creados con base en los controladores difusos (principalmente en Japón)[15]. La compañía **Hitachi** fue de las primeras en involucrarse con este tipo de controladores utilizando un **FLC** para el tren en Sendai, Japón [15].

A principios de la década de los 90 otra compañía llamada **Fuji**, utilizó un control difuso para el tratamiento de aguas, haciendo que este controlador inyectara los químicos necesarios para la limpieza según el proceso requerido[15], mostrando así la versatilidad de un controlador difuso.

En 1995 Raviraj y Sen [35], hacen una comparación entre controladores, en este estudio analizan el controlador PI convencional, un controlador mediante modos deslizantes y un controlador difuso. Se hace notar que el controlador basado en lógica difusa presenta similitudes con los controladores basados

en modos deslizantes. Los resultados son satisfactorios, el controlador difuso es sugerido para su uso en sistemas electrónicos de potencia.

Por otro lado de manera paralela en los años 90, los profesores **Takagi** y **Sugeno** realizaron investigación con el enfoque de aprendizaje (maquinas de aprendizaje), ahora las reglas deberían ser aprendidas dependiendo de las entradas y salidas (Fuzzy rules), este método se le conoce como modelo difuso de ***Takagi-Sugeno***. Dicho sistema se le conoce también por el nombre de *Fuzzy Learning* ( enfocado a métodos de aprendizaje y adaptación), se ha especializado en el modelado de sistemas. El modelado se hace mediante la recopilación de mediciones experimentales de entrada y salida del sistema en cuestión [15][40].

Gracias a la consolidación de los controladores difusos en la década de los 80, varias propuestas novedosas trajeron nuevos métodos de uso, uno de los mas importantes fue la configuración de los sistemas ***Neuro-Fuzzy*** [15]. La estructura de estos nuevos sistemas permitió que las redes neuronales se apoyaran de los sistemas difusos (su inferencia difusa) para la toma de decisiones del aprendizaje de las mismas[15][40]. Actualmente las técnicas empleadas para los sistemas Neuro-fuzzy están siendo utilizadas para el desarrollo de nueva tecnología, se ha incluido también en estos estudios a los algoritmos genéticos que, al igual que las redes neuronales dan cabida a que la lógica difusa complemente su operación [15].

Los sistemas difusos jerárquicos son simplemente una extensión de los sistemas tradicionales, fueron creados debido a que en los sistemas tradicionales las reglas tienen un crecimiento exponencial es decir, un crecimiento del tipo: $reglas = m^n$, donde **m** son las funciones de membresía y **n** el número de entradas[40]. Los sistemas son conocidos como ***HFS (Hierarchical Fuzzy System)***, al igual que los modelos tradicionales pueden estar asociados a un tipo de estructura, ya sea Takagi-Sugeno o Mamdani [4]. Estos sistemas involucran capas, dentro de cada una se alojan uno o varios sub-sistemas difusos; es decir, que la entrada al sistema pasará internamente por distintas capas para ser procesada antes llegar a la salida[4].

### 1.1.3 Controladores PID autosintonizables mediante lógica difusa

Durante los años 90 se localizan los primeros controladores PID autosintonizables[2] mediante lógica difusa.

Como se muestra en [32], las ganancias $K_p, K_i, K_d$,son sintonizadas a través de un controlador difuso ***(FLC)***, mediante reglas especialmente diseñadas para cada ganancia. Este tipo de sistemas fueron llamados ***Fuzzy PID controllers***, normalmente se abrevian como **FPIDC** tal como se muestra en [32] [50]; también es conocido como, ***STFL-PIDC (Self-Tunning Fuzzy Logic PID Controller)***, tal como aparece en ( *Controladores PID autosintonizables mediante Lógica Difusa* (**CPID-ASLD**))[11][12].

Ching-Long Shih, William A. y Gruver Yun Zhu en el año 1991 presentaron [7], dónde describen a detalle un controlador autosintonizable mediante lógica difusa para el control de fuerza de un robot bípedo; el control de fuerza tiene como objetivo distribuir el peso del robot en ambos pies para una postura estable.

[2]Los controladores PID autosintonizables existen desde la década de los años 70, no utilizan lógica difusa como método de sintonización

Al implementar dicho controlador se desmostró que la autosintonización mejora el desempeño respecto a controladores de ganancias constantes.

Pronto por todo el mundo se dieron avances en esta forma de sintonización, por ejemplo en el año de 1995 Chang-Goo Lee y Sung-Joong Kim presentan [25] en corea del sur. Elaboran un controlador PID autosintonizable mediante lógica difusa utilizando al método de Ziegler-Nichols como parte inicial de la sintonización.

En el mismo año en Reino Unido J. A. Dunlop, K. J. Burnham, D. J. G. James y P. J. Kingse presentan [21], dónde se trata un método basado en lógica difusa para la autosintonización de un controlador. Ésta propuesta se compone de un estimador que manipula al controlador según un compendio de reglas. En este articulo podemos destacar el uso de la lógica difusa para modificar características de los controladores, por ejemplo la adaptabilidad de las ganacias. En conclusión la autosintonización mediante lógica difusa permite que la robustez del controlador sea mayor, dando margen a que los controladores puedan ser utilizados en sistemas no-lineales [3], la unión entre sistemas difusos y controladores convencionales suma beneficios [4].

Los controladores PID autosintonizables mediante lógica difusa son considerados adaptables debido a que las ganancias son dinámicas [17], obedeciendo a las reglas provistas e implementadas utilizando la lógica difusa. Permiten al controlador contener áreas o zonas de trabajo bien definidas. Las reglas van caracterizando el tamaño de las ganancias según las entradas al sistema difuso; este tema se abordará con más detalle en el Capítulo 4, donde se habla a detalle del controlador difuso implementado.

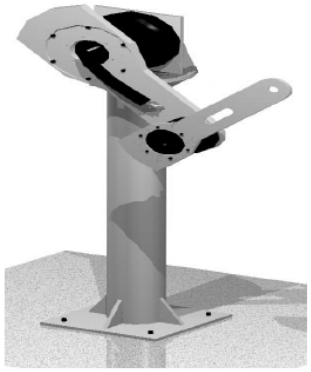

Figura 1.6: Manipulador [44]

Rainer Palma y Christiane Stutz en 2003 [33] elaboraron un generador de trayectorias dinámicas, con el objetivo de probar la autosintonización de controlador propuestos (Fuzzy gain schedulers) del tipo SISO y MIMO sobre una planta no lineal. Cabe mencionar que la planta fue modelada mediante un sistema difuso Takagi-Sugeno y este proporcionó la información necesaria para obtener una autosintonización acorde a la comportamiento de la planta.

En 2009 J.L. Meza, V. Santibañez, R. Soto y M.A. Llama[44] utilizan un controlador PID autosintonizable mediante lógica difusa para el control de manipuladores seriales. Presentan la comparación entre ganancias estáticas y el uso del controlador difuso para la sintonización de las ganancias del controlador propuesto. Las pruebas realizadas demostraron que la precisión obtenida con el controlador PID autosintonizable aumentó sustancialmente a comparación de las ganancias estáticas. En la figura 1.6, se muestra el manipulador serial en el que fue implementado el controlador PID-ASLD.

En 2013 Jana Paulusová, Lukáš Orlický y Mária Dúbravská presentan [32], donde trabajan con un sistema químico normalmente utilizado para evaluar controladores no lineales. Éste sistema contiene no linealidades propias de los sistemas químicos. Se demostró que el controlador PID autosintonizable puede trabajar con sistema no-lineales convencionales mejorando su desempeño.

[3] El desempeño del controlador aplicado en sistemas no lineales, en contraste con los controladores convencionales resulta muy bueno

[4] los controladores difusos pueden ser aplicados en sistemas SISO, MIMO, SIMO, MISO

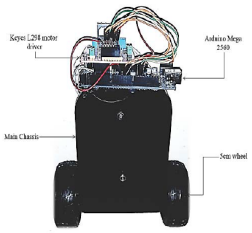

Figura 1.7: Robot de balance [39]

En el estudio experimental realizado por Rasoul Sadeghian y Mehdi Tale Masoule en 2016[39] se demuestra el uso de un PID autosintonizable para el balance de un robot con dos ruedas que se muestra en la figura 1.7. Los parametros iniciales utilizados en el controlador fueron obtenidos mediante algoritmos genétifcos. El análisis determinó que el desempeño del controlador PID autosintonizable fue superior a un controlador PID convencional probado en las mismas condiciones. Se utilizaron perturbaciones externas para poder probar la robustez de los controladores, a comparación del PID convencional el PID autosintonizable mejoró la estabilidad del robot, es decir que el objetivo de control de estabilización o correción del error a cero fue mejor y mas precisa que con un PID convencional.

Por último en 2016 Batıkan E Demir, Raif Bayir y Fecir Duran [3], realizaron un estudio sobre el seguimiento de trayectorias usando un dron autónomo, se utilizó un controlador PID autosintnizable mediante lógica difusa para efectuar el seguimiento deseado. Los objetivos del presente estudio se dividieron en dos:

- Mantener una altitud de referencia
- Seguimiento de trayectoria

Toda acción de control se hizo mediante un sistema en línea, implementado en MATLAB/simulink. En este estudio se concluye que en contraste con un controlador PID convencional, el controlador PID-ASLD difusa cumple con los propósitos establecidos en menos tiempo y con menos error. Un esquema del sistema utilizado puede ser visto en la Figura 1.8.

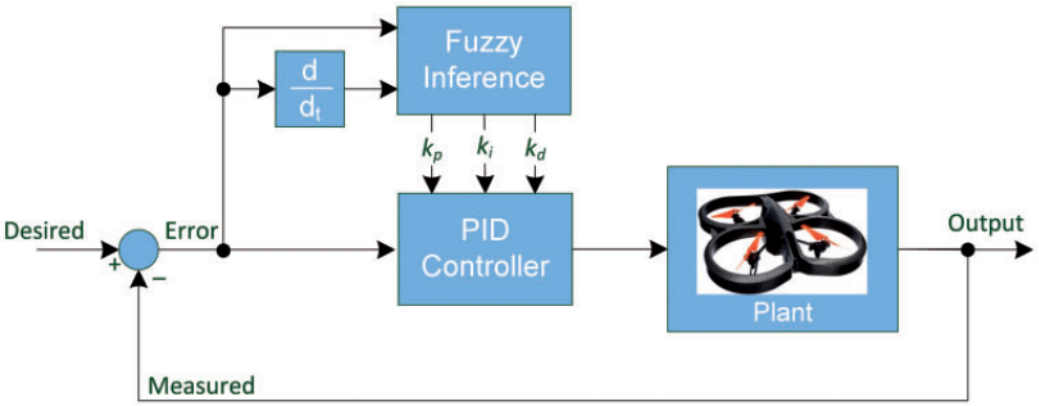


Figura 1.8: Diagrama de bloques [3]

## 1.2 Planteamiento del problema

En el verano del año 2008 el Departamento de Mecatrónica de la Facultad de Ingeniería de la UNAM realizó la adquisición del robot bípedo Scout, con el objetivo de que los alumnos de esta institución desarrollen las herramientas físico-matemáticas, electrónicas y de programación para poner en marcha dicho robot, bajo criterios previamente definidos.

Recientemente Ulises J. en [37] utilizó el modelado simplificado *carro-mesa* para la obtención de trayectorias de marcha en linea recta con el robot bípedo. En este trabajo las trayectorias fueron calculadas de manera externa al robot y fueron implementadas en lazo abierto utilizando LabView y arduino.

Por otro lado, Fernanda Merino [29] implementó *en lazo abierto* el modelado simplificado *carro-mesa parametrizado* para la obtención de trayectorias de marcha en linea recta, arco de circunferencia, subida de escalón y rampa para el robot bípedo. Las trayectorias implementadas en [29] consideran una altura no constante del centro de masa (COM, por sus siglas en inglés). éstas fueron mejoradas debido a la inclusión de un parámetro en la altura de la cadera, el cual hace más natural la caminata.

En este contexto, la presente Tesis abordará el siguiente problema: los cálculos para el equilibrio dinámico del robot bípedo Scout resultan complejos, debido a esto las trayectorias obtenidas para la caminata fueron convenientemente implementadas en lazo abierto. La marcha implementada en lazo abierto funciona bajo condiciones ideales; es decir, si las trayectorias para la marcha fueron generadas para un piso plano, funcionarán en ese contexto y carecerán de robustez ante perturbaciones. A partir de lo antes mencionado surge la necesidad de implementar un lazo de control cerrado que corrija las trayectorias en lazo abierto ante cambios de pendiente.

El lazo de control cerrado puede ser diseñado a través de las características principales del robot: usando la posición del ZMP (como criterio de estabilidad), la Orientación del centro de masa. Cada una de las estrategias de control serán detalladas a continuación.

**Estrategia de control - Posición del ZMP**

A través de sensores de fuerza es posible calcular el centro de presión (CoP, por sus siglas en inglés) del robot bípedo; es decir, el reemplazo de todas las fuerzas reactivas entre el pie del robot y el suelo en un solo punto. De la misma forma existirá una resultante en el centro de masa (CoM, por sus siglas en inglés) de todas la fuerzas activas del robot durante su movimiento como, fuerzas debidas a la gravedad (peso), inercias, momentos. El punto dónde la resultante en el CoM y la resultante en el CoP se anulan, se le conoce como punto de momento cero (ZMP, por sus siglas en inglés). Si el ZMP está dentro del polígono de soporte, se dice que la marcha es estable [13] [29].

En este contexto se puede utilizar el ZMP como discriminante para determinar la estabilidad de la marcha y de esta forma calcular acciones de control que permitan la corrección de las trayectorias ante perturbaciones con el objetivo de mantener una marcha estable.

**Estrategia de control - Orientación del centro de masa**

A través del uso de una Unidad de Medida Inercial (IMU), es posible medir la orientación del centro de masa (CoM). La orientación del centro de masa está totalmente relacionada con la estabilidad de la caminata y la postura; si el centro de masa permite que la posición del ZMP se mantengan en el polígono

de soporte, entonces la caminata y la postura serán estables [13] [29]. Las acciones de control estarán designadas para corregir la orientación del centro de masa con base en una referencia, con el objetivo de compensar el error que pudiera surgir ante perturbaciones como cambios de pendiente.

En el marco de desarrollar un esquema de control completo para el robot bípedo, el cual contenga control basado en el ZMP, en la orientación del centro de masa para su marcha y postura; se decidió trabajar en las estrategias de control antes mencionadas de manera independiente en el entendido de que éstas se complementen entre sí. Este trabajo tomó entonces la estrategia de control de orientación del CoM.

## 1.3 Hipótesis

- Se propone dar estabilidad a la caminata bípeda modificando los ángulos de entrada de la trayectoria implementada en lazo abierto en las articulaciones del robot bípedo, considerando una corrección en ángulo mediante la comparación entre la orientación del centro de masa (CoM) de referencia y el medido por la Unidad de Medida de Inercia (IMU).

## 1.4 Justificación

Dado que el modelo dinámico del robot bípedo es complejo, se decidió utilizar un controlador no basado en modelo. La solución para la corrección del error, era utilizando un control proporcional, sin embargo este controlador no llega al valor de referencia; por lo que una parte integral debe ser adicionada para llegar al valor deseado, la parte integral ocasiona oscilaciones y un desempeño lento y para llegar a un desempeño más rápido se le debe incorporar la parte derivativa. El controlador PID soluciona en primera instancia el problema de la orientación del centro de masa del robot bípedo en las direcciones más conflictivas. Debido a que las ganancias del PID son constantes, el controlador no presenta un comportamiento robusto ante perturbaciones, sólo funciona bien cerca del punto de equilibrio.

Se optó por desarrollar un controlador basado en heurística (experiencia), es decir diseñado a partir de conocimiento experto, la mejor manera de poder implementar este conocimiento experto es a través de inferencias lógicas, por lo que se propuso un controlador difuso para la generación de una marcha estable. El principal problema de esta propuesta es el número de reglas propias del sistema difuso, ya que crecen exponencialmente; es decir, que las reglas tienen un crecimiento del tipo:

$$reglas = m^n \tag{1.1}$$

Dónde: **m** son las funciones de membresía y **n** son las entradas. Esta definición aplica por cada salida del sistema difuso, entonces para cada salida del sistema se requerirá de $m^n$ reglas. Para un sistema dónde se deben actuar 10 servomotores y suponiendo que de cada servomotor se puede obtener el error, la suma del error y el cambio del error, éste sistema contaría con $10 * m^n = 10 * (5)^3 = 1250$ reglas, por lo tanto computacionalmente sería muy costoso. Además la definición de reglas para una marcha estable

utilizando este tipo de sistema difuso presenta demasiadas implicaciones de conocimiento experto [5] por lo que sería complejo y tardado implementarlo.

El problema antes mencionado tiene una solución inmediata, la cual es diseñar otro sistema difuso en jerarquias, denominados *Sistemas Jerárquicos Difusos (Hierarchical Fuzzy Systems)*. Estos sistemas sintetizan las reglas en un compendio de capas configuradas para obtener el mismo funcionamiento que si se hubiera implementado tradicionalmente, reduciendo la complejidad del sistema difuso. [4].

El control difuso tradicional fue descartado, por ser más complejo que el control PID tradicional. En aras de mejorar el controlador PID previamente propuesto, se propuso que el controlador estuviera a su vez sintonizado por medio de sus ganancias usando un controlador difuso tipo Mamdani. Como se menciona en el estado del arte, una de las principales ventajas del controlador PID autosintonizable mediante lógica difusa (PID-ASLD), es el aumento en el desempeño. El controlador difuso sintoniza las ganancias del controlador PID tomando en consideración el error, el cambio del error y la suma del error. Todas las mediciones que haga el controlador difuso serán en-linea por lo que la sintonización será tan rápida como el procesamiento de la tarjeta utilizada para la implemetación del controlador.

Las reglas en este caso estarán orientadas a corregir el error y actuarán conforme al cambio del mismo con respecto al tiempo así como evaluar la suma del error, para una posterior corrección. La diferencia con el controlador difuso tradicional antes mencionado yace en la implementación, ya que un controlador PID-ASLD solo tiene tres salidas, por lo que reduce sustancialmente el número de reglas a utilizar. Por otro lado en este mismo controlador existe una disminución de reglas aplicables según sus salidas ($K_p$,$K_i$ y $K_d$), esto se debe a que las ganancias son dependientes de solo 2 de sus entradas a la vez:

$K_p = f(e, \Delta e)$ está en función del error y del cambio de error.

$K_i = f(e, \Sigma_0^t e)$ está en función del error y de la suma del error.

$K_d = f(e, \Delta e)$ está en función del error y del cambio de error.

Cabe mencionar que la diferencia entre la dependencia de la ganancia $K_p$ y $K_d$, es la importancia que se le dá al error y al cambio del error para cada una de éstas durante la definición de las inferencias lógicas, además es evidente que la ganancia $K_p$, siempre estará asociada a la parte proporcional y la ganancia $K_d$ a la parte derivativa del control PID. Si se analiza la totalidad de reglas considerando las tres entradas serían $3 * (5)^3 = 375$, sin embargo las dependencias antes mencionadas hacen $3 * (5)^2 = 75$ reglas utilizables, el desarrollo de las reglas será descrito en el capítulo 4

Como complemento al control PID-ASLD, un controlador difuso tradicional es incluido. Con éste se manipula la longitud entre la cadera y los tobillos, permitiendo que el robot se adapte a una pendiente. El esquema de control detallado se encuentra en el capítulo 4

[5] Se tendrían que realizar numerosas pruebas experimentales hasta que la marcha fuera estable

## 1.5 Objetivo general

Diseñar, e implementar un controlador difuso tipo Mamdani y un controlador PID autosintonizable mediante lógica difusa que modifiquen ángulos constantes y ángulos de una trayectoria preestablecida para corregir la orientación del centro de masa de un robot bípedo, dando lugar a una postura y caminata estable.

## 1.6 Objetivos particulares

- Desarrollar, e implementar un sistema embebido eficiente para el traslado de información, así como aumentar la velocidad de adquisición, y la actuación en los motores en comparación con el sistema anterior.
- Diseñar un controlador difuso tipo Mamdani y un controlador PID autosintonizable mediante lógica difusa que permitan la modificación de ángulos constantes en las articulaciones del robot para mantener una postura estable ante perturbaciones.
- Diseñar un controlador PID autosintonizable mediante lógica difusa que permita la modificación de ángulos de una trayectoria preestablecida en lazo abierto para mantener una caminata estable ante perturbaciones.
- Confirmar mediante las coordenadas del ZMP que la marcha del robot bípedo es estable.

## 1.7 Alcances y limitaciones

### 1.7.1 Sistema embebido

**Alcances:**

- Almacenamiento de trayectorias.
- Procesamiento de los controladores difusos.
- Almacenamiento de datos recabados durante el control de postura y de marcha.
- Interfaz gráfica para seleccionar las distintas estrategias de control.

**Limitaciones:**

- Semi-autonomía, se utiliza una fuente en vez de pila
- No graficar en-linea, dado que consumiría procesamiento innecesariamente.

### 1.7.2 Esquema de control

**Alcances:**

- El control de postura funcionará en soporte doble.
- El control de marcha será aplicado durante una trayectoria previamente calculada.
- El desarrollo del controlador PID autosintonizable y el control difuso de postura estarán dirigidos específicamente a la corrección en la orientación del centro de masa.
- Las coordenadas del ZMP estarán destinadas a comprobar que la marcha es estable.

**Limitaciones:**

- El ZMP no se incluirá en la generación de las acciones de control.
- El control de postura soporta una inclinación de 20 grados hacía enfrente y 10 grados en con una pendiente lateral. Esto se debe principalmente a dos factores, la composición mecánica del robot en la suela del robot ya que se tiende a resbalar cuando se supera los ángulos previamente mencionados y el torque que proporcionan los servomotores del robot bípedo no es suficiente para la corrección del error.
- El control de marcha no soporta más de 10 grados hacía enfrente y 5 grados en una caminata con una pendiente lateral. Los principales motivos de estas limitaciones es debida a la composición mecánica de las piernas, ya que para pendientes mayores, la pierna no se levanta para dar un paso limpio; es decir que pega sobre el suelo, ésto propicia una caída inmediata.

# Capítulo 2

# Preliminares

## 2.1 Sistema de referencia

Para hacer el estudio y análisis de una marcha bípeda es necesario definir a los planos que dividen el movimiento. Se hace una división con tres planos perpendiculares entre si, para hacer una descripción mas detallada se utiliza el cuerpo humano, como se muestra en la Figura 2.1.

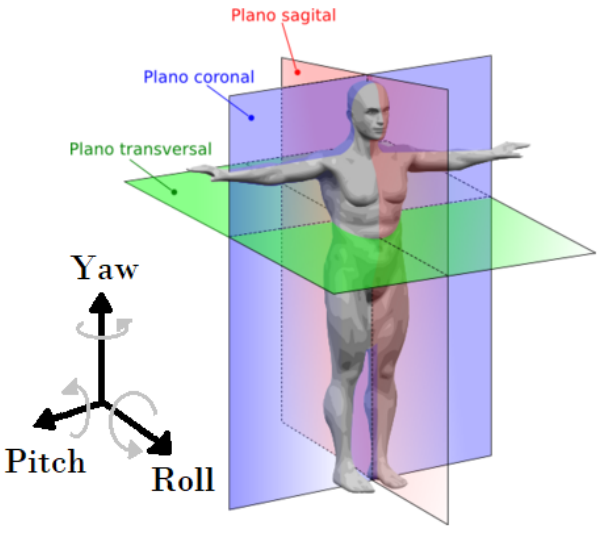


Figura 2.1: Sistema de referencia para el cuerpo humano [24]

El plano Sagital divide el cuerpo verticalmente en dos lados: derecho e izquierdo, en este mismo plano actúa el ángulo *Pitch*. Por otro lado el plano Transversal u Horizontal divide el cuerpo en dos mitades: superior e inferior, en este plano actúa el angulo *Yaw* y por último el plano Frontal o Coronal, divide el cuerpo verticalmente en anterior (parte trasera) y posterior (frente), en este plano se utiliza el ángulo *Roll* (ver Figura 2.1)[24].

## 2.2 Locomoción bípeda

La locomoción bípeda es una de las habilidades mas importantes del ser humano. La caminata bípeda es definida en [8], como:

*"Una sucesión de posturas utilizando las extremidades inferiores como puntos de apoyo en el suelo, generando así balance para poder moverse con una dirección preestablecida."*

Gracias a la versatilidad de la caminata bípeda se puede ir en cualquier dirección deseada, superando así a

los robots móviles con llantas. Sin embargo el mantener una postura o hacer una caminata bípeda requiere de una actuación precisa para un buen balance, ya que naturalmente es un problema inestable. La complejidad de la caminata impone la primera limitante, las articulaciones, en el cuerpo tenemos muchas de ellas. Para poder replicarlas los robots utilizan juntas, eslabones y servomotores, éstos emulan la estructura del cuerpo e imitan de manera similar el movimiento del cuerpo humano.

### 2.2.1 Ciclo de marcha

El ciclo de Marcha es un concepto utilizado para poder identificar el fin e inicio de la caminata bípeda, dado que la caminata bípeda es una sucesión de posturas, esta sucesión deber ser repetible, lo cual permite entonces reproducir la caminata. El ciclo de marcha se considera periódico idealmente, en realidad la periodicidad se ve mermada por las irregularidades del suelo [38].

A continuación en la figura 2.2 se definen las etapas mas importantes del ciclo, así como la distinción entre soporte simple y doble:

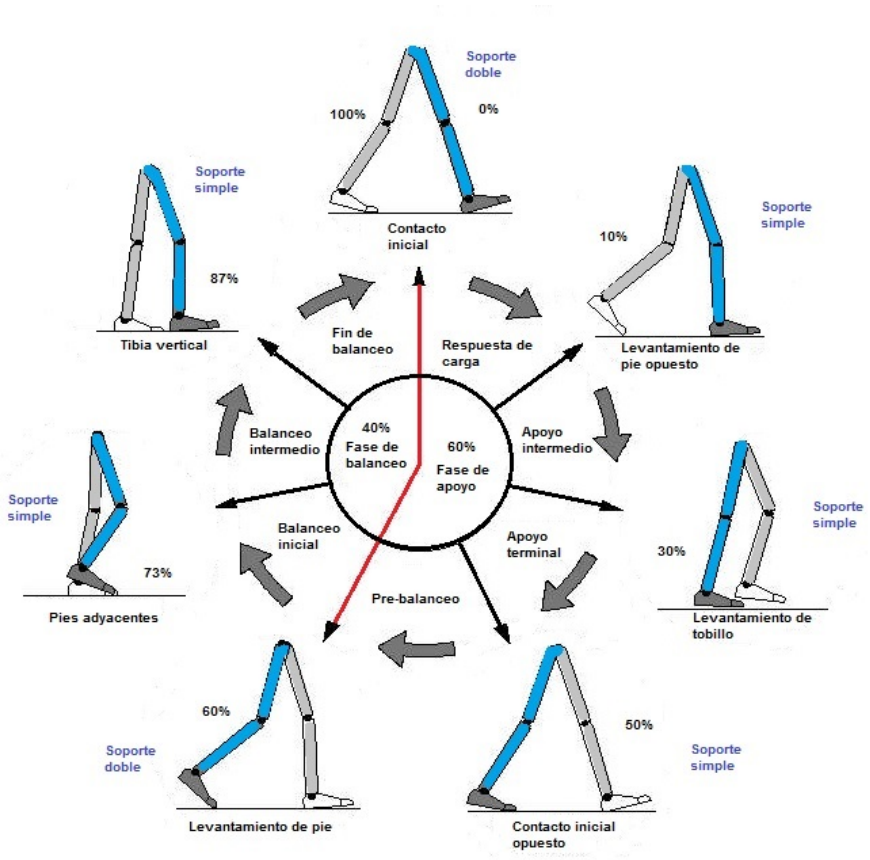


Figura 2.2: Fases o periodos del ciclo de una marcha bípeda [2]

Para hacer unas distinción más clara en la figura 2.3 se presenta el ciclo de manera lineal:

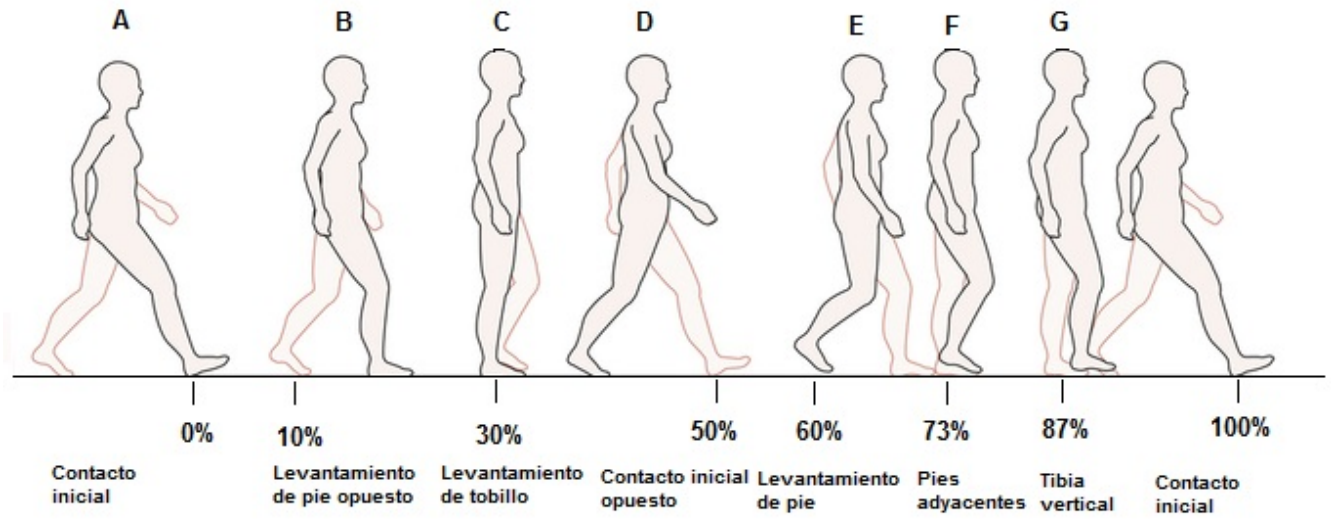


Figura 2.3: Fases o periodos del ciclo de marcha[2]

A su vez el ciclo de marcha se divide en dos grandes partes, como se puede observar en la figura 2.2, un 60 porciento pertenece al apoyo logrado durante la caminata, se caracteriza principalmente por el apoyo de los miembros en distintos tiempos del ciclo de marcha y la respuesta de carga justo después de apoyar en el suelo. Por otro lado tenemos que un 40 porciento pertenece al balanceo, incluye el avance de las extremidades[38] [45].

Dentro del ciclo de marcha existen dos fases intermitentes, la fase de soporte doble, y la fase de soporte simple, se caracterizan por que entre ellas se hace el cambio de cadena cinemática cerrada a cadena cinemática abierta.

**Soporte doble:**

Cuando el robot bípedo está completamente soportado por ambos pies [13].

**Soporte simple:**

Cuando el robot bípedo está completamente soportado por un solo pie [13].

En soporte doble el mecanismo del robot es considerado una cadena cinemática cerrada, todas las juntas del mecanismo deberían ser controlables, sin embargo el contacto entre la planta del pie y el suelo es considerado como junta pasiva, ésta junta jamás podrá ser controlable directamente. Para solucionar el problema indirectamente, la tarea más importante del mecanismo de locomoción, será asegurar que la planta del pie tenga contacto con el suelo[38].

Para el diseño de una caminata artificial con base en trayectorias, se deben considerar al menos éstos conceptos[38].

## 2.3 Estabilidad de la locomoción bípeda

### 2.3.1 Polígono de soporte

El polígono de soporte se define como una superficie envolvente convexa formada por la posición de los pies del robot bípedo, ésta cambia con el tiempo dependiendo del apoyo[13]. En pocas palabras, una envolvente convexa es aquella que permite unir todos los puntos que están dentro de ésta sin salir de ella, en la Figura 2.4 se puede observar la envolvente convexa que generan los pies de un bípedo, en este contexto, la forma del polígono de soporte (envolvente convexa) estará estrechamente ligada a la forma de los pies del robot. En la figura 2.4 se observa el cambio del polígono de soporte dependiendo del apoyo.

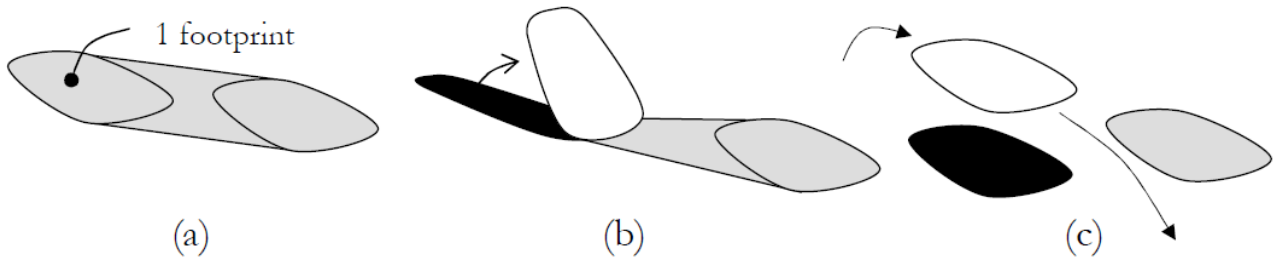


Figura 2.4: Poligono de soporte. (a) Soporte doble (b) fase de prebalanceo (soporte doble) (c) Soporte simple [13] [29]

### 2.3.2 Centro de masa - CoM (Center of Mass)

En todo momento un robot bípedo percibe a la gravedad, las fuerzas gravitacionales que experimenta el robot por todo su mecanismo pueden ser representadas por una sola fuerza, ésta fuerza equivalente actuaría en el centro de masa del robot bípedo[13]. El vector $P_{CoM}$ asociado a ésta fuerza equivalente puede ser descrito como:

$$P_{COM} = \frac{\sum_{i=1}^{n} m_i P_i}{\sum_{i=1}^{n} m_i} \tag{2.1}$$

Dónde: $m_i$ es la masa de cada uno de los eslabones y $P_i$ son los vectores que marcan la distancia respecto al CoM de cada uno de los eslabones. En un instante de tiempo dónde el robot permanezca estático, la proyección en el suelo de la fuerza gravitacional equivalente en el centro de masa puede ser determinada mediante la ecuación **??**. Se sabe que cuando dicha proyección esté dentro del polígono de soporte el robot bípedo no tenderá a caer.

$$\sum_{i=1}^{n} [(P_{FCoM} - P_i) \times m_i g] = 0 \tag{2.2}$$

Dónde: $m_i$ es la masa de cada uno de los eslabones, $P_i$ son los vectores que marcan la distancia respecto al CoM de cada uno de los eslabones y $P_{FCoM}$ es el vector proyeccion del centro de masa en el suelo.

### 2.3.3 Centro de presión - CoP (Center of Pressure)

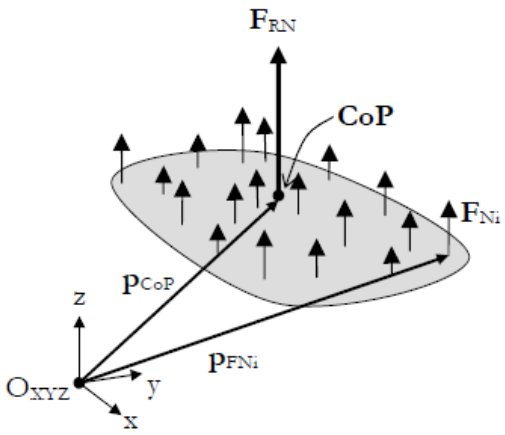


Figura 2.5: Posición del CoP [13]

El centro de presión se define como aquel punto, dónde **las fuerzas reactivas** distribuidas en la planta del pie del robot bípedo, debidas a la gravedad, se pueden reperesentar por una fuerza equivalente (resultante) en una dirección normal o perpendicular al plano de la planta del pie. En la Figura 2.5 se puede observar la posicion del CoP, así como la representación gráfica del vector asociado $P_{CoP}$ [13].

La fuerza resultante en ese punto puede ser obtenida mediante la suma de todas las fuerzas distribuidas en la planta del pie, tal y como se puede observar en la ecuación 2.3.

$$F_{RN} = \Sigma_{i=1}^{n} F_{Ni} \tag{2.3}$$

Para calcular las coordenadas del CoP, se debe calcular su vector asociado. Se toma en cuenta las posiciones de cada una de las fuerzas reactivas considerando su vector de posición y multiplicandolas por cada una de su magnitud, en la ecuación 2.4, se muestra el calculo del vector $P_{COP}$ [13].

$$P_{CoP} = \frac{\sum_{i=1}^{n} F_{Ni} P_i}{\sum_{i=1}^{n} N_i} = \frac{\sum_{i=1}^{n} F_{Ni} P_i}{F_{RN}} \tag{2.4}$$

Dónde: $i$ es el numero de fuerzas reactivas distribuidas por la planta del pie, $P_i$ son los vectores asociados a la posición de cada una de las fuerzas reactivas, $F_{RN}$ es la fuerza equivalente a la suma de todas las fuerzas reactivas y $P_{COP}$ es el vector asociado al centro de presión.

### 2.3.4 Punto de momento cero - ZMP (Zero Moment Point)

El punto de momento cero mejor conocido como ZMP (por sus siglas en inglés), **es el punto** en dónde todas *fuerzas reactivas* debidas a la gravedad sobre la planta del pie son remplazadas por una fuerza resultante en el CoP y ésta se cancela con la fuerza proyectada en el CoM del robot bípedo. Solo en ese punto se dice que está el ZMP [38], de otra forma se define como punto de presión (CoP, por sus siglas en inglés). En la figura 2.6 se observa en un caso hipotetico de balanceo la locación del ZMP.

El termino ZMP fue acuñado por Vukobratovic y Stepanenko en 1972 [49]. Antes de ser acuñado como ZMP, fue usado como un punto indicador cuando las fuerzas reactivas del suelo se balaceaban con aquellas debidas a la gravedad. Sin embargo no se tomo en consideración para la estabilidad en primera instancia. Cuando se le incluyo como criterio de estabilidad en la locomoción bípeda fue tal el grado de

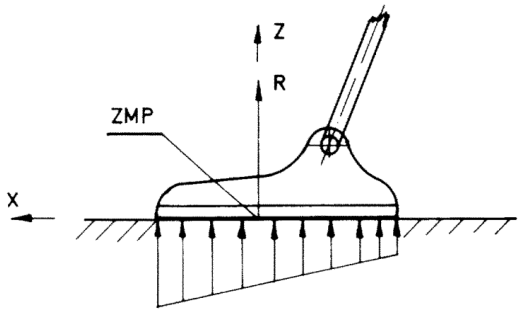


Figura 2.6: Locación gráfica del ZMP, vista lateral [49]

importancia que tomó, que revolucionó el estudio del control y la generación de trayectorias en los robots bípedos, siendo uno de los conceptos más populares para indicar la estabilidad de la marcha en un robot bípedo.[38].

**Criterio de estabilidad del ZMP**

*Para lograr una caminata estable,el ZMP debe estar en todo instante de tiempo dentro del poligono de soporte [13].*

Para clarificar el concepto, se presenta la Figura 2.7, donde se pueden observar las fuerzas reactivas en la planta de un robot bípedo en tres dimensiones. Dado que la planta del pie tiene un contacto totalmente plano con el suelo, podemos hacer evidente que solo habrá momentos respecto al eje Z. Al mantenerse en el mismo plano, no existen momentos respecto al eje X o Y, en el CoP; es decir:

$$\tau_x = \tau_y = 0 \tag{2.5}$$

Entonces si se considera un número finito de puntos de contacto $p_i$ con $(i = 1, 2, ..., n)$ y vectores de fuerza $f_i$ con componentes $f_{ix}$, $f_{iy}$, y $f_{iz}$. Las posiciones de contacto pueden ser calculadas mediante la asociación de puntos de contacto con las fuerzas reactivas en el eje Z, entre el total de las fuerzas en dirección Z:

$$p = \frac{\sum_{i=1}^{n} (p_i f_{iz})}{\sum_{i=1}^{n} f_{iz}} = \frac{\sum_{i=1}^{n} (p_i f_{iz})}{F_z} \tag{2.6}$$

De la ecuación 2.6, podemos renombrar:

$$\alpha_i = \frac{f_{iz}}{F_z} \tag{2.7}$$

Sustituyendo $\alpha_i$, en la ecuación 2.6, tenemos que:

$$p = \sum_{i=1}^{n} \alpha_i p_i \tag{2.8}$$

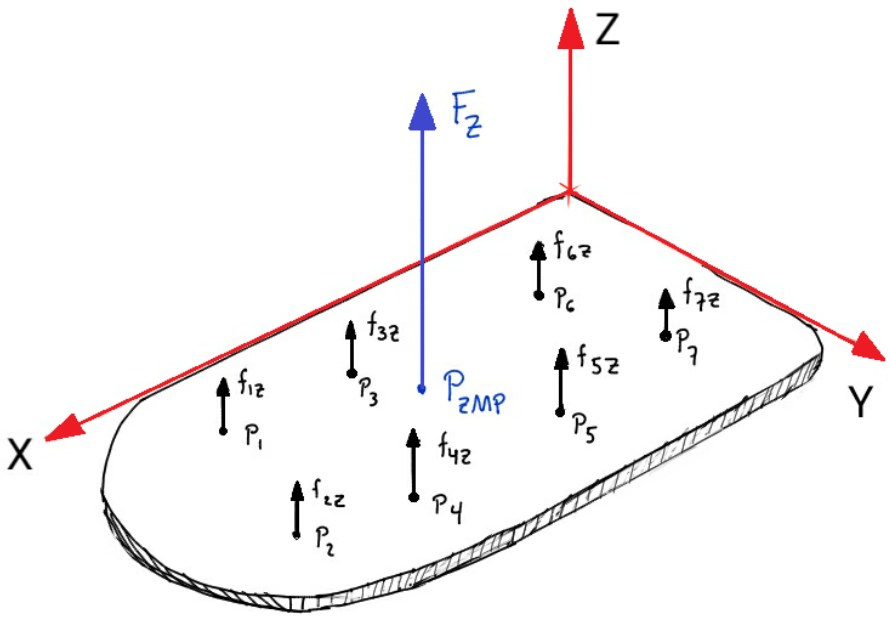


Figura 2.7: Fuerzas de reacción ante el contacto con el suelo, vista en tres dimensiones [49]

Dado que $f_{iz} \geq 0$ para cada $(i = 1, 2, 3..., n)$, entonces:

$$\begin{cases} \alpha_i \geq 0 \ \ para \ \ i = 1, ..., N \\ \sum_{i=1}^{n} \alpha_{i=1} \end{cases} \tag{2.9}$$

Como los puntos de contacto que satisfacen a la ecuación 2.9, generan el polígono de soporte, se asegura que el ZMP nunca salga de él.

El momento angular obtenido respecto al ZMP, se calcula mediante:

$$\tau = \sum_{i=1}^{n} (p_i - p) \times f_i \tag{2.10}$$

La ecuación 2.10 puede ser descrita por medio de las componentes de los puntos de contacto. De ésta forma se puede dividir el momento angular con coordenadas en X y Y, tal como se muestra en la ecuación 2.11:

$$\tau_z = \sum_{i=1}^{n} \left[ (p_{ix} - p_x) f_{iy} - (p_{iy} - p_y) f_{ix} \right] \tag{2.11}$$

Como se muestra en la ecuación 2.5, los momentos en dirección X y Y, son nulos, ésto implica que se

pueden reacomodar las ecuaciones de la siguiente forma:

$$\sum_{i=1}^{n} \tau_x = 0 \rightarrow \sum_{i=1}^{n} f_i(P_{ix} - P_x) = 0 \tag{2.12}$$

$$\sum_{i=1}^{n} \tau_y = 0 \rightarrow \sum_{i=1}^{n} f_i(P_{iy} - P_y) = 0 \tag{2.13}$$

Las ecuaciones 2.12 y 2.13, permitirán hacer el calculo de la posición del ZMP durante la marcha, en el capitulo 6 se detalla la implementación del ZMP en el robot bípedo Scout.

### 2.3.5 Marcha estáticamente estable

Se le llama a la marcha de un robot bípedo estáticamente estable, si en todo momento el ZMP y la proyección de la fuerza del centro de masa permanecen dentro del polígono de soporte. Ésto implica que si en algún momento el robot bípedo dejara de caminar la postura del mismo es estable. Para lograr lo antes mencionado la dinámica del robot debe ser muy lenta, por lo que la velocidad de la caminata misma es lenta.[1] [13].

### 2.3.6 Marcha dinámicamente estable

Si el ZMP se encuentra dentro del polígono de soporte mientras la proyección de fuerza del centro de masa lo abandona durante la marcha, se le denomina una marcha dinámica estable. La velocidad de la caminata es rápida y el centro de masa es móvil, es decir que a comparación de la marcha estática estable, el centro de masa se "balancea"[13].

## 2.4 Lógica difusa

### 2.4.1 Antecedentes

La idea de formular conocimiento humano a partir de matemáticas y aplicarlo en maquinas ha sido motivo de estudio, uno de los enfoques es conocido como inteligencia artificial (AI, por sus siglas en inglés), la cual es una rama de la computación dedicada al estudio del razonamiento, pensamiento y comportamiento de los sistemas con un enfoque *inteligente*. Los lenguajes orales (naturales) ocupados por el hombre son vagos o ambiguos, es decir que siempre existirá incertidumbre cuando se describa algún concepto o abstracción (la precisión de las palabras en cada lenguaje es relativa). La **AI** permite que el razonamiento humano utilizado en estos lenguajes pueda ser implementado en algoritmos computacionales. Una de las formas en las que el lenguaje natural puede ser implementado mediante inferencias lógicas del tipo

[1] La caminata lenta se logra al reducir todas las velocidades angulares de las juntas.

Modus ponens (MP) o Modus tollens (MT). Los sistemas difusos utilizan las inferencias lógicas como motor de toma de decisiones. La lógica difusa fue desarrollada con la intención de ser una extensión de la lógica booleana, con el objetivo de fungir como receptor de conceptos ambiguos, y transformar dichos conceptos en cantidades exactas; la teoría está basada en los conjuntos difusos, que a su vez es una generalización de la teoría de conjuntos clásicos, más adelante se describen éstas dos teorías.

En otras palabras, la lógica difusa empata el lenguaje humano (ambiguo), con la lógica de una computadora (preciso), esto se implementa usando reglas heuristicas (basadas en experiencia), éstas reglas servirán como estructura del comportamiento deseado, esto da pie a que los sistemas difusos sean utilizados como controladores para sistemas complejos y no lineales. Cabe mencionar que uno de los requisitos de esta implementación de reglas es la necesidad de agentes expertos, es decir que se conozca el comportamiento del sistema a fondo. Ya que los sistemas difusos están basados en un modelado heuristico (experiencia), se deber resaltar la diferencia entre modelos usados para la descripción de sistemas. A continuación se exponen las tres maneras principales para describir a los sistemas:

**Modelo Experimental**

Este modelo esta determinado por experimentar con el sistema a través de proporcionarle diferentes entradas y caracterizar sus salidas, de esta forma se pueden obtener gráficas entrada-salida necesarias para comprender la reacción del sistema. Con la información obtenida de los experimentos debería ser posible hacer una regresión y obtener un modelo aproximado del sistema (inversión de datos). Sería posible implementar un controlador que pueda funcionar en cierta región de operación, usando el modelo experimental obtenido.

**Modelo matemático**

El modelado matemático permite describir a un sistema en ecuaciones matemáticas, comúnmente se utilizan las ecuaciones diferenciales ordinarias ya que pueden describir un comportamiento en el tiempo, sin embargo cuando un modelo matemático es complejo, el procesamiento es lento y extenso. En muchas ocasiones el modelo matemático complejo se asume con ciertas idealizaciones, que realizamos en favor de cálculos mas rápidos, creando un modelo matemático más sencillo y facil de procesar. En ingeniería la teoría de control es muy amplia, pero en lo particular para los modelos matemáticos casi nunca hay un comportamiento lineal, por lo general se busca diseñar un controlador basado en un modelo linealizado para que la implementación sea sencilla y barata, aunque existan métodos en los cuales se diseñan controladores no lineales.

**Modelo heurístico**

Este tipo de modelado consiste principalmente en la comprensión de un sistema, es decir conocer cada uno de sus aspectos y comportamientos (agente experto). Al igual que el modelo experimental lo que se pretende es conocer al sistema en cuestión usando la caracterización de las salidas a través de las entradas, a diferencia del modelo experimental éste modelo utilizará la experiencia (inferencias lógicas) como motor de caracterización. La experiencia puede ser implementada usando inferencias lógicas del tipo, ***SI (condición) ENTONCES (consecuencia)***, que se podría comprender mejor como ***SI (condición) ENTONCES (acción)***, aunque también la experiencia se puede representar en forma de ecuaciones matemáticas, un ejemplo claro de ésto, es la distancia mas corta entre dos puntos (linea recta), que utiliza la ecuación $\sqrt{x^2 + y^2}$, sabiendo entonces que la distancia mas corta entre dos puntos estará predicha por

ésta ecuación, se puede tomar en consideración para hacer decisiones en el problema de navegación de robots móviles, con el fin de llegar a un punto deseado. En conclusión la experiencia es una inferencia lógica arbitraria, que podría en su momento representar un razonamiento aproximado y éste razonamiento puede ser utilizado para representar un sistema. Un ejemplo claro de ésto es el sistema difuso tipo Takagi-Sugeno, el cual es capaz de modelar un sistema usando datos entrada-salida, e inferencias lógicas.

### 2.4.2 Conjuntos difusos

En aras de tener una comprensión absoluta sobre los conjuntos difusos es necesario describir a los conjuntos clásicos y sus propiedades, haciendo hincapié en las diferencias con los conjuntos difusos.

La teoría de conjuntos es una de las ramas de las matemáticas que estudia a las agrupaciones de elementos que figuran como un todo [41] [14]. A continuación se presentan algunas ejemplificaciones de conjuntos clásicos:

$$A = \{1, 2, 3, 4, 5\} \tag{2.14}$$

$$B = \{c, k, l, j, n\} \tag{2.15}$$

$$C = \{hola, saludo, mano, cabeza, jugo\} \tag{2.16}$$

$$D = \{x < 1 / x \in \mathcal{R}\} \tag{2.17}$$

Como se observa hay conjuntos formados por letras, palabras, números, etc... . En la teoría clásica los elementos son pertenecientes a los conjuntos que los contienen con un valor de 1 o 0, es decir existen dentro del conjunto o no (exclusión inmediata).

Los conjuntos a su vez forman parte de un universo, que los contiene y que es la representación del todo, también es conocido como conjunto universal. En la Figura 2.8 se muestra la representación gráfica del conjunto universal, comunmente se denota con la letra $U$.

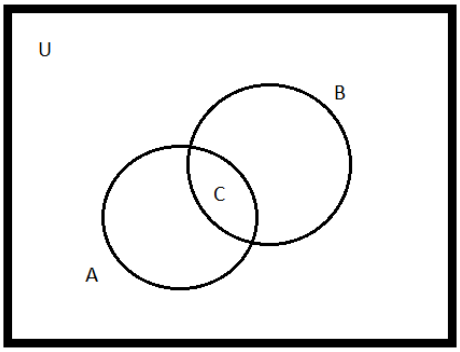


Figura 2.8: Funciones de pertenencia y conjuntos difusos

A comparación de los conjunto clásicos, los conjuntos difusos permiten valores de pertenencia entre 0 y 1, tomando valores como 0.5, 0.3, según el caso. Se puede ejemplificar ésto utilizando dos conjuntos. Un conjunto llamado hombres altos, y otro de hombres bajos. Ambos conjuntos serán representados de manera difusa y clásica, ver Figura 2.9. Si un hombre mide 1.80 [m], pertenecerá al conjunto de los altos con un valor de 1 (caso clásico), y 0 para el hombres bajos. Por otro lado, un hombre que mida 1.80 [m], en un conjunto difuso,

tendrá una pertenencia de 0.88 aproximadamente para el conjunto de los altos y un valor de 0.12 para el de los bajos. Se puede destacar que mientras en el caso clásico la pertenencia es absoluta, en el caso difuso existen valores de pertenencia intermedios entre [0,1] para ambos conjuntos. El cambio de pertenencia entre conjuntos también se le conoce como membresía, se le denota con la letra griega $\mu$.

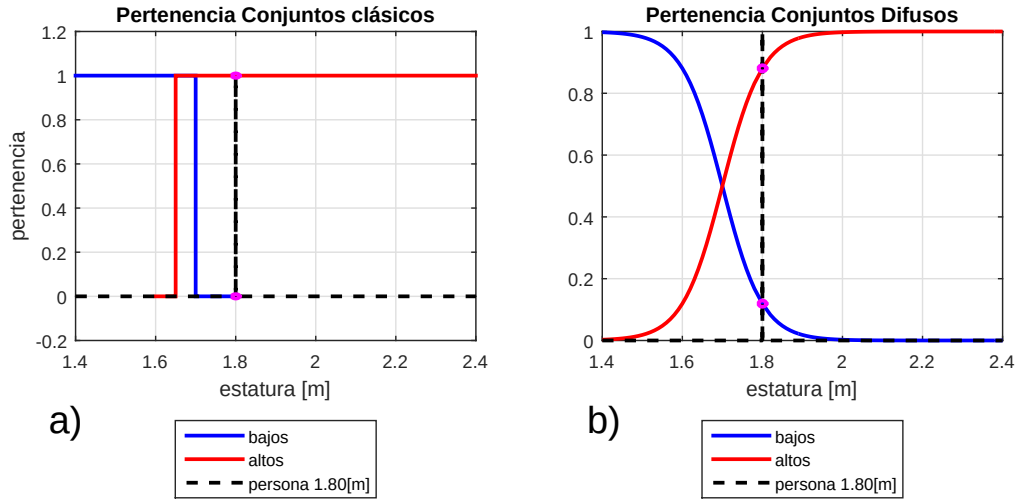


Figura 2.9: a) Pertenencia en conjuntos clásicos, b) pertenencia en conjuntos difusos

Se define a un conjunto difuso como:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{2.18}$$

Dónde $X$, representa el universo de discurso, y $x$, uno de sus elementos. En un conjunto, la pertenencia de un elemento se define como $\mu_A(x)$. Cuando la pertenencia cumple con una forma preestablecida, se le conoce como función de pertenencia.

En la Figura 2.10, se observa una representación gráfica y teórica de como sería un conjunto difuso a comparación de un conjunto clásico.

Los conjuntos difusos se pueden representar de manera continua o discreta. A continuación se presentan ambos casos, y su nomenclatura.

**Caso Continuo:**

$$A = \left\{ \int \left( \frac{\mu_A(x)}{x} \right) \right\} \tag{2.19}$$

**Caso Discreto:**

$$A = \left\{ \sum_i^n \left( \frac{\mu_A(x_i)}{x_i} \right) \right\} = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots + \frac{\mu_A(x_n)}{x_n} \right\} \tag{2.20}$$

Es importante aclarar la nomenclatura de éstos sistemas, el simbolo $+$, se utiliza para la unión, y no para la suma; por otro lado, el simbolo utilizado para la fracción, no define una división, sino una relación entre un elemento del universo de discurso y su pertenencia. El uso de $\sum$, y $\int$, hacen referencia a la unión, no a una suma. Este tipo de notación es utilizada de igual forma en circuitos digitales y en general, en los sistemas lógicos.

En casos de implementación se dará prioridad al uso de conjuntos difusos discretos, debido a que la discretización es una necesidad por el procesamiento utilizando tarjetas electrónicas digitales.

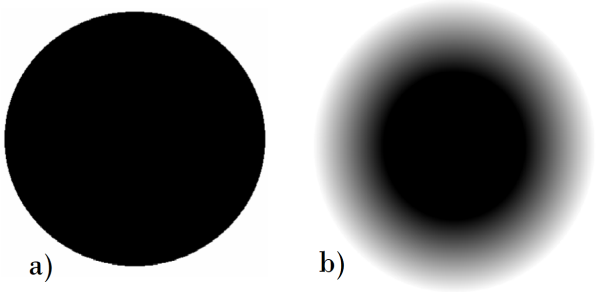


Figura 2.10: Representación gráfica de un conjunto difuso a comparación de un conjunto clásico a) conjunto clásico b) conjunto difuso[14]

### 2.4.3 Funciones de membresia

Como se mencionó anteriormente la membresía es sinonimo de pertencia a un conjunto. Se le conoce como función de membresía a aquella relación entre el grado de pertenencia de un valor dado proveniente del universo de discurso. La función de membresía cuenta con una forma definida matemáticamente. Las funciones de membresía son las encargadas de definir las fronteras de los conjuntos difusos, por ejemplo en la Figura 2.11, se observan 5 conjuntos difusos continuos, cada uno delimitado por el grado de pertenencia de cada uno de sus elementos; es decir, que un conjunto difuso puede ser representado mediante su función de membresia $\mu(x)$. Las funciones pueden ser de diferentes tipos (triangular, trapezoidal, sigmoidal, campana generalizada, etc...), y serán implementadas en los sistemas difusos según el caso de estudio. La forma de las funciones de membresía está estrechamente relacionada con la sensibilidad del sistema difuso. Asumiendo un universo de discurso $X$ y un conjunto difuso $A$, la función de membresía de dicho conjunto difuso se denota como $\mu_A(x)$, dónde $x \in X$ y $\mu_A(x) \rightarrow [0, 1]$.

En el presente trabajo se utilizará la siguiente nomenclatura para abreviar el tipo y la estructura de una función de membresía. Cabe mencioanr que esta nomenclatura es la más comun en lenguajes de progración, tal es el caso de python y MATLAB.

Función de membresía triangular - **trimf [X,(a,b,c)]**

Función de membresía trapezoidal - **trapmf [X,(a,b,c,d)]**

Dónde: **a,b,c y/o d** representan los vértices de la figura geométrica, ya sea triangular o trapezoidal. **X** es el universo de discurso dónde se alojan los conjuntos difusos.


Figura 2.11: Funciones de pertenencia y conjuntos difusos

Función de membresía gaussiana - **gaussmf[X,(a,b)]**

Dónde: **a** es el centro o promedio de la curva, **b** es la desviación estandar al centro de la curva y **X** es el universo de discurso.

Función de membresía campana generalizada - **gbellmf[X,(a,b,c)]**

Dónde: **a** es el parámetro que controla el ancho de la curva, **b** es el centro de la curva, **c** es el cambio en la pendiente en los límites de la curva y **X** es el universo de discurso.

Para una identificación mas apropiada, en la Figura 2.12 se presentan los atributos básicos de toda función de membresía, éstos atributos distinguen los tipos y forma de una función de membresia.

- **Núcleo:** $\{x \in X \mid \mu(x) = 1\}$
- **Frontera:** $\{x \in X \mid 0 < \mu(x) < 1\}$
- **Soporte:** $\{x \in X \mid \mu(x) > 0\}$
- **Puntos de cruce:** $\{x_1, x_2 \in X \mid \mu(x_1) = 0.5\ ,\ \mu(x_2) = 0.5\}$
- **Ancho de banda:** $\mid x_2 - x_1 \mid$

Además de los atributos básicos, las funciones de membresía tienen características morfológicas, éstas son propiedades que distinguen la forma y comportamiento de la membresía. A continuación se describen algunas propiedades. En la Figura 2.13 se encuentran ejemplificaciones de cada una.

- **Normal:** Aquella función de membresía que tiene núcleo; es decir, que el valor de membresia es igual a uno, $\{x \in X \mid \exists \mu(x) = 1\}$.
- **Subnormal:** Aquella función de membresía que no tiene núcleo, es decir que ningún valor de pertenencia alcanza el valor de uno , $\{x \in X \mid \nexists \mu(x) = 1\}$.
- **Simétrica:** Aquella función de membresía que tiene fronteras iguales.
- **No simétrica:** Aquella función de membresía con fronteras de distinta proporción.

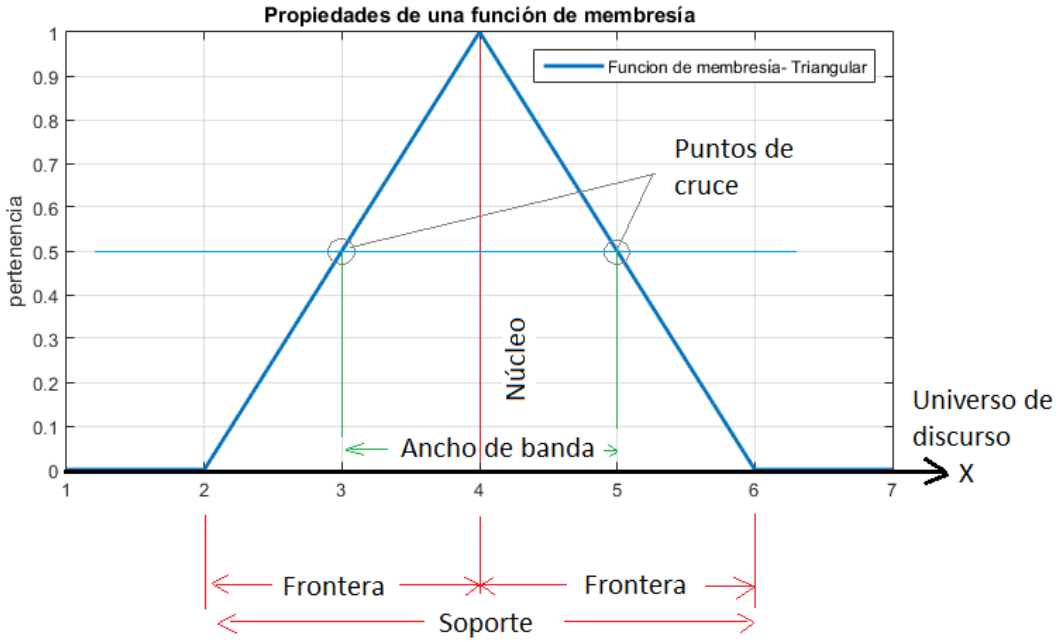


Figura 2.12: Propiedades de una función de membresía

- **Abierta:** Es aquella función de membresía que tiene un soporte con valor de pertenencia uno, existen dos casos, abiertas por la derecha y por la izquierda.
- **Cerrada:** Es aquella función de membresía dónde su soporte siempre se mantiene en cero.
- **Convexa:** Una función de membresía se dice convexa, si en un conjunto $A$ cumple con: $\mu_A(x_1\lambda + (1-\lambda)x_2) \geq min(\mu_A(x_1), \mu_A(x_2))$, dónde $\forall x_1, x_2 \in X$, y $\forall \lambda \in [0,1]$.
- **No convexa:** En caso que no se cumpla, $\mu_A(x_1\lambda + (1-\lambda)x_2) \ngeq min(\mu_A(x_1), \mu_A(x_2))$ , entonces, se dice que la función no es convexa.
- **Singleton:** Se define cuando su soporte solo tiene un elemento.

## 2.5 Razonamiento difuso

Con anterioridad se desarrolló la base matemática de los conjuntos difusos, así como sus generalizaciones. En ésta sección se abordará la relación que existe entre los conjuntos difusos y el lenguaje natural o cotidiano de un humano. Además se definirán las reglas **Si- Entonces**, por último se define el Modus Ponens difuso, el cual es el núcleo principal para los sistemas difusos o controladores difusos.

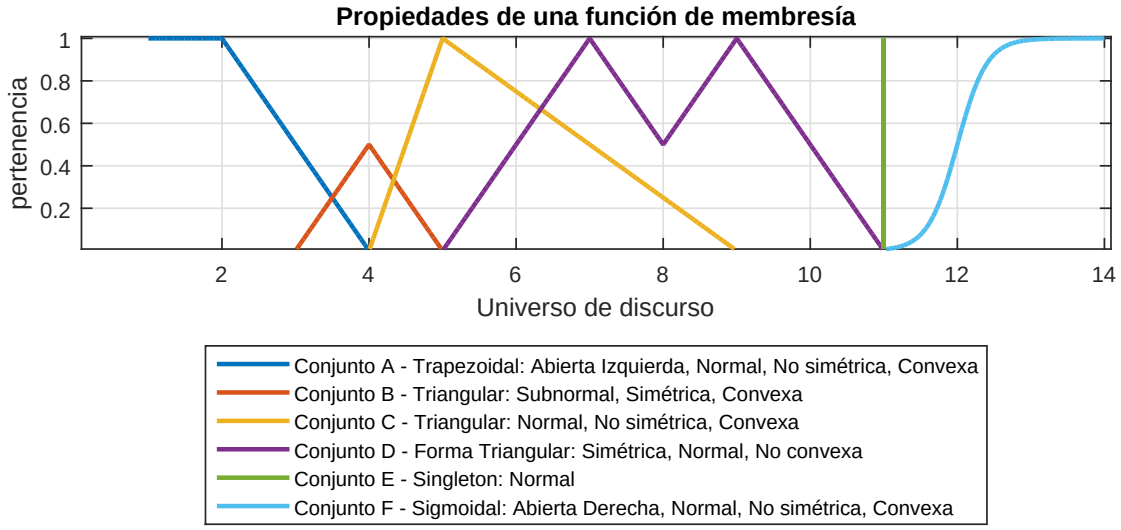


Figura 2.13: Características de las funciones de pertenencia.

### 2.5.1 Variables lingüísticas

Una variable lingüística es una representación del lenguaje natural o humano, que tiene valores difusos, dichos valores estarán definidos con terminología lingüística. Se puede comprender a la variable lingüística como el principio del razonamiento aproximado o difuso.

La terminología lingüística permitirá trasladarse desde términos del lenguaje natural a términos difusos, ésto se hace utilizando conjuntos difusos como representación inmediata de la terminología lingüística.

Una variable lingüística consta de 4 propiedades principales: (v,X,$T_v$,M). Con el uso de éstas es posible definirla con exactitud.

- v: nombre de la variable. Puede tomar cualquier valor dentro del universo de discurso
- X: universo de discurso; es decir, el intervalo de valores en los que la variable lingüística existirá
- $T_v$: valores lingüísticos; es decir, el conjunto de términos lingüísticos que corresponderán a los distintos conjuntos difusos
- M: Regla semántica, relación directa de un termino lingüístico y un conjunto difuso

Por ejemplo la temperatura en un día cualquiera, puede ser definida mediante una variable lingüística. En primera instancia se debe proponer un universo de discurso, ya que éste será donde esté alojado la variable. En un día imaginario, la temperatura está contenida entre 0 y 50 grados Celsius, ntonces:

**v:** Temperatura de un día

**X:** [0,50] grados Celsius

$T_v$**:** $T_v(v) = \{$ Muy frío, Frío , Templado, Cálido, Muy caliente $\}$

**M:**

M(Muy frío) = gaussmf(v;0,10)

M(Frío) = gaussmf(v;0,10)

M(Templado) = gaussmf(v;0,10)

M(Cálido) = gaussmf(v;0,10)

M(Caliente) = gaussmf(v;0,10)

### 2.5.2 Reglas Si-Entonces

Las reglas difusas son una derivación directa del razonamiento difuso y son utilizadas para concluir hechos, las reglas constan de una estructura **Antecedente-Consecuente**. El principal objetivo de éstas es traducir el razonamiento humano o natural al lenguaje utilizado por una maquina.[11] [18]

Algunos ejemplos de éstas reglas son:

**Si** la naranja es dulce, **entonces** el jugo es rico.

**Si** la computadora es lenta, **entonces** el procesador es viejo.

**Si** el hombre es alto, **entonces** su perro es inteligente.

**Si** el jugo de manzana es ácida, **entonces** el puré de manzana es delicioso.

Se puede destacar que no todas las reglas son ciertas pero mantienen la estructura **Antecedente-Consecuente**.

Este razonamiento puede ser también interpretado como:

**Si** x es A, **entonces** y es B

Donde, $x$ y $y$, son variables lingüísticas (cualquier valor del universo de discurso). A y B son valores lingüísticos (la correspondencia de los valores lingüísticos con los conjuntos difusos).

Existen dos formas matemáticas de hacer dicha interpretación:

- Relación difusa: $\mathcal{R} = A \times B$
- Implicación difusa: $I = A \rightarrow B$

Usando al **antecedente**, y al **consecuente**, como variables lógicas, las interpretaciones matemáticas estarán basadas en la tablas de verdad 2.1, y 2.2.

Tabla 2.1: Tabla de verdad Relación difusa

| **Antecedente p** | **Consecuente q** | **Relación difusa** $p \times q = p \wedge q$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tabla 2.2: Tabla de verdad Implicación difusa

| **Antecedente p** | **Consecuente q** | **Implicación difusa** $p \rightarrow q$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

En la relación difusa solo es **verdadera** la conclusión, cuando al **antecedente**, y **consecuente**, son verdaderos. Por otro lado, la tabla de verdad para la implicación difusa muestra que cuando el **consecuente** es falso y el **antecedente**, verdadero, la conclusión es falsa [11][18].

Cada una de las interpretaciones es usada para diferentes propósitos. La relación difusa es utilizada para controladores difusos ya que, cuando la relación entre el **antecedente-consecuente** es muy intensa (cuando ambos son verdaderos), entonces la conclusión es verdadera. La implicación difusa, es utilizada para máquinas de aprendizaje y otro tipo de controladores difusos donde su objetivo es verificar las diferentes posibilidades de veracidad; es decir, sólo cuando el **antecedente** es verdadero y el **consecuente** es falso, la conclusión es falsa. Esto corresponde a un razonamiento orientado a medir la veracidad de los argumentos y no tanto la intensidad de su relación [11] [18].

A continuación se utilizan las dos interpretaciones en un mismo caso para una descripción más precisa. Suponiendo que el cliente de un restaurante hace la siguiente aseveración, “Si el servicio es excelente, entonces la propina es buena”, dónde existen dos variables lingüísticas, el servicio y la propina

**Servicio:**

v: $x_{servicio}$

X: [0,10]

$T_v(v) =$ { Malo, Regular, Excelente }

M(Malo) = sigmf(v;0,10)

M(Regular) = gaussmf(v;0,10)

M(Excelente) = sigmf(v;0,10)

**Propina:**

v: $x_{Propina}$

X: [0,20] %

$T_v(v) =$ { Mala, Regular, Buena}

M(Mala) = sigmf(v;0,10)

M(Regular) = gaussmf(v;0,10)

M(Buena) = sigmf(v;0,10)

La relación y la implicación difusa tomarían el aspecto que se observa en la Figura 2.14. En los costados de la Figura 2.14 están colocados el antecedente y consecuente para cada caso de interpretación con el fin de visualizar el comportamiento de implementar una regla con cada caso [18].

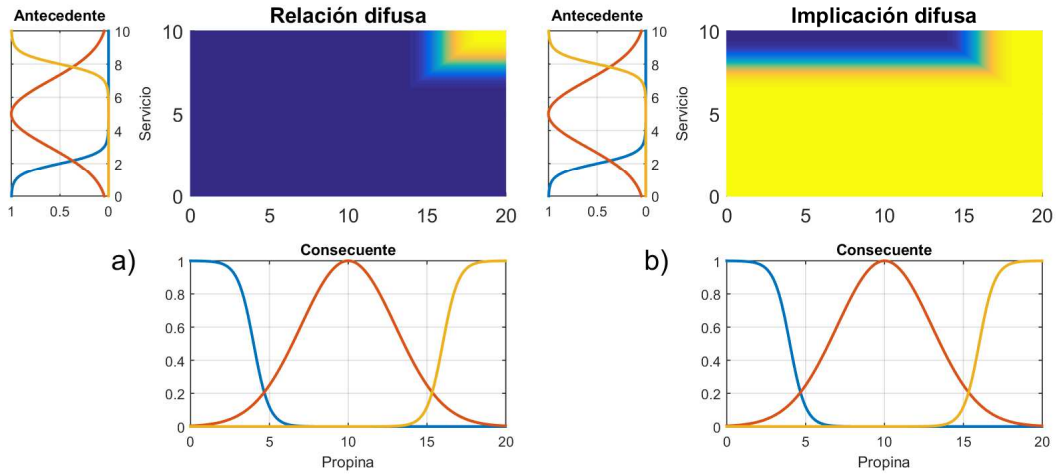


Figura 2.14: Interpretación del razonamiento difuso, a) Relación difusa b) Implicación difusa

### 2.5.3 Modus ponens difuso (inferencia lógica)

En lógica tradicional, el Modus onens (MP, método que al afirmar, afirma), es una inferencia lógica que permite obtener un argumento válido. A partir de un Hecho y una Regla, se presenta una conclusión válida [18].

Formalmente en lógica el MP se le considera un método de separación ya que, elimina la implicación (regla), concluyendo un argumento válido [18].

El MP difuso es el núcleo en el cual están basados los sistemas difusos ya que, generalizando este método se pueden implementar las reglas necesarias para el control de sistema usando inferencia lógica [18].

Regla (Implicación) **P**$\rightarrow$ **Q**

Hecho **P**

---

Conclusión $\therefore$ **Q**

Representando el mismo método en lenguaje natural o humano se tiene que:

Regla - premisa 1 **Si** llueve mucho tiempo, **entonces** mi ropa estará mojada

Hecho - premisa 2 Está lloviendo

---

Conclusión **Por lo tanto**, mi ropa está mojada

Matemáticamente este método de inferencia se representa como:

Regla - premisa 1 **Si** $x$ es $A$, **entonces** $y$ es $B$

Hecho - premisa 2 $x$ es $A'$

---

Conclusión $\therefore y$ es $B'$

Regla - premisa 1 $\mathcal{R} = A \times B$

Hecho - premisa 2 $A'$

---

Conclusión $\therefore B' = A' \circ \mathcal{R}$

Asegurando que $x_0 \in X$, y el conjunto $A'$, es representado mediante una función de pertenencia $\mu_{A'}$ de tipo singleton para obtener un solo dato puntual del universo de discurso, se tiene que el hecho lógico, se obtiene mediante la evaluación $\mu_{A'}(x_0)$.

Regla - premisa 1 $\mu_{\mathcal{R}}(x, y) = \mu_A(x) \wedge \mu_B(y) = T_{min}(\mu_A(x), \mu_B(y)$

Hecho - premisa 2 $x_0 \implies \mu_{A'}(x_0)$

---

Conclusión $\therefore \mu_{B'}(y) = \mu_{A'}(x_0) \circ \mu_{\mathcal{R}}(x, y) = \mu_{A'}(x_0) \circ (\mu_A(x) \wedge \mu_B(y))$

Mediante el uso de operaciones difusas, se puede simplificar de la siguiente manera (usando la composición **max-min**):

$$\mu_{B'}(y) = \bigvee_x [\mu_{A'}(x_0) \wedge \mu_A(x) \wedge \mu_B(y)] = \left\{ \bigvee_x [\mu_{A'}(x_0) \wedge \mu_A(x)] \right\} \wedge \mu_B(y) \tag{2.21}$$

Se observa en la ecuación 2.21, que se puede excluir $\mu_B(y)$, dado que la composición **max-min**, solo afecta a los elementos pertenecientes al universo de discurso $X$, es decir $x_0, x \in X$, y $y \in Y$.

Para simplificar aún más se utiliza la definición de la ecuación 2.22, la cual se obtiene mediante la composición **max-min**. Es evidente que $\mu_{A'}$, siendo un conjunto difuso de tipo singleton evaluado en $x_0$, su valor de pertenencia siempre valdrá 1, ésto permite que cuando $x_0 = x$, y se le aplique el operador $T_{min}$, dé como único resultado $\mu_A$, evaluado en $x_0$. visualmente ésto se puede apreciar en la Figura 2.15.

$$\bigvee_x [\mu_{A'}(x_0) \wedge \mu_A(x_0)] = S_{max_x} [T_{min} [\mu_{A'}(x_0), \mu_A(x_0)]] = \mu_A(x_0) \tag{2.22}$$

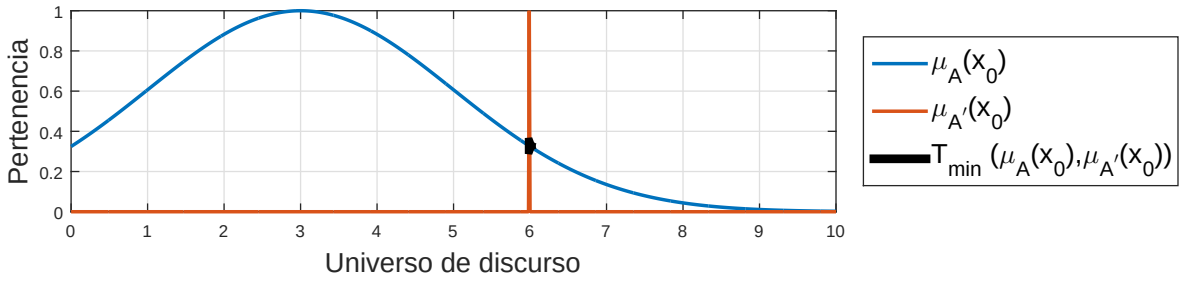


Figura 2.15: Descripción gráfica $T_{min}(\mu_{A'}(x_0), \mu_A(x_0))$

Finalmente la conclusión del MP, es representada mediante la ecuación 2.23. La conclusión simplificada del MP difuso, es esencial para el calculo en computadoras, haciendo más eficiente en cuanto a procesamiento.

$$\mu_{B'}(y) = \mu_A(x_0) \wedge \mu_B(y) = T_{min}(\mu_A(x_0), \mu_B(y)) \tag{2.23}$$

## 2.6 Control

### 2.6.1 Antecedentes

Se tiene conocimiento que los primeros inventos utilizados en la Grecia clásica utilizan algunos elementos de control, por ejemplo relojes de agua. Estos relojes mantenían un nivel predefinido. En ésta época no

se tenían elementos matemáticos para el modelado de sistemas por lo que, no hubo incursión en este campo[47].

El primer avance ingenieril tecnológico surgió hasta 1789 con la invención de James Watt y su regulador de velocidad para una maquina de vapor. El regulador de velocidad mostró defectos, en ocasiones la velocidad de la maquina oscilaba o crecía sin limite[47].

Para determinar las condiciones de un funcionamiento estable, entre 1826 y 1851 J.V. Poncelet y G.B. Airy mostraron que era posible utilizar ecuaciones diferenciales para a la maquina de vapor junto con el regulador de velocidad. En esas fechas los matemáticos definían el comportamiento de una ecuación diferencial mediante sus soluciones [47].

Fue hasta 1868 que J. C. Maxwell mostró la manera de determinar la estabilidad de una maquina de vapor usando nada mas que los coeficientes de la ecuación diferencial asociada a la maquina. Este método se utiliza para ecuaciones diferenciales de orden no mayor a 4, por su complejidad.

A finales del siglo XIX E.J. Routh y A. Hurwitz de manera independiente dedujeron varias formas practicas de conocer la estabilidad. En 1922 N. Minorsky presentó sistemas para control de posición y formuló lo que actualmente se le conoce como controlador PID [47].

### 2.6.2 Definiciones básicas

**Sistema**

Es un compendio de elementos o componentes que actuan conjuntamente para lograr un proceso específico. El concepto de sistema es aplicable a conceptos teóricos o físicos [47].

**Planta**

Una planta se define como el conjunto de piezas de una maquinaria que funcionan juntas, para lograr alguna actividad en concreto. En sistemas de control se entiende por planta, el sistema que se desea controlar [47].

**Variable de entrada**

Es la variable asociada a la planta que al variar su magnitud altera el estado del sistema [47].

**Variable de salida**

Es la variable que puede ser medida (cumple con la condición de poder ser medida) [47].

**Perturbación**

Existen dos tipos de perturbación: externa e interna. Las perturbaciones externas se pueden definir como todos los agentes externos que afectan el funcionamiento del sistema; es decir, que no permiten cumplir su objetivo. Las perturbaciones internas son todas aquellas que ocurren en el sistema, pueden deberse a los elementos que componen al sistema, un ejemplo es la fricción de un balero en alguna junta. Si el balero no está lubricado tenderá a sumar fuerzas de fricción al sistema y puede evitar su buen funcionamiento [47].

**Variable controlada**

Es la cantidad o condición que se mide y controla. Normalmente la variable controlada es considerada la salida del sistema de control cerrado [47].

**Variable manipulada**

Es la variable o condición de la planta que se modifica a fin de influir sobre la variable controlada a través de la dinámica de la planta [47].

**Sistema de control**

Un sistema de control es un compendio de elementos que sirven en conjunto para alcanzar algún objetivo especifico. Algunos de los objetivos de control son:

- **Estabilización:** Cuando el valor deseado de la variable a controlar es cero y el error en estado permanente tiende a cero.
- **Regulación:** Cuando la referencia de la variable a controlar es diferente de cero y el error debe estabilizarse en estado permanente(tender o ser cero).
- **Seguimiento:** Cuando la variable a controlar debe "seguirüna referencia, con respecto al tiempo (trayectoria) y la dinámica del error tiende a cero.

Existen otros objetivos de control, como la robustez, o el desempeño optimo, que pueden ser adquiridos por un sistema según el diseño del controlador utilizado.

### 2.6.3 Control en lazo abierto

Los sistemas de control de lazo abierto son aquellos donde la variable controlada (salida del sistema) no tiene ningún efecto en la acción de control (variable de control). Estos sistemas son comúnmente diseñados para realizar operaciones secuenciales (operados por bases de tiempo), es decir que no existe ninguna regulación de las variables [47].

Algunos ejemplos de estos sistemas son las lavadoras, los tostadores, los semáforos. Las lavadoras por su parte no les interesa cuanta suciedad tenga la ropa (ni el jabón con el que tenga que quitar esa suciedad), sino más bien, funcionará respecto a su calibración y programación. Los tostadores son sistemas muy simples que funcionan al ingresar un pan y las resistencias disipan calor hasta cierto punto de temperatura preestablecido por una perilla o selector. El tostador no tendrá capacidad para distinguir por ejemplo el color del pan cocido necesario para asegurar que no se tueste de mas o menos (funciona solo secuencial mente). Por otro lado, un semáforo es un sistema de control secuencial que no distingue entre una calle vacía o con mucho transito, por lo que no podrá activarse conforme al flujo vehicular para generar un transito mas fluido.

**Simplicidad (Funcionamiento preestablecido)**

Un sistema de control en lazo abierto al no requerir comparación entre cantidades no tiene la dependencia de medir la variable de salida, por lo que se puede prescindir del uso de sensores, haciendo el sistema mucho más fácil de implementar y desarrollar.

**Precisión**

En los sistemas de control de lazo abierto la precisión estará definida unicamente por el diseño del sistema de control (su calibración). En un contexto ideal (no existen perturbaciones) los controladores de lazo abierto funcionan bien. Sin embargo, en la realidad estos controladores en ocasiones no son nada precisos debido a las perturbaciones externas o internas.

**Perturbaciones**

Los controladores de lazo abierto están a merced de las perturbaciones, sin embargo se puede asegurar su funcionamiento mediante el uso de componentes que soporten las adversidades, como la intemperie o los esfuerzos. Un caso clásico es la lavadora, que sigue funcioanndo aún al aumentar la carga. El que siga funcionando o se dañe, depende de que los componentes internos soporten las perturbación.

En la figura 2.16 se muestra el diagrama básico de control en lazo abierto:

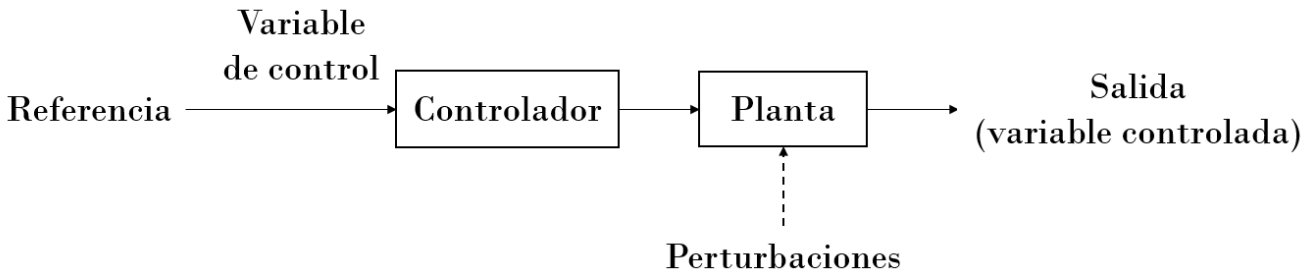


Figura 2.16: Diagrama de bloques del control en lazo abierto

### 2.6.4 Control en lazo cerrado

Los sistemas de control en lazo cerrado se pueden definir como aquellos sistemas donde la señal de salida o variable controlada tiene un efecto directo en la acción de control. El efecto sobre la acción de control se hace mediante la comparación de la salida del sistema y una referencia preestablecida. La acción de control debe ser capaz de corregir el error (comparación) en el sistema. La implementación de los objetivos de control previamente definidos dependera de la tarea u objetivo que deba cumplir el sistema de control.

En el ámbito del control automático es común que los sistemas de control en lazo cerrado ocupen la función de transferencia como medio para el diseño de controladores. La función de transferencia se obtiene mediante el uso de la transformada de Laplace, transformando la ecuación o las ecuaciones diferenciales, del sistema en cuestión, a ecuaciones algebraicas facilitando así, el proceso de análisis. Considere una función de transferencia como se muestra en la ecuación 2.24, la función puede ser obtenida a partir de la reducción de bloques de la Figura 2.17.

$$T(s) = \frac{Y(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{2.24}$$

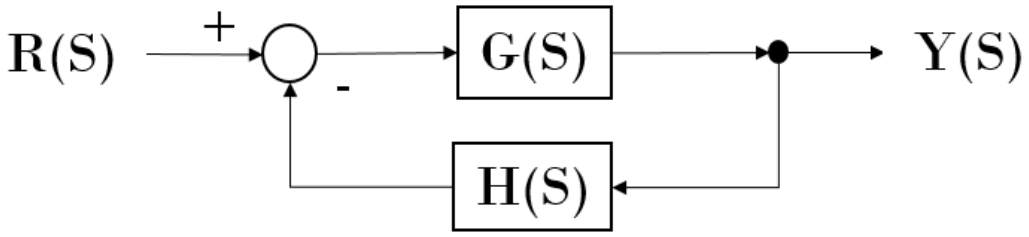


Figura 2.17: Diagrama de bloques, retroalimentación positiva

Entonces el problema de diseño será elegir convenientemente los polos de la función de transferencia (ecuación 2.25), ya que éstos permitirán manipular las características dinámicas del sistema, ademas de asegurar la estabilidad si los polos tienen parte real negativa. Existen criterios de selección para la locación de los polos como, ruth-hurwitz y ruth-locus .

$$1 + G(s)H(s) = 0 \tag{2.25}$$

**Complejidad del controlador**

Como se observa en la Figura 2.17, un controlador de lazo cerrado se caracteriza por la existencia de comparación (retroalimetnación) entre la salida y la entrada. Esto da pie a que se conozca la magnitud exacta con la que el sistema se desvía de una referencia preestablecida; es decir, el error. Para hacer ésta comparación entre salida y entrada. Es necesario el uso de sensores que midan las variables físicas del sistema. El uso de sensores, hace más complejo el diseño e implementación del controlador, así como costoso.

**Desempeño**

El desempeño de funcionamiento en los controladores de lazo cerrado puede ser modificado mediante parámetros de diseño. Son aquellos parámetros que tienen un efecto directo en el sistema a través de los polos de la función de transferencia en lazo cerrado mostrada en la ecuación 2.24, permitiendo un comportamiento deseado.

El tiempo de asentamiento es uno de los parámetros más utilizados ya que permite aumentar o reducir el tiempo que se tarda en pasar del periodo de transición al periodo en estado permanente.

**Perturbaciones**

Un controlador en lazo cerrado es robusto ante perturbaciones. Dado que se cuenta con la magnitud del error, es posible generar acciones de control que estabilicen al error. Las ganancias utilizadas en los controladores juegan un papel esencial en la robustez y la estabilidad del sistema. En el caso de un controlador proporcional, una ganancia pequeña hará que el controlador se tarde mucho en llegar al valor de referencia, pero tendrá una mayor precisión. Una ganancia media tendrá un efecto oscilatorio cerca del punto de referencia. Por otro lado una ganancia muy grande hará que el sistema se desestabilice.

La Figura 2.18 muestra el diagramas de bloques correspondiente a la implementación de un lazo de control cerrado, se detalla que de la planta deben obtenerse las variables a ser comparadas a través de un

sensor o sensores. Además la planta es susceptible a perturbaciones externas y/o internas.

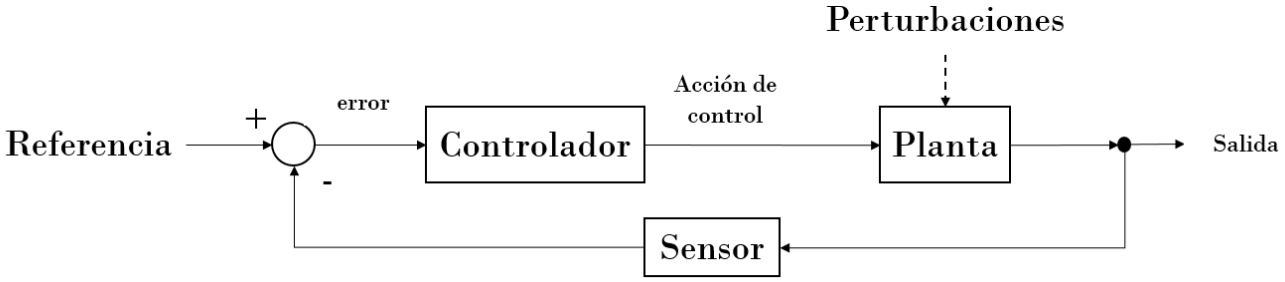


Figura 2.18: Diagrama de bloques, esquema de controlador en lazo cerrado

### 2.6.5 Control PID

El controlador PID es un tipo de controlador inicialmente basado en modelo matemático, con una estructura simple en lazo cerrado. Está compuesto por tres acciones de control: proporcional - eP, integral - eI y derivativa - eD. Permite diseñar un controlador basado en parámetros de diseño, como pueden ser el tiempo de asentamiento o el sobrepaso deseado.

El tener una arquitectura de lazo cerrado le da robustez ante perturbaciones. Cada acción de control (P,I,D) consta de una ganancia, normalmente se denotan como: $K_p$,$K_i$ y $K_d$. En la Figura 2.19 se muestra la estructura de dicho controlador. La acción de control total del control PID es la suma se las tres acciones antes mencionadas, de manera continua se describe mediante la ecuación (2.26), cuando se le aplica la transformada de Laplace se tiene la ecuación (2.27).

Cuando se tiene un sistema físico de orden dos representado por una función de transferencia del estilo de la ecuación (2.24), es posible obtener las ganancias mediante su ecuación característica asociada al sistema igualada a cero, igualando y agrupando términos semejantes de esta ecuación respecto otra ecuación prediseñada por medio de parámetros de diseño; por ejemplo, el factor de amortiguamiento.

$$u(t) = K_p\left(e(t)\right) + K_i \int_0^t e\left(\tau\right)d\tau + K_d\left(\dot{e}(t)\right) \tag{2.26}$$

$$\alpha(s) = K_p + K_i\left(\frac{1}{s}\right) + K_d\left(s\right) \tag{2.27}$$

A pesar de ser un controlador basado en modelo matemático, se ha sabido aprovechar sin un modelo definido, sintonizando sus ganancias por distintos medios: uno de ellos es el método de Ziegler-Nichols el cual sintoniza las ganancias del controlador mediante la gráfica de mediciones de la entrada y salida del sistema. Otro ejemplo es el control PID-ASLD, éste se discutirá posteriormente en el Capitulo 4.

Las ganancias son consideradas como agentes que magnifican o reducen el efecto de la parte proporcional ($K_p$), la parte integral ($K_i$) y la parte derivativa ($K_d$). Enseguida se presenta la descripción de cada una de las acciones de control y los efectos que que tienen en el sistema de control.

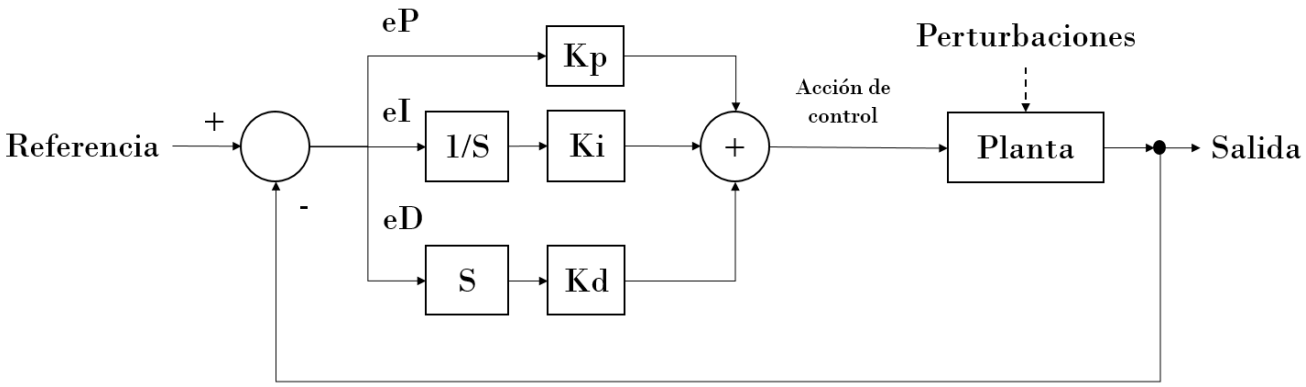


Figura 2.19: Diagrama de bloques de un controlador PID

**Acción proporcional - P**

Es la encargada de acercar al sistema al punto de referencia; sin embargo, esta acción tiene un comportamiento oscilatorio, el cual solo se acerca al punto de referencia pero no llega a él. Es oscilatorio ya que por más pequeño que sea el valor del error (comparación entre la referencia y el valor medido), la acción de control tenderá a ir en sentido opuesto a esta comparación, con el fin de compensar y llegar al valor de referencia, lo que tendrá como consecuencia que oscile perpetuamente alrededor de la referencia.

**Acción integral - I**

Se encarga de alcanzar el punto de referencia de manera perfecta (idealmente), dado que está definida por la integral del error, esto hace que el valor de la integral vaya creciendo conforme se aleje del punto de referencia, y la acción de control sea cada vez mas grande. Así que por esto, se elige la ganancia $K_i$ dentro de un intervalo relativamente cercano al punto deseado. Los sistemas que incorporan una acción integral sufren de un atraso en el tiempo de asentamiento, por lo que puede no se cumpla con el tiempo de asentamiento preestablecido en los parámetros de diseño.

**Acción derivativa - D**

Al estar definida por el cambio del error agiliza al sistema, permitiendo que su tiempo de asentamiento disminuya y de manera similar a la acción proporcional, permite acercarse al punto de referencia rápidamente. Sin embargo introduce ruido en el sistema al derivar error, esto puede llevar al mal funcionamiento del controlador.

## 2.7 Controlador difuso tipo Mamdani

Como se muestra en el estado del arte, la idea principal de un controlador difuso es funcionar igual o mejor que un controlador basado en modelo. Su estructura es en lazo cerrado, esto indica que tiene como variable de entrada al error, al igual que un controlador tradicional. El error es ocupado para tomar decisiones en la obtención de una salida de control hacía la planta, con el fin de obtener una salida deseada. Actualmente este controlador es aplicado en muchos campos de estudio, debido a que proporciona

una solución sencilla a problemas complejos. El interés surge de la reducción de costos y mejoras en el desempeño.

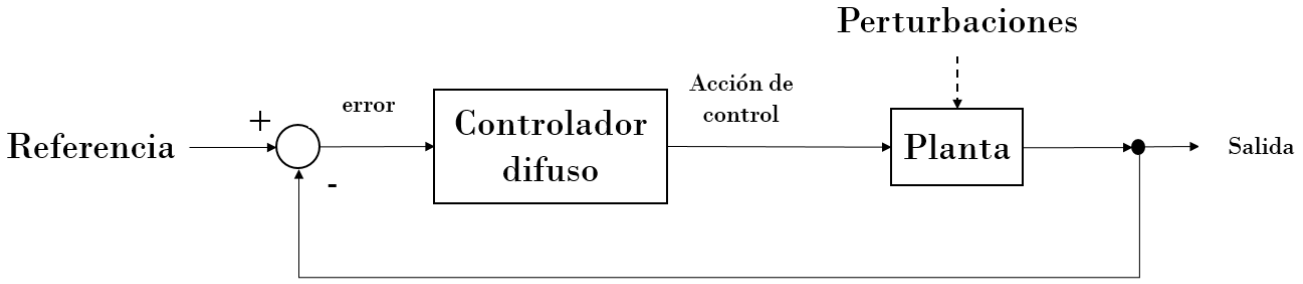


Figura 2.20: Diagrama de bloques de un controlador difuso

Algunas de las aplicaciones industriales más conocidas son: control de balanceo en grúas, control de llama en plantas de incineración de residuos, control de dosificación en plantas de tratamiento de aguas negras, control de robots en inspección de túneles, control de posición en prensas, control de temperatura en máquinas de modelado plástico, control de clima y automatización de edificios, control en conversores de generadores eólicos, entre otras aplicaciones.

El controlador difuso es una alternativa práctica a los controladores convencionales, ya que permite diseñar e implementar un controlador para sistemas no lineales mediante conocimiento humano de manera relativamente sencilla [11]. En concreto se salta la parte del diseño matemático riguroso y es sustituido por la toma de decisiones basadas en el agente experto, haciendo el proceso mucho más sencillo de concretar.

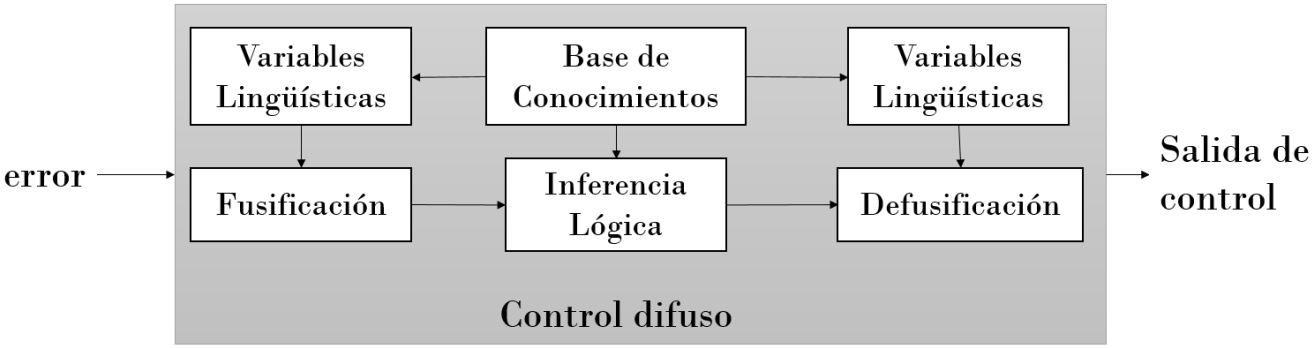


Figura 2.21: Esquema conceptual interno de un controlador difuso de tipo Mamdani

Un controlador difuso tipo Mamdani consta de cuatro secciones: Base de conocimientos, Fusificación, Inferencia lógica, y defusificación (ver Figura D.1). Las reglas utilizadas para la inferencia lógica o toma de decisiones tienen la forma **Si (ANTECEDENTE) Entonces (CONSECUENTE)**. Este controlador cuenta con variantes que surgen de las diferentes operaciones existentes en los operadores generalizados difusos (ver Apéndice C), sin embargo se mantiene su estructura intacta, por lo que puede ser replicado.

Un controlador difuso puede imitar el *modus operandi* de otros controladores: on/off, P, PD, PID, retroalimentación de estados, entre otros, lo único que necesita es un compendio de reglas que le digan que hacer, entre más reglas haya, habrá más precisión en el comportamiento.

Las desventajas del control difuso radica en la mezcla de tres conceptos: robustez, precisión, y velocidad. Mientras más robustez y precisión se requieran para un controlador, se necesitarán más reglas de control, éso aumenta el tiempo de calculo y por lo tanto disminuye la velocidad de respuesta. Por otro lado si el sistema tiene implementado un controlador con pocas reglas de control, gozará de excelente velocidad de respuesta pero la precisión y robustez menguarán.

En conclusión, para la aplicación de un control difuso deben establecerse parámetros de diseño, que permitan a la inferencia lógica y la base de conocimientos tomar las mejores decisiones. Considerar además la velocidad, precisión y robustez del controlador.

Para una descripción más detallada de cada una de las secciones, así como el diseño basado en un ejemplo práctico de control ir al Apéndice D.

## 2.8 Control PID-ASLD

Un controlador PID-ASLD se puede considerar un controlador PID que cambia sus ganancias dependiendo del error, cambio del error y la suma del error; es decir, la autosintonización tendrá el objetivo de calcular las ganancias $K_p$,$K_i$ y $K_d$ más adecuadas según el caso. La acción de control está estructurada como un control PID tradicional, ver ecuación (2.26) [11].

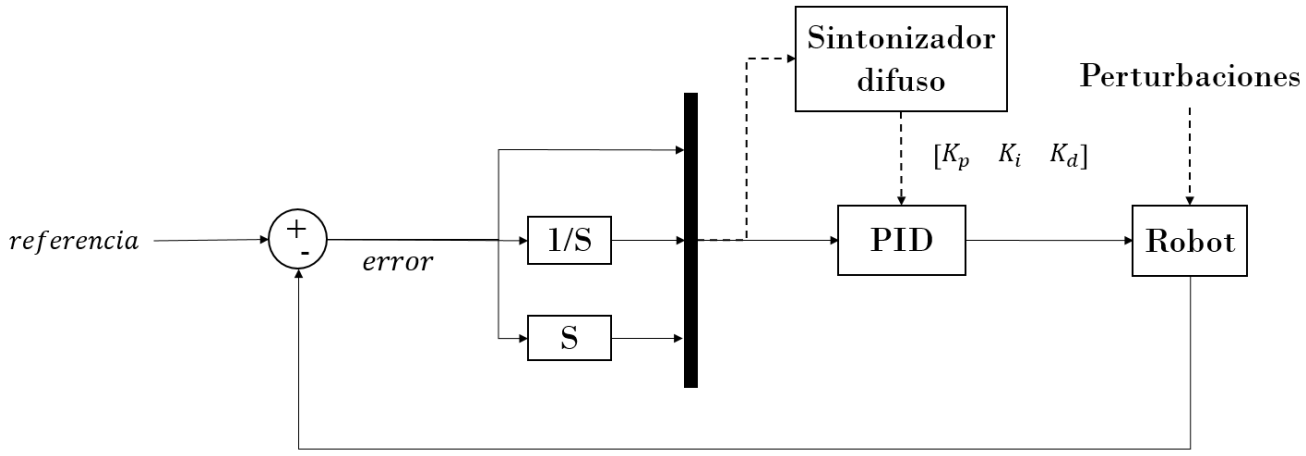


Figura 2.22: Control PID autosintonizable

Vale la pena considerar que el control PID-ASLD no es el mismo que un controlador PID difuso (PID-LD). En contraste al control PID-ASLD, el control PID-LD genera una acción de control basado en el error, el cambio del error y la suma del error imitando la respuesta de un controlador PID tradicional. La acción de control está directamente ligada al sistema por lo que no hay ganancias involucradas.

El objetivo de autosintonizar las ganancias con un control difuso es para poder tener un comportamiento adaptable que cambie con base en las entradas al sistema, con esto se logra un aumento en la velocidad de respuesta, además la autosintonización permite que el controlador sea robusto a perturbaciones externas. Un control PID-ASLD es equiparable a un control de ganancias preprogramadas, con la diferencia que las ganancias están calculada en tiempo real y no almacendas en memoria, sin embargo el concepto se conserva [44].

Las reglas de control están orientadas al comportamiento deseado de las ganancias $K_p$, $K_i$ y $K_d$, respecto a las entradas del sistema difuso. Las entradas al sistema difuso se denominan eP, eI y eD, para simplificar la descripción (error, integral del error, derivada del error), respectivamente.

En la Figura 2.23 se muestran las partes esenciales del sintonizador difuso, se destaca la similitud con un control difuso tipo mamdani clásico, la diferencia radica en que el objetivo del control difuso estará orientado a la determinación de las ganancias, y no en una acción de control que vaya directamente a la planta.

Las variables lingüísticas para eP, eI y eD, deben ser tomadas directamente del sistema físico, se debe definir el universo de discurso de cada parte haciendo mediciones en el sistema. Las variables lingüísticas de las ganancias pueden ser obtenidas experimentalmente. Por medio del, se puede utilizar el método de oscilaciones de Ziegler- Nichols se pueden obtener rangos aproximados de operación.

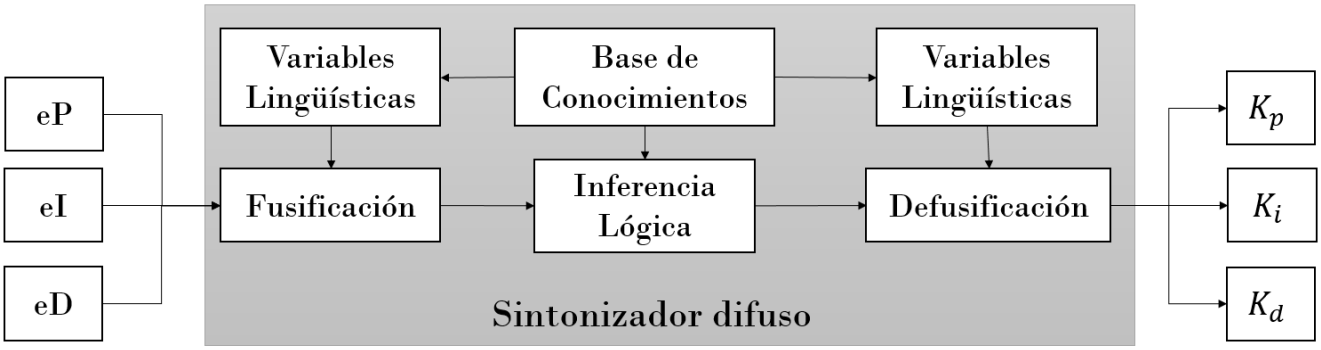


Figura 2.23: Sintonizador difuso para el PID autosintonizable

# Capítulo 3

# Descripción robot bípedo Scout

## 3.1 Arquitectura

El robot bípedo Scout fue diseñado por Lynuxmotion[1] ®, consta de 12 articulaciones, 6 en cada pierna. Las dos piernas están unidas por una cadera denominada eslabón central, en total el robot tiene 13 eslabones. Cada una de sus articulaciones está actuada mediante servomotores.

En la figura 3.1 se muestra la composición del robot, de color rojo se muestran los servomotores. De color negro se tiene el eslabón *B*, este eslabón se identifica como la cadera del robot en conjunto con los eslabones amarillos y azules fuerte. Se tiene que el eslabón "B"no solo sirve como cadera sino también como soporte al sistema embebido al mantener un pequeño espacio en forma de gabinete.

Las rodillas están definidas por cuatro eslabones, dos de color azul claro, destinadas a la parte superior, y dos eslabones de color azul cielo, destinadas a la parte inferior.

Los tobillos constan de cuatro eslabones, dos de color café claro, destinados a la unión de la rodilla con el tobillo y dos eslabones de color morado, estos últimos emulan el efector final de un robot serial, en este lugar se tienen los pies del robot.

| Características robot bípedo Scout | |
|---|---|
| **Altura** | 35 [cm] |
| **Peso: robot + sistema electrónico** | 1.3 [Kg] |
| **Grados de Libertad** | 12 , 6 en cada pierna |
| **Alimentación** | 5 a 6 [v] |

[1]El robot puede ser localizado en: http://www.lynxmotion.com/c-67-scout.aspx

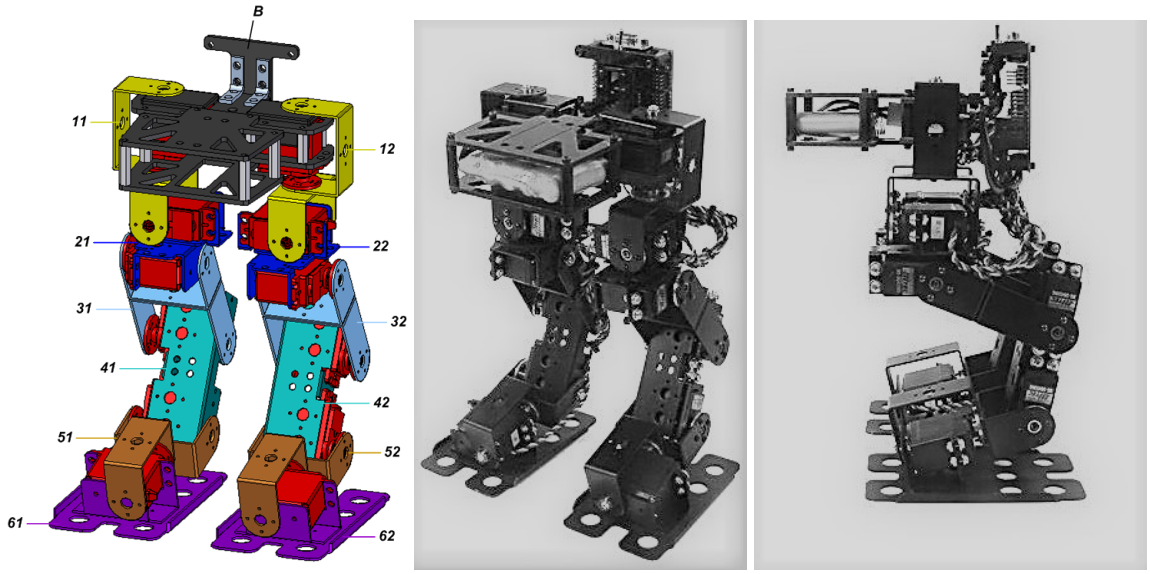


Figura 3.1: Robot Bípedo Scout ®

La Figura 3.2 muestra la convención ángulos usados para cada una de las juntas, de hecho se ocupa el mismo sistema de nomenclatura que en los eslabones de la Figura 3.2. Con ésta figura se puede hacer una descripción mucho mas detallada del robot bípedo.

La siguiente tabla muestra la organización de juntas del robot bípedo. Para hacer la comparación Humano - Robot si se considera un grupo de juntas como una articulación humana. En el robot bípedo se tienen tres grupos de juntas: la cadera, la rodilla y el tobillo.

En la Tabla 3.1 se muestra cada una de las direcciones angulares donde los servomotores operan, la configuración toma en cuenta la dirección mostrada en la Figura 3.2.

Tabla 3.1: Descripción de juntas Robot Bípedo Scout

| **Articulaciones robot bípedo Scout** | | | **Dirección** |
|---|---|---|---|
| **Cadera** | Servomotor 11 | Servomotor 12 | Yaw |
| | Servomotor 21 | Servomotor 22 | Roll |
| | Servomotor 31 | Servomotor 32 | Pitch |
| **Rodilla** | Servomotor 41 | Servomotor 42 | Pitch |
| **Tobillo** | Servomotor 51 | Servomotor 52 | Pitch |
| | Servomotor 61 | Servomotor 62 | Roll |

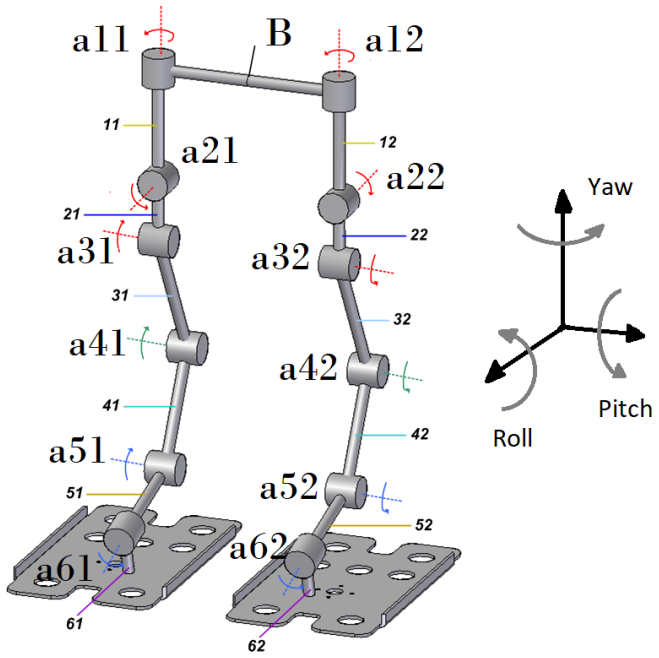


Figura 3.2: Ángulos y eslabones del Robot Bípedo Scout ®[1]

## 3.2 Renovación y mejoramiento del robot bípedo Scout

En el trabajo "*Renovación y mejoramiento de actuadores e instrumentación de un robot bípedo*", realizado por Saddan Hernández [30], se elaboró la restauración y caracterización de los elementos electrónicos para una posterior implementación del control en lazo cerrado. Con anterioridad el robot había sido objeto de trabajo por lo que sufrió desgaste. Para optimizar el error de medición de los elementos electrónicos era inevitable su renovación.

En [30] se cambiaron los actuadores, por unos que brindaran mejores características. Los sensores FlexiForce® antes utilizados se renovaron, utilizando en esta ocasión 8 sensores por cada pie[2]. Al colocar 8 sensores se da una lectura más precisa de las coordenadas del ZMP. Se rediseñó la tarjeta de filtrado para todos los sensores del robot, además de la caracterización de cada uno de los sensores, así como de los actuadores, dando lugar a una gran disminución en el error de medida.

Uno de los aspectos importantes, es el cambio la Unidad de Medición Inercial (IMU). La unidad anterior, dió resultados con un error grande debido al ruido. Se decidió implementar la unidad UM7-lt, ya que esta contiene un filtro de Kalman embebido, por lo que la transmisión de datos es rápida y confiable. Otro de los beneficios de la unidad UM7-lt es su comunicación, ya que permite dos salidas seriales de tipo UART (Rx, Tx) y un bus de comunicación SPI.

En [30] se contempló un punto de partida para el control de lazo cerrado. Se examinó el control de postura utilizando un control PID ubicado en los tobillos del robot. En este caso de estudio, se demostró que era posible utilizar un controlador del tipo PID para el control de postura.

[2]El ZMP será el criterio utilizado para la verificación de la estabilidad.

En [30] se usaba el microcontrolador Arduino Mega para la transmisión de datos al robot. Se propuso como trabajo futuro el uso de tarjetas con mejor procesamiento para un lazo de control cerrado. Dado que la carga de procesamiento en ese momento .

El robot era manipulado por medio de una interfaz gráfica programada en LabView. Esta interfaz tenia dos finalidades: el movimiento de los servomotores de manera automática por medio de *sliders* y la transmisión de ángulos teóricos correspondientes a las trayectorias generadas en [29].

## 3.3 Necesidades para la implementación del control en lazo cerrado

Para poder implementar un sistema de control de lazo cerrado como se planteó en el objetivo general del presente trabajo, es necesario acelerar la lectura de la trayectoria y el envío de datos hacia el robot. Se requiere la lectura de uno o muchos sensores para cerrar el lazo de control, así que el nuevo sistema deberá contemplar este nuevo procesamiento de información. Otro aspecto importante en el desarrollo del nuevo sistema es la velocidad de procesamiento de los microcontroladores utilizados, si bien el Arduino MEGA tiene un reloj de 25 MHz, se requiere de una velocidad más grande de procesamiento, como la TivaC (72 Mhz) o el microntrolador Teensy 3.1( hasta 96 Mhz en overclock). Los cuales fueron actualmente adquiridos para su uso en el robot bípedo.

Para el proceso de información es requerida una microcomputadora como unidad central de procesamiento (CPU), capaz de calcular, recibir y transmitir información para el sistema de control en lazo cerrado. Dado que el CPU tendrá el puesto dominante de procesamiento se requiere definirle como maestro, haciendo a los demás sistemas esclavos. En conclusión, se requiere un sistema embebido, con la finalidad del control en lazo cerrado, capaz de adquirir y enviar información de una manera eficiente.

Las principales ventajas del diseño de este nuevo sistema radican en la velocidad de procesamiento y en la autonomía. Con este nuevo sistema se podría pensar en la implementación de diversos controladores utilizándolo como plataforma de pruebas; es decir, que el usuario se dedicará a la programación y no a la electrónica. La autonomía del robot se vería aumentada por el hecho de tener una computadora embebida; sin embargo, existen diversos conflictos, como la energía requerida del sistema, actualmente se utiliza una fuente fija de 5 volts, sin posibilidad de cambio a una pila por el consumo de corriente mínimo y el peso que ésta representaría si se coloca.

# Capítulo 4

# Diseño de los controladores difusos

## 4.1 Esquema de control para el robot bípedo Scout

Un robot bípedo es un sistema complejo y altamente no lineal, una solución para realizar su control es seccionar los problemas a controlar. Se propone dividir en dos secciones el esquema de control: la postura en soporte doble y la marcha. Al tener estas dos secciones el problema de control se puede hacer más sencillo, ya que evita que la unidad central de procesamiento se sobrecargue de trabajo y el control del robot se vuelve una tarea secuencial de controladores.

El control de postura permitirá al robot reposicionarse si hubiese perturbaciones externas, como el cambio de pendiente en dirección pitch o en dirección roll, las más conflictivas justo antes de iniciar la caminata, esto ayuda en los casos en los que la caminata no inicie desde una postura donde, idealmente el suelo sea plano. Con la misma lógica, al final de la caminata el control de postura hará que el robot termine en una postura estable; de tal forma que si se repitiera el esquema de control, se pueda lograr un ciclo de marcha estable cuantas veces sea requerida.

En la Tabla 4.1 se muestra el esquema de control seccionado utilizado para el robot bípedo. En las siguientes secciones se describirá a detalle el diseño del control de postura y marcha.

Tabla 4.1: Particiones de control para el robot bípedo Scout

| | **Objetivo control postura** | **Objetivo control marcha** |
|---|---|---|
| **Aplicación** | Unicamente en soporte doble | En soporte doble y simple |
| **Control PID autosintonizable dirección pitch** | Mantener una postura deseada en el ángulo de referencia | Mantener una caminata estable alrededor de un ángulo de referencia |
| **Control PID autosintonizable dirección roll** | Mantener el centro de masa en una posición estable, usando cadera y tobillos | Mantener el centro de masa en una posición estable, usando cadera y tobillos |
| **Control difuso de postura dirección roll** | Establecer una relación entre el ángulo roll y la distancia de cadera a tobillos para mantener el centro de masa en una posición estable | |

En la Figura 4.1, se observa el esquema de control del robot bípedo en dirección pitch (plano sagital), idealmente se muestra el comportamiento del controlador bajo cambios de pendiente.

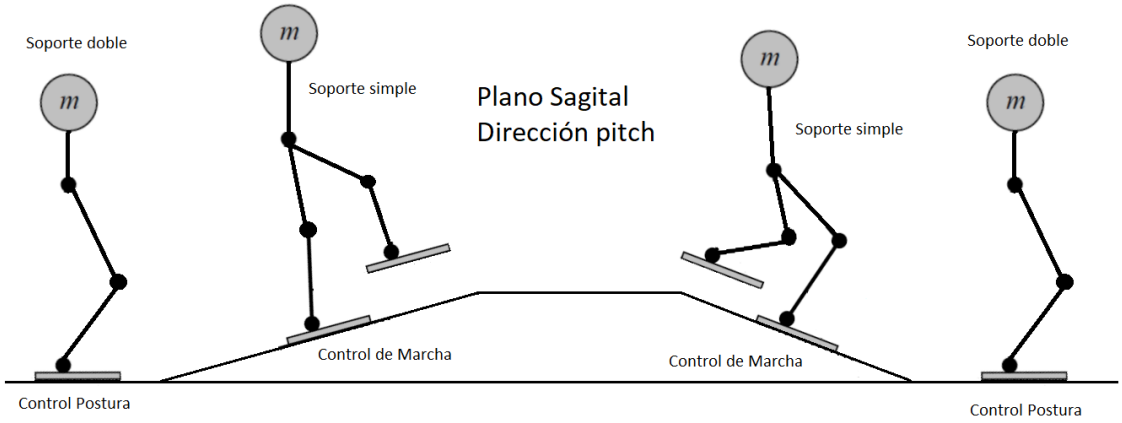


Figura 4.1: Esquema de control del robot bípedo dirección pitch (Esquema basado en [23])

## 4.2 Control de postura

En esta sección se desarrollará a detalle el control de postura para el robot bípedo Scout. Previamente se introdujo el esquema de control en la Tabla 4.1. Como se mencionó previamente, el control de postura tendrá como objetivo mantener el centro de masa en una posición estable, asegurando así una postura

estable, con el fin de lograr lo anterior se debe controlar la posición de la cadera con el uso de las extremidades existentes.

La solución propuesta para éste problema fue un control PID autosintonizable mediante lógica difusa (PID-ASLD) y un control difuso de postura (PID-ASLD + CDP), respectivamente. La finalidad es que sean capaces de modificar los ángulos iniciales de una postura estática para mantener a la cadera en un ángulo de referencia.

Como se vio en la Sección 2.8, el controlador PID-ASLD se comporta como un controlador PID al que se le proporcionó, a manera sintonizador, un controlador difuso. El control difuso depende del error, del cambio del error y de la suma del error. Las salidas de éste son las ganancias $K_p$, $K_i$ y $K_d$. Este controlador será el núcleo del control de postura en dirección Pitch, como se puede observar en la Figura 4.2. Por otra parte, este controlador determinará en dirección Roll el ángulo más adecuado para que la cadera y los tobillos se adapten en esta dirección, dicho comportamiento se muestra en la Figura 4.3.

El CDP se encargará de ajustar la longitud entre los tobillos y la cadera para su adaptación en una pendiente en dirección Roll. Básicamente el controlador se encargará de alzar más una u otra pierna según la pendiente en dirección Roll. En la Figura 4.3, se demuestra que la distancia inicial de los tobillos no es la misma en una pendiente nula que en una pendiente diferente de cero en dirección Roll, por lo tanto este controlador se encargará de solucionar ese problema.

Las ecuaciones (4.1) y (4.2) muestran la definición del error para cada una de las direcciones de control. Se debe destacar que los ángulos $\theta_{ref\_horizontal}$ valen cero, es decir que el objetivo de control del controlador PID-ASLD es estabilizar el error en dirección Pitch y Roll.

$$error_{pitch} = \theta_{ref_{Picth}} - \theta_{pitch} = -\theta_{Pitch} \tag{4.1}$$

$$error_{roll} = \theta_{ref_{Roll}} - \theta_{roll} = -\theta_{Roll} \tag{4.2}$$

En una pendiente positiva en dirección Pitch, las acciones de control harán que el robot mantenga una postura estable haciendo más pequeña la distancia del CoM a los pies, esto se debe a que en una pendiente positiva entre mas lejos esté el centro de masa de los pies, habrá más posibilidad a la caída. En el caso de las pendientes negativas, el CoM se debe alejar de los pies para mantener el equilibrio y contrarrestar las fuerzas debidas a la gravedad. Esta metodología fue introducida e implementada después de hacer un análisis de postura del robot bípedo sobre su plataforma de pruebas y sobre una prueba humana, dónde se observo que la postura de los humanos tiende a ser de la misma forma; sin embargo, hay una diferencia, los humano inclinan el centro de masa hacía adelante cuando se encuentra en pendientes positivas y hacia atrás par pendiente negativas, así que por lo general, no se busca el ángulo cero de la horizontal.

Las acciones de control para el robot bípedo tendrán la forma de un control PID, sin embargo se adiciona el elemento perteneciente al control difuso de postura, su valor dependerá de si la acción de control reside en la pierna derecha o izquierda . Se debe tener en cuenta que el control PID-ASLD funciona en dirección Pitch y Roll mientras que, el control difuso de postura sólo trabaja en dirección Roll. La diferencia principal entre controladores se contrasta en las acciones de control, ya que el control difuso depende del ángulo Roll, pero toma acción en dirección Pitch, ésto se hace por que la distancia entre

cadera y pies es controlada por los ángulos de cadera, rodilla y tobillo; todos dispuestos en dirección Pitch. En la ecuación (4.3), se presenta de manera general la configuración de las acciones de control (Posteriormente se detallará cada una).

$$u = \pm K_p(error) \pm K_i(error_{suma}) \pm K_d(error_{cambio}) \pm a_S \tag{4.3}$$

### 4.2.1 Dirección pitch

Suponiendo al robot en una plataforma con un ángulo $\phi$ en dirección Pitch, el robot medirá con ayuda de la IMU el error generado por estar parado en esta plataforma y el control determinará la acción de control conveniente para acercarse a $\theta_{ref\_horizontal_{Picth}}$. En la Figura 4.2 se muestran las acciones de control tomadas por el control PID autosintonizable en dirección pitch. Se presentan bajo la siguiente nomenclatura, establecida con la misma numeración utilizada para los eslabones en la Figura 3.1:

- servomotores cadera en dirección Pitch: **u31** y **u32**
- servomotores rodilla en dirección Pitch: **u41** y **u42**
- servomotores tobillo en dirección Pitch: **u51** y **u52**

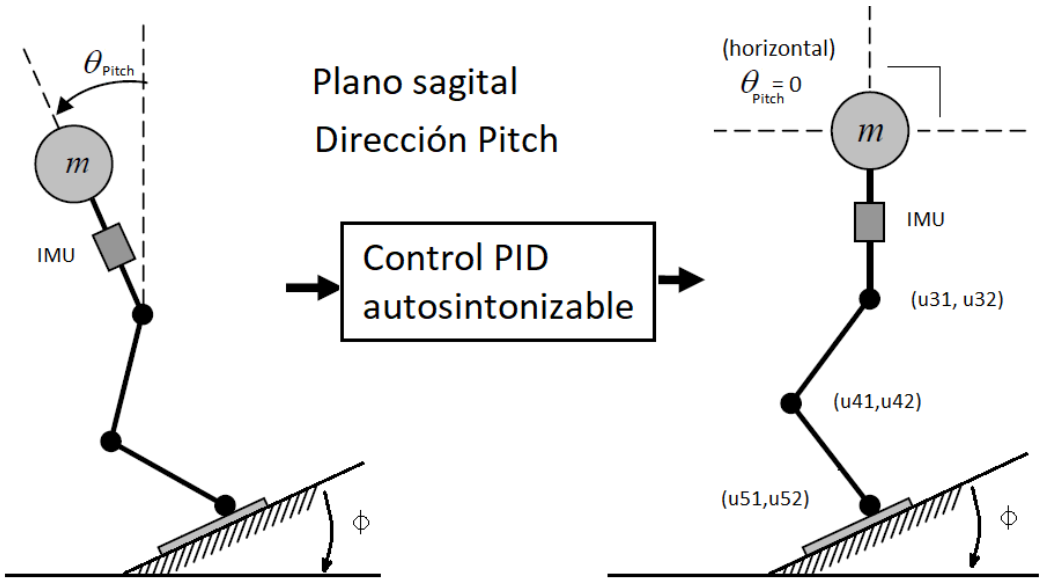


Figura 4.2: Operación del control PID autosintonizable para la postura en dirección picth (Esquema basado en [23])

### 4.2.2 Dirección roll

En la Figura 4.3 se muestra la operación del control de postura en dirección roll. El control en dirección Roll es más complejo, ya que se debe tomar en cuenta la longitud de los tobillo a la cadera de forma que la proyección del CoM en el suelo se mantenga dentro del polígono de soporte, esta acción de control será determinada por el CDP. Para poder lograr, lo antes mencionado, se debe medir el ángulo en Roll y correspondiente a eso se actúan los servomotores de la cadera, rodilla y tobillos en dirección Pitch. Esta distancia entre cadera y tobillos es distinta para cada pierna y dependerá totalmente del ángulo medido en dirección Roll. Por ejemplo, si el ángulo medido fuera positivo, la pierna izquierda tendría que ser mas pequeña que la derecha y viceversa, si la pendiente fuera negativa. Se denota como $a_D$ (pierna derecha) y $a_I$ pierna izquierda. Hay que agregar, que los servomotores de la cadera y los pies con un ángulo de operación en dirección Roll, estarán actuados por el control PID-ASLD.

Las acciones de control se presentan bajo la siguiente nomenclatura, establecida con la misma numeración utilizada para los eslabones en la Figura 3.1:

- servomotores cadera, rodilla, y tobillo en dirección Pitch: $a_D$ y $a_I$
- servomotores cadera en dirección Roll: **u21** y **u22**
- servomotores tobillo en dirección Roll: **u61** y **u62**

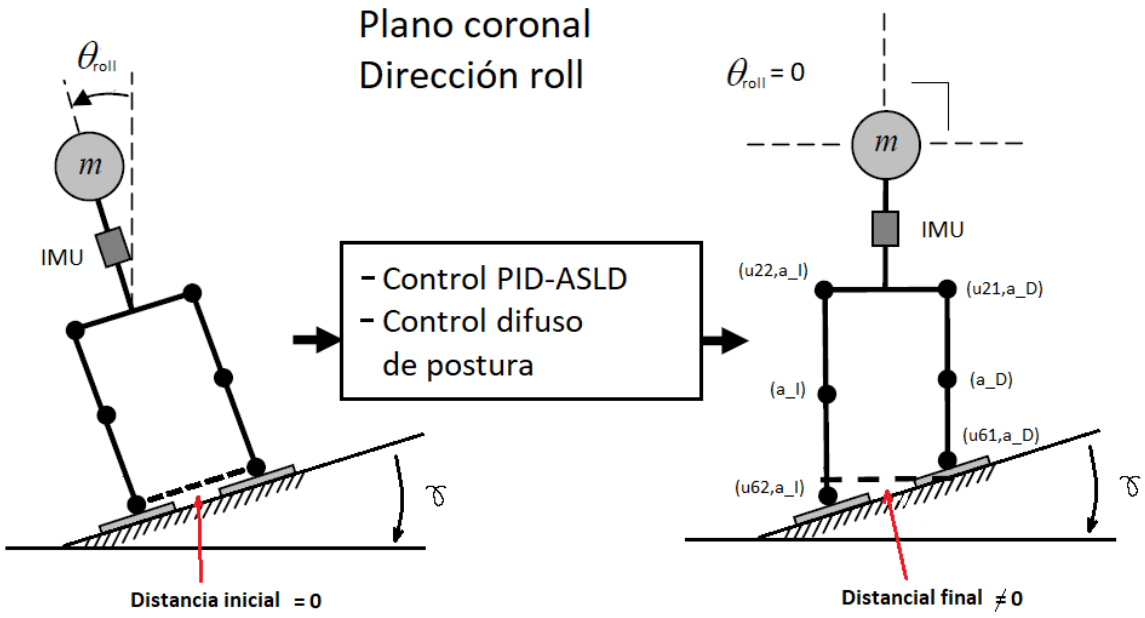


Figura 4.3: Operación del control de postura en dirección Roll[23]

### 4.2.3 Estructura control de postura

En la Figura 4.4 se presenta el diagrama de bloques establecido para el control de postura. Se puede observar el controlador difuso de postura que sólo depende del error en dirección Roll y el controlador difuso que funge como sintonizador de ganancias para el control PID de postura. Se observa también la existencia de ángulos iniciales, estos ángulos iniciales son aquellos dónde el robot se encuentra en posición de "home.º de descanso. La variable $eP$, hace referencia a la parte proporcional del control PID y según la dirección será, $error_{pitch}$ o $error_{roll}$, de igual forma la parte integral $eI$, la cual considera $\sum_0^t error_{pitch}$, y $\sum_0^t error_{roll}$. Por último se tiene la parte proporcional $eD$, con $\Delta error_{pitch}$ y $\Delta error_{roll}$.

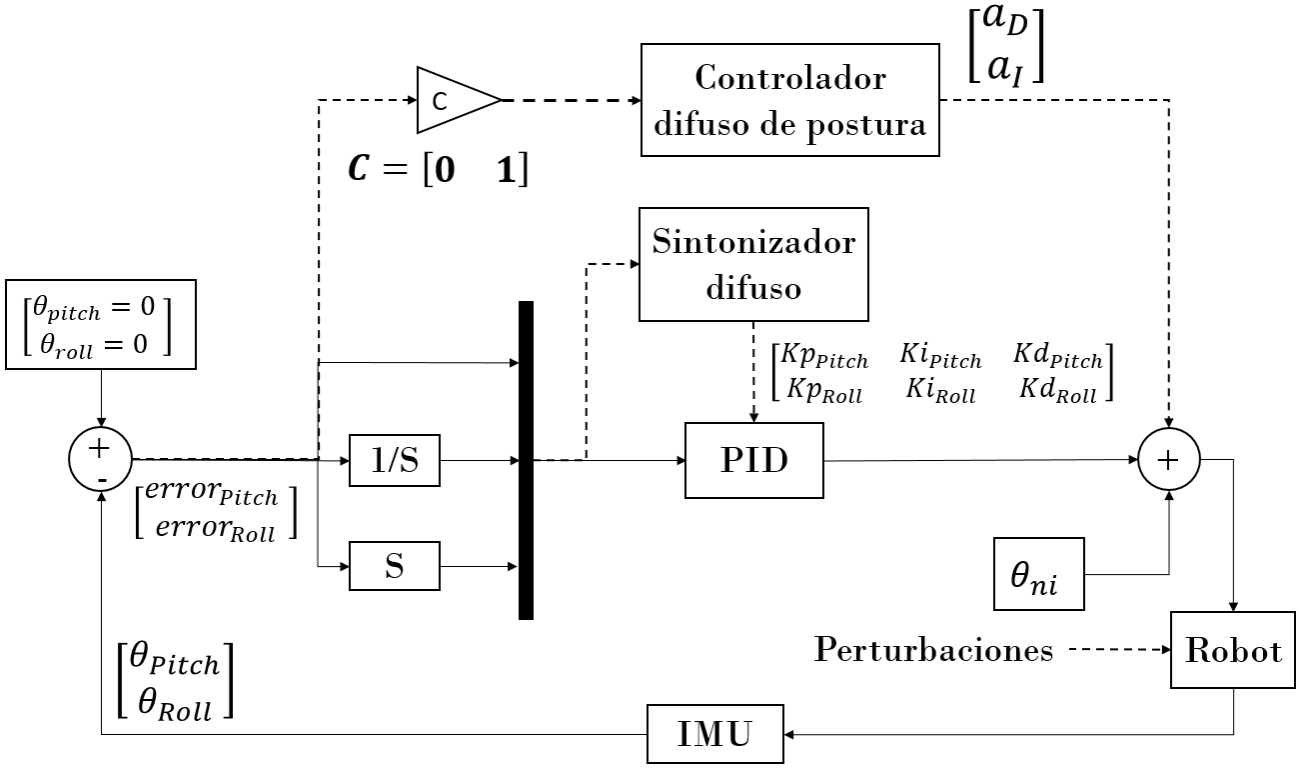


Figura 4.4: Diagramas de bloques del control de postura

Con base en la Figura 3.2, a continuación se muestra la forma y configuración de las acciones de control de manera discreta para los servomotores del robot bípedo. En esta lista de ecuaciones se puede observar la yuxtaposición de controladores. Como se describió previamente, existen dos controladores difusos que serán desarrollados en las Secciones 4.2.4 (CDP) y 4.2.5 (PID-ASLD), respectivamente.

En las ecuaciones (4.4) y (4.5), se muestra la forma general, de las salidas de control generadas por los controladores. En la ecuación 4.4, se aprecia la forma del controlador PID-ASLD en dirección Roll y en la ecuación 4.5, se observa el controlador PID-ASLD y el CDP.

$$\theta'_{ni} = \theta_{ni} \pm PID_{roll} \tag{4.4}$$

$$\theta'_{mi} = \theta_{mi} \pm PID_{pitch} \pm a_S \tag{4.5}$$

donde $i = 1$ para la acción de control de la pierna izquierda, e $i = 2$ para la derecha. $n$ toma los valores (2 y 6), para modificar las articulaciones de la cadera y tobillos respectivamente en dirección Roll. $m$ toma los valores (3, 4 y 5), para compensar las articulaciones de la cadera, rodillas y tobillos, respectivamente, en dirección Pitch. La compensación $a_S$ toma el valor $a_D$ cuando $i = 1$, y $a_I$ cuando $i = 2$.

En la ecuación 4.6, se señala la parte interna de los controladores PID, ya sea en dirección Pitch o Roll.

$$\begin{aligned} PID_{pitch} &= K_{p_{Pitch}}(e_{Pitch}) \pm K_{i_{Pitch}} \textstyle\sum_0^t e_{Pitch}(\tau)\Delta\tau \pm K_{d_{Pitch}}(\Delta e_{Pitch}) \\ PID_{roll} &= K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \textstyle\sum_0^t e_{roll}(\tau)\Delta\tau \pm K_{d_{Roll}}(\Delta e_{Roll}) \end{aligned} \tag{4.6}$$

Se presenta en seguida el desarrollo detallado de cada una de las acciones de control, aplicadas en el ángulo correspondiente de operación, los ángulo coinciden con el número de cada servomotor, además se describe a que parte de la arquitectura del robot pertenecen.

- Cadera $\rightarrow \theta'_{21} = \theta_{21} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Cadera $\rightarrow \theta'_{22} = \theta_{22} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Cadera $\rightarrow \theta'_{31} = \theta_{31} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch}) - a_D$
- Cadera $\rightarrow \theta'_{32} = \theta_{32} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch}) + a_I$
- Rodilla $\rightarrow \theta'_{41} = \theta_{41} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch}) - a_D$
- Rodilla $\rightarrow \theta'_{42} = \theta_{42} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch}) + a_I$
- Tobillo $\rightarrow \theta'_{51} = \theta_{51} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch}) - a_D$
- Tobillo $\rightarrow \theta'_{52} = \theta_{52} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch}) + a_I$
- Tobillo $\rightarrow \theta'_{61} = \theta_{61} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Tobillo $\rightarrow \theta'_{62} = \theta_{62} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$

### 4.2.4 Diseño del controlador difuso de postura

En esta sección se desarrollará a detalle el diseño del controlador empleado para la postura del robot bípedo en dirección Roll. Especificamente el controlador dispuesto para establecer la distancia entre la cadera y los tobillos. Este controlador tendrá como entrada el error en dirección Roll y tendrá como salidas $a_D$ y $a_I$, cada una de éstas salidas actuará en servomotores en dirección Pitch como se ve en la ecuación (4.5).

En la Figura 4.5, se establece la configuración previamente mencionada del controlador difuso, así como las secciones más importante de un controlador tipo Mamdani, con el cual será hecho el controlador. Cada una de las partes de éste controlador serán desarrolladas a continuación.

En el presente trabajo se utilizaron funciones de membresía **triangulares** y **trapezoidales**, esto se debe a que en términos de procesamiento, las reglas de correspondencia se resuelven en menor tiempo.

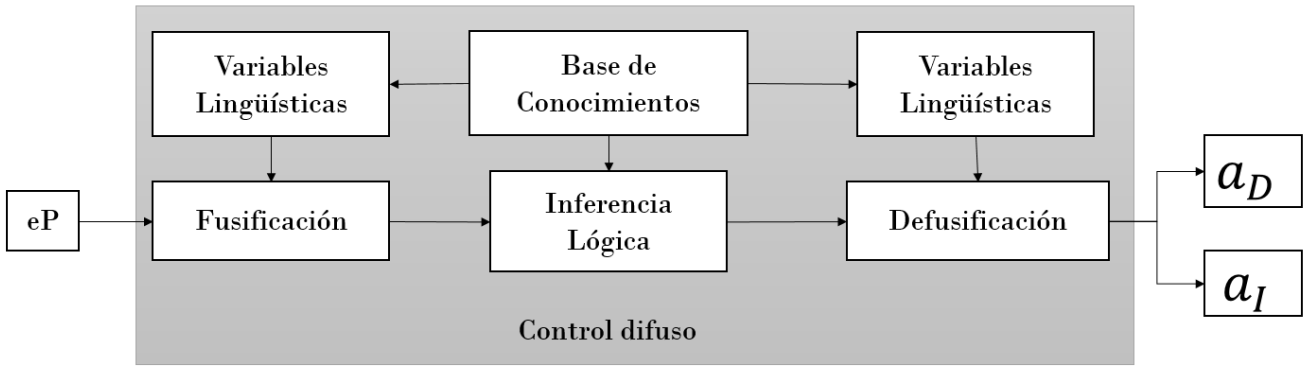


Figura 4.5: Operación del control difuso de postura en dirección Roll

La nomenclatura utilizada en las siguiente secciones, se presenta enseguida:

Error Negativo Grande → **NG**
Error Negativo Pequeño → **NP**
Error CEro → **CE**
Error Positivo Pequeño → **PP**
Error Positivo Grande → **PG**

Ángulo Bajo → **AB**
Ángulo Medio Bajo → **AMB**
Ángulo Medio → **AM**
Ángulo Medio Alto → **AMA**
Ángulo Alto→ **AA**

#### 4.2.4.1. Base de conocimientos

La base de conocimientos para este controlador se adquirió empíricamente usando la plataforma del robot bípedo y haciendo pruebas sobre el comportamiento deseado que debería tener el robot en el caso

de inclinarse en sentido Roll. El concepto de hacer una concordancia entre la distancia de la cadera con los tobillos fue tomado del articulo [23], en dónde generan este comportamiento teniendo la medida de las piernas en todo momento. En la Tabla 4.2, se muestra la correspondencia para el control CDP

Tabla 4.2: Tabla de verdad para el controlador difuso de postura

| **Error** | Correspondencia | **a_D** | **a_I** |
|---|---|---|---|
| NG | $\Longrightarrow$ | AA | AA |
| NP | $\Longrightarrow$ | AM | AM |
| CE | $\Longrightarrow$ | AB | AB |
| PP | $\Longrightarrow$ | AM | AM |
| PG | $\Longrightarrow$ | AA | AA |

#### 4.2.4.2. Variables lingüísticas

Para este sistema difuso existe una entrada y dos salidas, por lo tanto existirán tres variables lingüísticas, cada variable lingüística, como se trata en la Sección 2.5.1, consta de (v,X,$T_v$,M). Asumiendo la misma nomenclatura se tiene que:

**Entrada**

- eP: $error_{roll} \rightarrow x_e \in [-90, 90] \rightarrow T(x_e) = \{NG, NP, CE, PP, PG\}$

**M(NG) = trimf(**$error_{roll}$**,[-90,-90,-15])**

**M(NP) = trimf(**$error_{roll}$**,[-60,-15,0])**

**M(CE) = trimf(**$error_{roll}$**,[-15,-5,5,15])**

**M(PP) = trimf(**$error_{roll}$**,[0,15,60])**

**M(PG) = trimf(**$error_{roll}$**,[15,90,90])**

**Salidas**

- a_D $\rightarrow y_{aD} \in [0, 50] \rightarrow T(x_e) = \{AB, AMB, AM, AMA, AA\}$
- a_I $\rightarrow y_{aI} \in [0, 50] \rightarrow T(x_e) = \{AB, AMB, AM, AMA, AA\}$

**M(AB) = trimf(**$a_D$**,[0,0,5,10])** **M(AB) = trimf(**$a_I$**,[0,0,5,10])**

**M(AMB) = trimf($a_D$,[5, 10,15])** **M(AMB) = trimf($a_I$,[5, 10,15])**

**M(AM) = trimf($a_D$,[5,15,25])** **M(AM) = trimf($a_I$,[5,15,25])**

**M(AMA) = trimf($a_D$,[15,30,35])** **M(AMA) = trimf($a_I$,[15,30,35])**

**M(AA) = trimf($a_D$,[30,35,50,50])** **M(AA) = trimf($a_I$,[30,35,50,50])**

En la Figura 4.6, se muestra del lado izquierdo el arreglo visual de las funciones de membresía utilizadas en el controlador, del lado derecho se muestra la configuración de las funciones de membresía de las salidas $a_D$ y $a_I$.

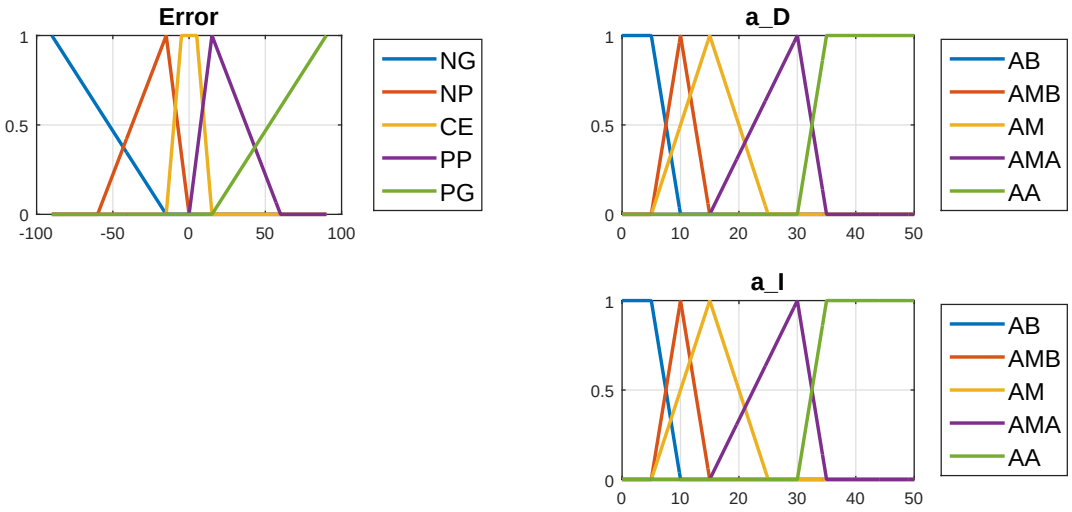


Figura 4.6: Operación del control difuso de postura en dirección roll

#### 4.2.4.3. Fusificación

Como se establece en la Sección 2.5.3, la fusificación de un dato de entrada $x_0$ es necesario transformarlo a un conjunto difuso del tipo singleton, esto tendrá como resultado la intersección de ese dato con los conjuntos previamente definidos para el universo de discurso del error en dirección Roll, los ángulos $a_D$ y $a_I$. En la ecuación (4.7), $x_{e0}$ representa al dato de entrada al controlador difuso y el subíndice **i**, está determinado por el número de conjuntos difusos en el universo de discurso.

$$\bigvee_{x_e}\left[\mu_{eP_i}(x_e) \wedge \mu_{singleton_{entrada_{eP}}}(x_e)\right] = \mu_{eP_i}(x_{e0}) \tag{4.7}$$

#### 4.2.4.4. Inferencia lógica

Para el controlador difuso de postura, se detallará continuación las operaciones inherentes a la inferencia lógica o toma de decisiones.

**Ángulo de la rodilla derecha:**

Para el universo de discurso $x_{errorRoll}$

$$\mu_{aD'_1}(y_{a\_D}) = \mu_{eP_1}(x_{e0}) \wedge \mu_{aD_1}(y_{a\_D}) \tag{4.8}$$

$$\mu_{aD'_2}(y_{a\_D}) = \mu_{eP_2}(x_{e0}) \wedge \mu_{aD_2}(y_{a\_D}) \tag{4.9}$$

$$\mu_{aD'_3}(y_{a\_D}) = \mu_{eP_3}(x_{e0}) \wedge \mu_{aD_3}(y_{a\_D}) \tag{4.10}$$

$$\mu_{aD'_4}(y_{a\_D}) = \mu_{eP_4}(x_{e0}) \wedge \mu_{aD_4}(y_{a\_D}) \tag{4.11}$$

$$\mu_{aD'_5}(y_{a\_D}) = \mu_{eP_5}(x_{e0}) \wedge \mu_{aD_5}(y_{a\_D}) \tag{4.12}$$

---

Función de membresía agregada para a_D:

$$\mu_{aD'}(y_{a\_D}) = \mu_{aD'_1}(y_{a\_D}) \vee \mu_{aD'_2}(y_{a\_D}) \vee \mu_{aD'_3}(y_{a\_D}) \vee \mu_{aD'_4}(y_{a\_D}) \vee \mu_{aD'_5}(y_{a\_D}) \tag{4.13}$$

**Ángulo de la rodilla izquierda:**

Para el universo de discurso $x_{errorRoll}$

$$\mu_{aI'_1}(y_{a\_I}) = \mu_{eP_1}(x_{e0}) \wedge \mu_{aI_1}(y_{a\_I}) \tag{4.14}$$

$$\mu_{aI'_2}(y_{a\_I}) = \mu_{eP_2}(x_{e0}) \wedge \mu_{aI_2}(y_{a\_I}) \tag{4.15}$$

$$\mu_{aI'_3}(y_{a\_I}) = \mu_{eP_3}(x_{e0}) \wedge \mu_{aI_3}(y_{a\_I}) \tag{4.16}$$

$$\mu_{aI'_4}(y_{a\_I}) = \mu_{eP_4}(x_{e0}) \wedge \mu_{aI_4}(y_{a\_I}) \tag{4.17}$$

$$\mu_{aI'_5}(y_{a\_I}) = \mu_{eP_5}(x_{e0}) \wedge \mu_{aI_5}(y_{a\_I}) \tag{4.18}$$

Función de membresía agregada para a_I:

$$\mu_{aI'}(y_{a\_I}) = \mu_{aI'_1}(y_{a\_I}) \vee \mu_{aI'_2}(y_{a\_I}) \vee \mu_{aI'_3}(y_{a\_I}) \vee \mu_{aI'_4}(y_{a\_I}) \vee \mu_{aI'_5}(y_{a\_I}) \tag{4.19}$$

#### 4.2.4.5. Defusificación

Utilizando el método del centroide para la defusificación, se tienen las ecuaciones (4.20) y (4.20) para el calculo de $a_D$ y $a_I$, respectivamente.

$$a\_D = \frac{\sum_0^{50}(y_{a\_D})\,[\mu_{aD'}(y_{a\_D})]}{\sum_0^{50}(\mu_{aD'}(y_{a\_D}))} \tag{4.20}$$

$$a\_I = \frac{\sum_0^{50}(y_{a\_I})\,[\mu_{aI'}(y_{a\_I})]}{\sum_0^{50}(\mu_{aI'}(y_{a\_I}))} \tag{4.21}$$

### 4.2.5 Diseño del controlador PID autosintonizable

Una vez establecido el esquema de control PID para cada junta, se desarrollará el controlador difuso destinado a cada una de las ganancias de dicho controlador, así como el control difuso para el control de postura en dirección Roll.

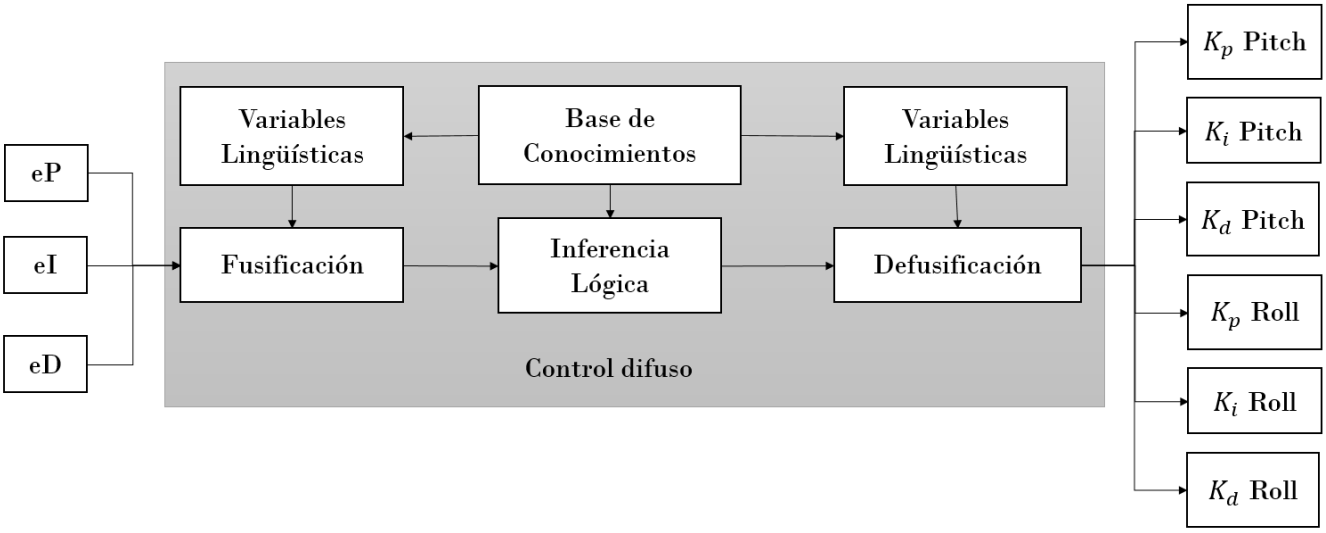


Figura 4.7: Operación del sintonizador para el control PID-ASLD

El universo de discurso para cada una de las ganancias se obtuvo experimentalmente, al hacer pruebas manuales con el robot bípedo Scout, sobre su plataforma. El universo de discurso para el error, el cambio

del error y la suma del error, fue propuesto después de realizar varias pruebas. Las pruebas fueron hechas sobre la plataforma empujando al robot, cuando éste mantenía una pose "estable", la idea principal de la prueba es medir la sensibilidad y magnitud de cada tipo de error. Se determino que con 5 funciones de membresía o conjuntos difusos se daba solución al problema. Dado que las ganancias serán siempre positivas el signo del error dará la dirección de control.

Se ocupará la siguiente nomenclatura:

| | |
|---|---|
| Error Negativo Grande → **NG** | Ganancia Baja → **GB** |
| Error Negativo Pequeño → **NP** | Ganancia Media Baja → **GMB** |
| Error CEro → **CE** | Ganancia Mediana → **GM** |
| Error Positivo Pequeño → **PP** | Ganancia Media Alta → **GMA** |
| Error Positivo Grande → **PG** | Ganancia Alta→ **GA** |

#### 4.2.5.1. Base de conocimientos

La relación de reglas que dan lugar a la sintonización de la ganancia $K_p$ por medio del controlador difuso se muestran en la Tabla a) de la Figura 4.8. En ésta tabla se muestra un conjuntos de reglas orientadas para un comportamiento asintóticamente estable. Cerca de un punto de equilibrio ficticio (este punto es donde todos los errores son iguales a cero), se tiene una ganancia $K_p$ baja (**GB**), pero si en algún momento se aleja de este supuesto punto de equilibrio la ganancia crecerá de tal forma que tendería a regresar a dicho punto.

La Tabla b) de la Figura 4.8 contiene las reglas para la ganancia $K_i$, el objetivo de éstas es evitar que la acción de control se incremente desfavorablemente cuando la suma del error ($\sum_0^t e_{pitch}$) sea grande. Se consideró también que la suma del error afecta la dinámica del controlador, haciendo mayor el tiempo de asentamiento; por lo que, se decidió hacer dependiente a la ganancia $K_i$ del error, teniendo así dos entradas que pueden alterar su valor, de esta forma, cuando está en el punto de equilibrio ficticio $error = 0$ la ganancia es alta.

En la tabla c) de la Figura 4.8 se presenta la relación de reglas del controlador difuso para la ganancia Kd. Se debe destacar que al igual que la ganancia $K_p$, la ganancia $K_d$, depende del cambio del error y del error mismo. Se determinó que las reglas fueran colocadas con un comportamiento deseado de Kd asintóticamente estable. Este comportamiento se definió con el siguiente razonamiento, conforme el sistema se aleje del punto de equilibrio ficticio, más trabajo le costará volver, y por ende lo hará en un mayor tiempo. Con el objetivo de reducir el tiempo de asentamiento, pero manteniendo la ganancia Kd lo bastante pequeña para evitar inestabilidades por impulsos repentinos, se utilizó la misma configuración que en la ganancia Kp, sin embargo en lugar del error como entrada preponderante, se eligió el cambio del error ($e_{rate}$), dado que ésta ganancia influirá en la dinámica del sistema.

| Ganancia Kp | | | | | |
|---|---|---|---|---|---|
| $\dot{e}$ / $e$ | **NG** | **NP** | **CE** | **PP** | **PG** |
| **NG** | GA | GMA | GM | GMA | GA |
| **NP** | GMA | GM | GMB | GM | GMA |
| **CE** | GM | GMB | GB | GMB | GM |
| **PP** | GMA | GM | GMB | GM | GMA |
| **PG** | GA | GMA | GM | GMA | GA |

a)

| Ganancia Ki | | | | | |
|---|---|---|---|---|---|
| $\sum_0^t e$ / $e$ | **NG** | **NP** | **CE** | **PP** | **PG** |
| **NG** | GB | GMB | GM | GMB | GB |
| **NP** | GMB | GMB | GMA | GMB | GMB |
| **CE** | GM | GMA | GA | GMA | GM |
| **PP** | GMB | GMB | GMA | GMB | GMB |
| **PG** | GB | GMB | GM | GMB | GB |

b)

| Ganancia Kd | | | | | |
|---|---|---|---|---|---|
| $e$ / $e_{rate}$ | **NG** | **NP** | **CE** | **PP** | **PG** |
| **NG** | GA | GMA | GM | GMA | GA |
| **NP** | GA | GMA | GMB | GMA | GA |
| **CE** | GA | GMA | GB | GMA | GA |
| **PP** | GA | GMA | GMB | GMA | GA |
| **PG** | GA | GMA | GM | GMA | GA |

c)

Figura 4.8: Tablas de reglas de control para las ganancias $K_p$, $K_i$ y $K_d$

#### 4.2.5.2. Variables lingüísticas

Para el sistema difuso existen 3 entradas y 8 salidas, por lo tanto existiran 11 variables linguisticas, cada variables lingüística con sus respectivas propiedades: (v,X,$T_v$,M).

**Entradas**

- eP $\rightarrow x_e \in [-90, 90] \rightarrow T(x_e) = \{NG, NP, CE, PP, PG\}$
- eI $\rightarrow x_{se} \in [-200, 200] \rightarrow T(x_{se}) = \{NG, NP, CE, PP, PG\}$
- eD $\rightarrow x_{ce} \in [-300, 300] \rightarrow T(x_{ce}) = \{NG, NP, CE, PP, PG\}$

**M(NG) = trimf(eP,[-90,-90,-15])**
**M(NG) = trimf(eP,[-60,-15,0])**
**M(NG) = trapmf(eP,[-15,-5,5,15])**
**M(NG) = trimf(eP,[0,15,60])**
**M(NG) = trimf(eP,[15,90,90])**

**M(NG) = trimf(eI,[-200,-200,-15])**
**M(NG) = trimf(eI,[-100,-50,0])**
**M(NG) = trimf(eI,[-50,0,50])**
**M(NG) = trimf(eI,[0,50,100])**
**M(NG) = trimf(eI,[15,200,200])**

**M(NG) = trimf(eD,[-300,-300,-50])**
**M(NP) = trimf(eD,[-100,-50,0])**
**M(CE) = trapmf(eD,[-50,-5,5,50])**
**M(PP) = trimf(eD,[0,50,100])**
**M(PG) = trimf(eD,[50,300,300])**


Figura 4.9: Operación del control difuso de postura en dirección roll

**Salidas**

- KP_pitch $\rightarrow y_{kp_{pitch}} \in [0, 1.2] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$
- KI_pitch $\rightarrow y_{ki_{pitch}} \in [0, 0.9] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$
- KD_pitch $\rightarrow y_{kd_{pitch}} \in [0, 0.3] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$
- KP_roll $\rightarrow y_{kp_{roll}} \in [0, 1.2] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$
- KI_roll $\rightarrow y_{ki_{roll}} \in [0, 0.9] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$
- KD_roll $\rightarrow y_{kd_{roll}} \in [0, 0.3] \rightarrow T(x_e) = \{GB, GMB, GM, GMA, GA\}$

**M(GB) = trimf($KP_{pitch}$,[0, 0,0.3,0.5])**
**M(GMB) = trimf($KP_{pitch}$,[0.3, 0.5, 0.7])**
**M(GM) = trimf($KP_{pitch}$,[0.5,0.7,0.9])**
**M(GMA) = trimf($KP_{pitch}$,[0.7, 0.9, 1.1])**
**M(GA) = trapmf($KP_{pitch}$,[0.9, 1.0, 1.2,1.2])**

**M(GB) = trapmf($KI_{pitch}$,[0, 0,0.2,0.4])**
**M(GMB) = trimf($KI_{pitch}$,[0.2, 0.4, 0.5])**
**M(GM) = trimf($KI_{pitch}$,[0.4,0.5,0.6])**
**M(GMA) = trimf($KI_{pitch}$,[0.5,0.6,0.7])**
**M(GA) = trapmf($KI_{pitch}$,[0.6,0.7,0.9,0.9])**

**M(GB) = trimf($KD_{pitch}$,[0,0,0.05])**
**M(GMB) = trimf($KD_{pitch}$,[0, 0.05, 0.1])**
**M(GM) = trimf($KD_{pitch}$,[0.05,0.1,0.15])**
**M(GMA) = trimf($KD_{pitch}$,[0.1, 0.15, 0.2])**
**M(GA) = trimf($KD_{pitch}$,[0.15, 0.2, 0.3,0.3])**

**M(GB) = trimf($KP_{roll}$,[0,0,0.1,0.3])**
**M(GMB) = trimf($KP_{roll}$,[0.1, 0.3, 0.5])**
**M(GM) = trimf($KP_{roll}$,[0.3,0.5,0.7])**
**M(GMA) = trimf($KP_{roll}$,[0.5, 0.7, 0.9])**
**M(GA) = trapmf($KP_{roll}$,[0.7, 0.9, 0.9,0.9])**

**M(GB) = trapmf($KI_{roll}$,[0, 0,0.1,0.3])**
**M(GMB) = trimf($KI_{roll}$,[0.1, 0.3, 0.5])**
**M(GM) = trimf($KI_{roll}$,[0.3,0.5,0.7])**
**M(GMA) = trimf($KI_{roll}$,[0.5, 0.7, 0.9])**
**M(GA) = trapmf($KI_{roll}$,[0.7, 0.9, 0.9,0.9])**

**M(GB) = trimf($KD_{roll}$,[ 0,0,0.1])**
**M(GMB) = trimf($KD_{roll}$,[0, 0.1, 0.2])**
**M(GM) = trimf($KD_{roll}$,[0.1,0.2,0.3])**
**M(GMA) = trimf($KD_{roll}$,[0.2, 0.3, 0.4])**
**M(GA) = trimf($KD_{roll}$,[0.3, 0.4, 0.5,0.5])**

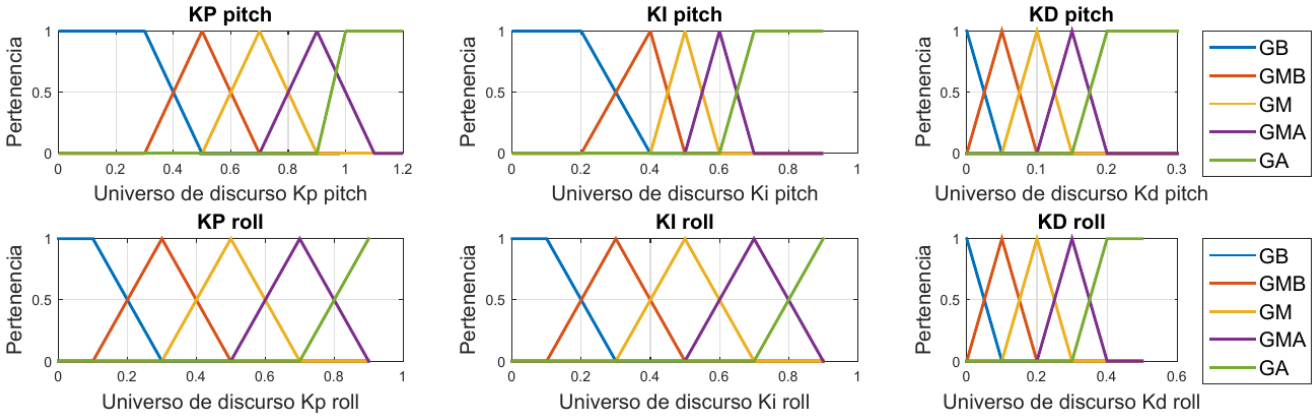


Figura 4.10: Operación del control difuso de postura en dirección roll

#### 4.2.5.3. Fusificación

Para la defusificación de las entradas se utiliza la siguiente expresión:

$$\bigvee_{x_e}\left[\mu_{eP_i}(x_e) \wedge \mu_{singleton_{entrada_{eP}}}(x_e)\right] = \mu_{eP_i}(x_{e0}) \tag{4.22}$$

$$\bigvee_{x_{se}}\left[\mu_{eI_i}(x_{se}) \wedge \mu_{singleton_{entrada_{eI}}}(x_{se})\right] = \mu_{eI_i}(x_{se0}) \tag{4.23}$$

$$\bigvee_{x_{ce}}\left[\mu_{eD_i}(x_{ce}) \wedge \mu_{singleton_{entrada_{eD}}}(x_{ce})\right] = \mu_{eD_i}(x_{ce0}) \tag{4.24}$$

#### 4.2.5.4. Inferencia lógica

**Ganancia Kp pitch:**

Para los universos de discurso $x_e$ , $x_{ce}$

$$\mu_{KP'_{pitch_i}}(y_{kp_{pitch}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eD_i}(x_{ce0})] \wedge \mu_{KP_{pitch_i}}(y_{kp_{pitch}}) \tag{4.25}$$

---

Función agregada:

$$\mu_{KP'_{pitch}}(y_{kp_{pitch}}) = \mu_{KP'_{pitch_1}}(y_{kp_{pitch}}) \vee \mu_{KP'_{pitch_2}}(y_{kp_{pitch}}) \vee \mu_{KP'_{pitch_3}}(y_{kp_{pitch}}) \vee \mu_{KP'_{pitch_4}}(y_{kp_{pitch}})$$
$$\vee \mu_{KP'_{pitch_5}}(y_{kp_{pitch}}) \tag{4.26}$$

**Ganancia Ki pitch:**

Para $x_{se}$ , $x_e$

$$\mu_{KI'_{pitch_i}}(y_{ki_{pitch}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eI_i}(x_{se0})] \wedge \mu_{KI_{pitch_i}}(y_{ki_{pitch}}) \tag{4.27}$$

---

Función de membresía agregada:

$$\mu_{KI'_{pitch}}(y_{ki_{pitch}}) = \mu_{KI'_{pitch_1}}(y_{ki_{pitch}}) \vee \mu_{KI'_{pitch_2}}(y_{ki_{pitch}}) \vee \mu_{KI'_{pitch_3}}(y_{ki_{pitch}}) \vee \mu_{KI'_{pitch_4}}(y_{ki_{pitch}})$$
$$\vee \mu_{KI'_{pitch_5}}(y_{ki_{pitch}}) \tag{4.28}$$

**Ganancia Kd pitch:**

Para $x_{ce}$ , $x_e$

$$\mu_{KD'_{pitch_i}}(y_{kd_{pitch}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eD_i}(x_{ce0})] \wedge \mu_{KD_{pitch_i}}(y_{kd_{pitch}}) \tag{4.29}$$

---

Función agregada:

$$\mu_{KD'_{pitch}}(y_{kd_{pitch}}) = \mu_{KD'_{pitch_1}}(y_{kd_{pitch}}) \vee \mu_{KD'_{pitch_2}}(y_{kd_{pitch}}) \vee \mu_{KD'_{pitch_3}}(y_{kd_{pitch}}) \vee \mu_{KD'_{pitch_4}}(y_{kd_{pitch}})$$
$$\vee \mu_{KD'_{pitch_5}}(y_{kd_{pitch}}) \tag{4.30}$$

**Ganancia Kp roll:**

Para $x_e$ , $x_{ce}$ y $y_{kp_{roll}}$

$$\mu_{KP'_{roll_i}}(y_{kp_{roll}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eD_i}(x_{ce0})] \wedge \mu_{KP_{roll_i}}(y_{kp_{roll}}) \tag{4.31}$$

---

Función agregada:

$$\mu_{KP'_{roll}}(y_{kp_{roll}}) = \mu_{KP'_{roll_1}}(y_{kp_{roll}}) \vee \mu_{KP'_{roll_2}}(y_{kp_{roll}}) \vee \mu_{KP'_{roll_3}}(y_{kp_{roll}}) \vee \mu_{KP'_{roll_4}}(y_{kp_{roll}})$$
$$\vee \mu_{KP'_{roll_5}}(y_{kp_{roll}}) \tag{4.32}$$

**Ganancia Ki roll:**

Para $x_{se}$ , $x_e$ y $y_{kp_{pitch}}$

$$\mu_{KI'_{roll_i}}(y_{ki_{roll}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eI_i}(x_{se0})] \wedge \mu_{KI_{roll_i}}(y_{ki_{roll}}) \tag{4.33}$$

---

Función agregada:

$$\mu_{KI'_{roll}}(y_{ki_{roll}}) = \mu_{KI'_{roll_1}}(y_{ki_{roll}}) \vee \mu_{KI'_{roll_2}}(y_{ki_{roll}}) \vee \mu_{KI'_{roll_3}}(y_{ki_{roll}}) \vee \mu_{KI'_{roll_4}}(y_{ki_{roll}}) \vee \mu_{KI'_{roll_5}}(y_{ki_{roll}}) \tag{4.34}$$

**Ganancia Kd roll:**

Para $x_{se}$ , $x_e$ y $y_{kp_{pitch}}$

$$\mu_{KD'_{roll_i}}(y_{kd_{roll}}) = [\mu_{eP_i}(x_{e0}) \vee \mu_{eD_i}(x_{ce0})] \wedge \mu_{KD_{roll_i}}(y_{kd_{roll}}) \tag{4.35}$$

---

Función agregada:

$$\mu_{KD'_{roll}}(y_{kd_{roll}}) = \mu_{KD'_{roll_1}}(y_{kd_{roll}}) \vee \mu_{KD'_{roll_2}}(y_{kd_{roll}}) \vee \mu_{KD'_{roll_3}}(y_{kd_{roll}}) \vee \mu_{KD'_{roll_4}}(y_{kd_{roll}}) \vee \mu_{KD'_{roll_5}}(y_{kd_{roll}}) \tag{4.36}$$

#### 4.2.5.5. Defusificación

Utilizando el método del centroide para la defusificación se obtienen las siguientes ecuaciones:

$$Kp_{pitch} = \frac{\sum_0^{1.2}(y_{kp_{pitch}})(\mu_{KP'_{pitch}}(y_{kp_{pitch}}))}{\sum_0^{1.2}(\mu_{KP'_{pitch}}(y_{kp_{pitch}}))} \tag{4.37}$$

$$Ki_{pitch} = \frac{\sum_0^{0.9}(y_{ki_{pitch}})(\mu_{KI'_{pitch}}(y_{ki_{pitch}}))}{\sum_0^{0.9}(\mu_{KI'_{pitch}}(y_{ki_{pitch}}))} \tag{4.38}$$

$$Kd_{pitch} = \frac{\sum_0^{0.5}(y_{kp_{pitch}})(\mu_{KD'_{pitch}}(y_{kd_{pitch}}))}{\sum_0^{0.5}(\mu_{KD'_{pitch}}(y_{kd_{pitch}}))} \tag{4.39}$$

$$Kp_{roll} = \frac{\sum_0^{1.2}(y_{kp_{roll}})(\mu_{KP'_{roll}}(y_{kp_{roll}}))}{\sum_0^{1.2}(\mu_{KP'_{roll}}(y_{kp_{roll}}))} \tag{4.40}$$

$$Ki_{roll} = \frac{\sum_0^{0.9}(y_{ki_{roll}})(\mu_{KI'_{roll}}(y_{kp_{roll}}))}{\sum_0^{0.9}(\mu_{KI'_{roll}}(y_{ki_{roll}}))} \tag{4.41}$$

$$Kd_{roll} = \frac{\sum_0^{0.5}(y_{kd_{roll}})(\mu_{KD'_{roll}}(y_{kp_{roll}}))}{\sum_0^{0.5}(\mu_{KD'_{roll}}(y_{kd_{roll}}))} \tag{4.42}$$

#### 4.2.5.6. Superficies de control

Las superficies de control son la representación gráfica de las salidas de control del controlador difuso en cuestión. Las salidas de control pueden depender de una o más variables de entrada, en el caso de las superficies de control del presente trabajo se tiene que la ganancia Kp depende del error, y del cambio del error (Pitch, Roll), la ganancia Ki depende del error, y la suma del error (Pitch, Roll), por último la ganancia Kd también depende del error, y del cambio del error (Pitch, Roll). En la ecuación (4.43), se muestran dichas dependencias y en la Figura 4.11 se muestran las superficies de control de las ganancias $K_p$, $K_i$ y $K_d$, en dirección Pitch. Ésta dirección es la más importante para el control de marcha y postura dado el cambio de pendiente en la plataforma de prueba utilizada para el robot bípedo.

$$
\begin{aligned}
Kp_{Pitch} &= f(error_{Pitch}, \Delta error_{Pitch}) \\
Ki_{Pitch} &= f(error_{Pitch}, \textstyle\sum_0^t error_{Pitch}) \\
Kd_{Pitch} &= f(error_{Pitch}, \Delta error_{Pitch})
\end{aligned}
\tag{4.43}
$$

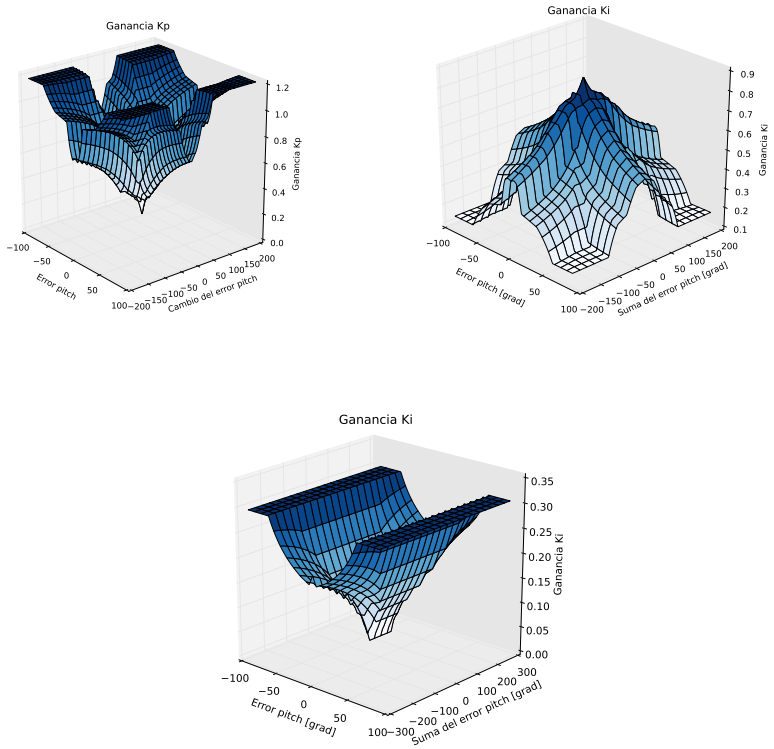


Figura 4.11: Gráfica que muestra el comportamiento de las ganancia $K_p$, $K_i$ y $K_d$ en el control de postura en dirección pitch

## 4.3 Control de Marcha

### 4.3.1 Estructura control PID autosintonizable para la marcha

En la Figura 4.12 se presenta el diagrama de bloques para el control de marcha. Se observa que el controlador difuso fue descartado y la caminata será totalmente dominada por el control PID-ASLD. En la Figura 4.12, al igual que en el control de postura, se utiliza $\theta_{ref}$, equivalente a $\theta_{refPitch} = 0$ y $\theta_{refRoll} = 0$, según la dirección en la que los servomotores del robot bípedo actúen. La variable $eP$, hace referencia al error, según la dirección será, $error_{pitch}$ o $error_{roll}$, de igual forma la la suma del error $eI$, la cual considera $\sum_0^t error_{pitch}$ y $\sum_0^t error_{roll}$. Por último se tiene el cambio del error $eD$, con $\Delta error_{pitch}$ y $\Delta error_{roll}$. En la Figura 4.12, se muestra la adición de *ángulos trayectoria*, estos ángulos fueron calculados teóricamente y fuera de linea en [29].

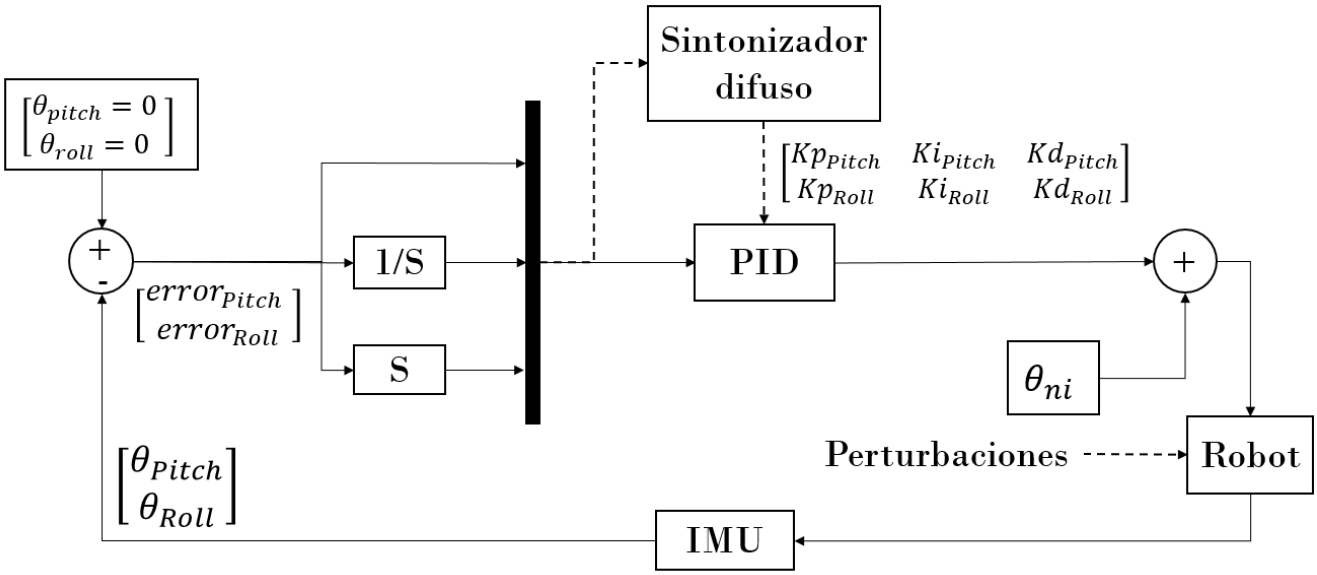


Figura 4.12: Diagramas de bloques del control de marcha

Con base en el control de postura se configuró el control de marcha, con el objetivo de modificar trayectorias previamente calculadas para la caminata del robot bípedo; esto se logro al compensar el ángulo de la trayectoria por medio de una corrección hecha por el controlador PID-ASLD. En las ecuaciones (4.44) y (4.45), se muestra la forma general de las salidas de control generadas por el control de marcha. Se destaca el uso exclusivo del control PID-ASLD.

$$\theta'_{modificados_{ni}} = \theta_{trayectoria_{ni}} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll}) \tag{4.44}$$

$$\theta'_{modificados_{mi}} = \theta_{trayectoria_{mi}} \pm K_{p_{Pitch}}(e_{Pitch}) \pm K_{i_{Pitch}} \sum_0^t e_{Pitch}(\tau)\Delta\tau \pm K_{d_{Pitch}}(\Delta e_{Pitch}) \tag{4.45}$$

donde $i = 1$ para la acción de control de la pierna izquierda e $i = 2$ para la derecha. $n$ toma los valores (2 y 6), para modificar las articulaciones de la cadera y tobillos, respectivamente, en dirección Roll. $m$ toma los valores (3, 4 y 5), para compensar las articulaciones de la cadera, rodillas y tobillos, respectivamente, en dirección pitch.

Se presenta en seguida el desarrollo detallado de cada una de las acciones de control aplicadas en el ángulo correspondiente de operación, los ángulos coinciden con el número de cada servomotor y además se describe a que parte de la arquitectura del robot pertenece cada ángulo.

- Cadera $\rightarrow \theta'_{mod_{21}} = \theta_{tray_{21}} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Cadera $\rightarrow \theta'_{mod_{22}} = \theta_{tray_{22}} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Cadera $\rightarrow \theta'_{mod_{31}} = \theta_{tray_{31}} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch})$
- Cadera $\rightarrow \theta'_{mod_{32}} = \theta_{tray_{32}} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch})$
- Rodilla $\rightarrow \theta'_{mod_{41}} = \theta_{tray_{41}} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch})$
- Rodilla $\rightarrow \theta'_{mod_{42}} = \theta_{tray_{42}} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch})$
- Tobillo $\rightarrow \theta'_{mod_{51}} = \theta_{tray_{51}} - K_{p_{Pitch}}(e_{pitch}) - K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau - K_{d_{Pitch}}(\Delta e_{Pitch})$
- Tobillo $\rightarrow \theta'_{mod_{52}} = \theta_{tray_{52}} + K_{p_{Pitch}}(e_{pitch}) + K_{i_{Pitch}} \sum_0^t e_{pitch}(\tau)\Delta\tau + K_{d_{Pitch}}(\Delta e_{Pitch})$
- Tobillo $\rightarrow \theta'_{mod_{61}} = \theta_{tray_{61}} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$
- Tobillo $\rightarrow \theta'_{mod_{62}} = \theta_{tray_{62}} + K_{p_{Roll}}(e_{roll}) + K_{i_{Roll}} \sum_0^t e_{roll}(\tau)\Delta\tau + K_{d_{Roll}}(\Delta e_{Roll})$

# Capítulo 5

# Sistema embebido

## 5.1 Antecedentes

Los primeros sistemas embebidos fueron creados para las misiones espaciales en la decada de los años 60, los sistemas debían ser autónomos y poder funcionar bajo condiciones espaciales; es decir, radiación y temperaturas extremas principalmente[27], entre otras.

Un sistema embebido se puede definir como un sistema electrónico diseñado para tareas específicas, con un procesamiento digital, con lo cual un procesador es implícito, pueden venir en varios tipos, como microprocesadores, microcontroladores, DSP, FPGA. Los sistemas embebidos son considerados inteligentes, ya que gobiernan al hardware donde son anfitriones; podemos poner ejemplos como cámaras de video, celulares, juguetes [27]. En el caso de las microcomputadoras, existen sistemas operativos específicos o de propósito general que posteriormente puede ser programado para un objetivo determinado; por ejemplo, Embedded Linux.

En Linux hay diferentes distribuciones (diferente kernel) dedicadas a sistemas embebidos, en general están orientadas a las tareas en paralelo (hilos) y el uso de interrupciones, funcionando solo lo que electrónicamente es necesario, esto permite el ahorrar energía a largo plazo. [48].

Los sistemas embebidos constan de dos partes esenciales:

**Hardware**

El sistema embebido comúnmente se aloja en otro llamado anfitrión. Para el diseño de un sistema embebido se deben considerar variables externas del hardware, tales como las condiciones ambientales de uso, temperatura, humedad, l consumo de energía, la robustez mecánica (minimización de vibraciones), costo del producto, cables de comunicación, etc. Todo dependerá totalmente de la tarea o proceso a realizar.

**Software**

El software del sistema estará configurado según el producto y el usuario quien lo maneje y estará limitado por los recursos del hardware, por ejemplo el tipo de compilador y lenguaje utilizado.

La expectativa general en los sistemas embebidos es que trabajen en tiempo real, o al menos con gran

velocidad. Los programas utilizados deberán ser óptimos para la aplicación deseada según los recursos con que se cuenten (fuente de energía, microcontroladores, dispositivos externos, etc).

Por lo general los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo o utilizando los compiladores específicos, también pueden utilizarse lenguajes como C o C++ cuando el el tiempo de respuesta de la aplicación no es un factor crítico.

Existen también plataformas desarrolladas por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos con sistemas embebidos desde ambientes gráficos, y académicos, algunos ejemplos son: Arduino, Mbed, Raspberry Pi, BeagleBone, etc.

## 5.2 Desarrollo del sistema embebido

Con el fin de hacer un sistema eficiente en el procesamiento, recibo y envío de datos, se decidió hacer las tareas de procesamiento, adquisición y actuación por partes. Existen tres problemas principales: el procesamiento de los controladores difusos, la adquisición de datos (sensores de fuerza en la planta de los pies y la posición angular real en cada articulación) y la actuación de los servomotores del robot bípedo. Siguiendo esta misma línea, se decidió utilizar un CPU para los controladores difusos, un microcontrolador como tarjeta de adquisición y otro microcontrolador como una tarjeta de actuación de los servomotores del robot bípedo (tarjeta driver de servomotores).

Como tarjeta de adquisición se definió al microcontrolador Teensy 3.1 previamente mencionado en el capítulo 3, y como tarjeta driver, al microcontrolador Tiva-C. Posteriormente se describen a detalle las funciones de cada tarjeta.

La solución más asequible de sistemas embebidos comerciales para un CPU, es una microcomputadora académica llamada Raspberry Pi 3B+, la cual no cumple con ningún estándar industrial ni de seguridad. Sin embargo, esta microcomputadora es pequeña, potente y capaz de ser embebida; pero sobre todo, puede realizar procesamiento con mayor velocidad que un microcontrolador y almacenar las trayectorias de caminata al mismo tiempo. Ésta microcomputadora utiliza una distribución de Linux recortado llamado **Raspbian** por lo que, presenta todas las ventajas de Linux; es decir, un sistema operativo abierto. Otra ventaja es que puede ser programada en distintos lenguajes. Un aspecto que se destaca es la comunicación ya que, al poderse conectar remotamente con un router, permite al robot ser semi-autónomo (exceptuando la energía eléctrica, que proviene de una fuente externa).

### 5.2.1 Unidad central de procesamiento - Raspberry pi 3B+ ®

La Raspberry Pi 3B+ es la tercera generación de las microcomputadoras Raspberry Pi ®. Reemplazó a la Raspberry Pi 2B+ en febrero 2015. Esta tarjeta contiene las siguiente propiedades:

- Un procesador quad-core ARM Cortex-A7 de 1200 Mhz
- 1 GB de RAM
- 4 puertos USB
- 40 pines (entrada/salida) de propósito general
- Puerto HDMI
- Puerto Ethernet
- Ranura de tarjeta micro SD

La tarjeta Raspberry Pi 3B+ se empleará para el procesamiento de los controladores difusos y la adquisición de datos de la IMU. Por la velocidad de procesamiento, y la capacidad de manejo de archivos, se empleó como CPU del robot.

Las ventajas de utilizar una microcomputadora con un sistema operativo basado en linux son sustanciales. Se puede tener total control sobre los procesos que se ejecutan. Se pueden emplear distintos protocolos de comunicación, en especifico las comunicaciones usadas por el sistema serán: comunicación serial e $I^2C$.

La comunicación $I^2C$ tiene diferentes modos de uso, maestro-esclavo, esclavo-esclavo y multi-maestro. Cabe mencionar que este protocolo de comunicación es bidirecional; es decir, de lectura y de escritura. Es un protocolo que utiliza un mecanismo de transmisión de punto a punto; sin embargo, el protocolo permite asignar direcciones, lo que permite tener una organización en el envío, cuando se conecta por ejemplo, un maestro con muchos esclavos. Es un tipo de comunicación síncrona; es decir, que su funcionamiento está basado en el uso de un reloj para la sincronía (SCL- Serial CLock) entre módulos y otro cable para la transmisión de datos (SDA - Serial DAta) entre módulos. Otro aspecto importante es la calidad de la información. Este protocolo involucra un orden especifico en su envío (coordinación) por lo que, la transmisión suele ser segura y rápida. En la Figura 5.1 se muestra un paquete de envío, a modo de ejemplo para la transmisión de datos.

El paquete se organiza de la siguiente forma:

- Inicio: Condición de inicio.
- Dirección: Para el reconocimiento entre Maestro-Esclavo, normalmente escrita en hexadecimal, por ejemplo 0x7F.
- L/!E: Se utiliza para diferenciar entre modo lectura o escritura.
- R: Se ocupa para el reconocimiento del envío; es decir, es un bit de seguridad que asegura que el paquete llegue al destino.
- Parada: Se termina la transmisión de datos.

El protocolo $I^2C$ es utilizado para la transmisión de los ángulos teóricos a la trajeta esclavo driver de los servomotores y la adquisición de los ángulos reales será mediante la otra tarjeta de adquisición esclavo. Por otro lado, el CPU establece una comunicación serial a 115200 baudios con la IMU para obtener la orientación del CoM. Específicamente la IMU envía 4 datos: ángulo pitch, roll y el cambio de éstos: pitch_rate, roll_rate. Para conocer detalles sobre instrumentación, implementación y lectura de la IMU consultar [30].

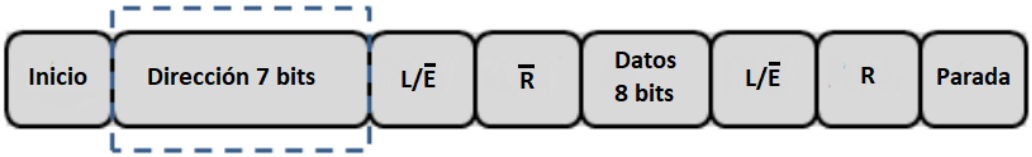


Figura 5.1: Paquete de transmisión $I^2C$

Por su parte la Raspberry Pi 3B+, recolectará datos en-linea, es decir durante el control de postura o marcha, los datos recabados son: tiempo de ejecución en segundos (cuenta de tiempo que aumenta conforme avanza el tiempo), error pitch, error roll, cambio del error pitch, cambio del error roll, suma del error pitch, suma del error roll, ganancias (6), acciones de control (12), ángulos modificados (10). Para la suma del error se consideró $\Delta\tau = 8\,[mS]$. Para detalles sobre la implementación del esquema de control para el robot bípedo Scout, ver sección 5.3.

### 5.2.2 Tarjeta de adquisición - Teensy 3.1 ®

Teensy 3.1 (ver Figura 5.2 a)), es una tarjeta de desarrollo con arquitectura ARM Cortex M4. Es la tercera generación de éste tipo. Mantiene una gran velocidad de procesamiento con un reloj en overclock de 96 Mhz. Su tamaño, y potencia de procesamiento le permiten utilizarse para la adquisición de datos, y su procesamiento. Posee las siguientes características:

- Microcontrolador MK20DX256VLH7, 72 MHz, o 96 Mhz en overclock
- SRAM de 64KB
- 34 pines GPIO
- 1 DAC de 12 bits
- 1 SPI
- PWMs de control de 12 bits
- Flash de 256KB
- EEPROM de 2KB
-Interfaz USB 2.0
- 3 UART, 2 I2C, 1 I2S Audio,
- módulo CAN
- ADCs de 12-bits

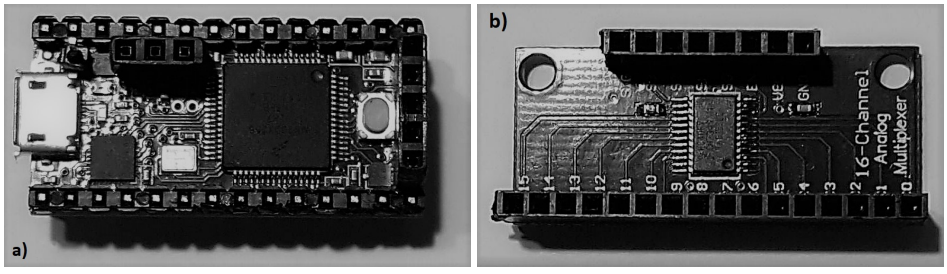


Figura 5.2: a) Teensy 3.1 b) Multiplexor 16 canales

Esta tarjeta permite hacer el procesamiento de los sensores de fuerza y los potenciómetros de cada servomotor, con el uso de dos multiplexores de 16 canales (ver Figura 5.2 b)). Para hacer una correcta interpretación en la medición de los sensores de fuerza, así como de los potenciómetros de cada servomotor se utilizaran las caracterizaciones hechas por Saddan Hernández en su trabajo de Tesis [30].

La lectura de los sensores de fuerza se hace mediante el uso del ADC incluido en la tarjeta, el objetivo de utilizar un multiplexor es no saturar las entradas analógicas del controlador, ya que son 16 sensores en total.

Para el procesamiento de los sensores, se utilizaron las caracterizaciones hechas en [30], con el fin obtener las fuerzas reactivas en la planta de los pies del robot bípedo. Posteriormente estas fuerzas reactivas se utilizaron para el calculo del CoP y a su vez del ZMP; tal y como se muestra en la Sección 2.3.3 y 2.3.4 .

Al igual que los sensores de fuerza, la lectura de los ángulos reales de los servomotores se hace utilizando uno de los ADC incluidos en la tarjeta, con ayuda de un multiplexor de 16 canales. Cada servomotor fue caracterizado así que, simplemente se transforma el voltaje medido al ángulo real o actual del servomotor. En la Figura 5.3 se detllada el esquema de adquisición elaborado.

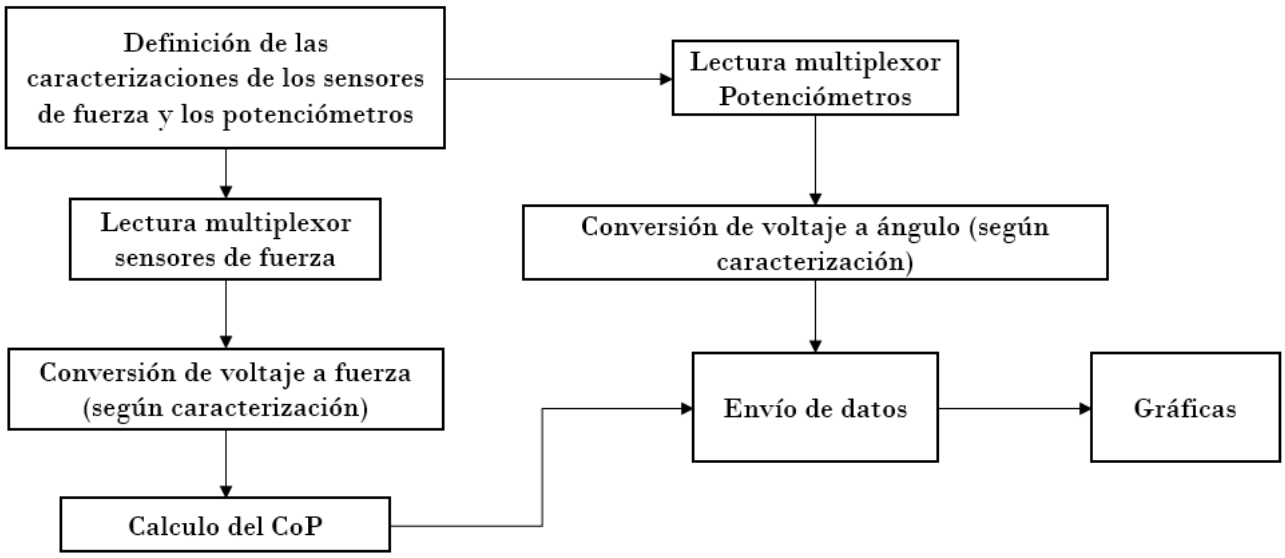


Figura 5.3: Esquema conceptual de adquisición

### 5.2.3 Tarjeta driver de servomotores - Tiva-C TM4C123GXL ®

La tarjeta TM4C123GXL es una tarjeta de desarrollo de bajo costo para microcontroladores basados en la arquitectura ARM Cortex M4F de Texas Instruments. La tarjeta contiene el microcontrolador TM4C123GH6PM con una interfaz USB 2.0 y un módulo de hibernación. Posee las siguientes características:

- Microcontrolador TM4C123GH6PM de 32 bits con 80 MHz
- SRAM de 32KB
- 40 pines GPIO
- Interfaz integrada de depuración
- 4 SPI
- PWMs de control de 12 bits
- Entrada USB micro
- Flash de 256KB
- EEPROM de 2KB
-Interfaz USB 2.0
- 8 UART, 2 I2C
- 2 módulos CAN
- ADCs de 12-bits

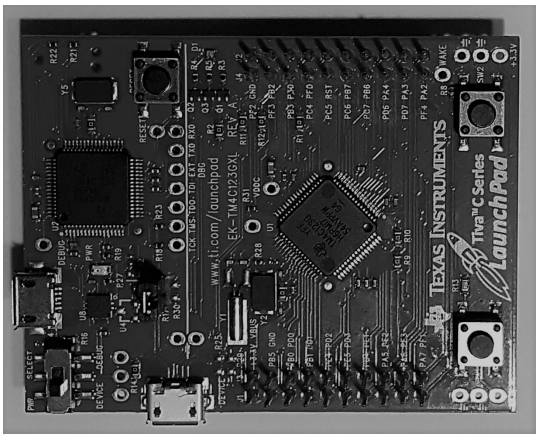


Figura 5.4: Tiva-C TM4C123GXL

La tarjeta Tiva-C (así comúnmente conocida) se empleará para el control (driver) de los servomotores. El principal objetivo de esta tarjeta será mantener una postura en posición de "home"para el robot bípedo. Para asegurar la postura de "home.º ángulos iniciales siempre que reinicie la tarjeta se utilizaron funciones tipo **callback**; es decir, que solo trasmiten información que viene desde el maestro solo si él lo pide, sino pasa esto, no hay procesamiento de ningún tipo. El CPU (maestro) le enviará la información correspondiente a la posición angular deseada y éste le corresponderá. Las señales de **modulación por ancho de pulso** (PWM , por sus siglas en inglés) serán enviadas y sincronizados por el timer interno de la tarjeta.

Generalmente el rango de operación de los servomotores va desde $0$ a $180°$; sin embargo, existen otro tipo de servomotores *continuos*, capaces de dar más de una vuelta, por ejemplo los servomotores Dynamixel ®. Los servomotores utilizados para el robot bípedo Scout son del primer tipo y se controlan mediante el ancho de pulso con un tiempo de 900 [$\mu s$], para el ángulo $0°$ y 2100 [$\mu s$], para $180°$.

Dado que la **Raspberry pi 3B+**, **la Tiva - C** y la tarjeta **Teensy 3.1** tienen una lógica de 3.3 [v], la comunicación es directa, sin convertidores de voltaje.

### 5.2.4 Diagrama de conexión

Se muestra en la Figura 5.5 la conexión entre las tarjetas del sistema embebido. Para asegurar la llegada de los datos de las acciones de control; es decir, los datos dirigidos a los servomotores, se utilizó un protocolo $I^2C$. La comunicación $I^2C$ también fue implementada para capturar los datos reales de los potenciómetros de los servomotores, en caso de requerir enviarlos a la Raspberry para su posterior procesamiento (opcional). Mientras que los demás protocolos de comunicación son del tipo serial debido a su sencillez. La comunicación serial tendrá dos conexiones: la IMU UM7-lt hacia la Raspberry y de la tarjeta de adquisición de datos Teensy 3.1 hacía una computadora externa para la extracción de datos y analizarlos posteriormente.

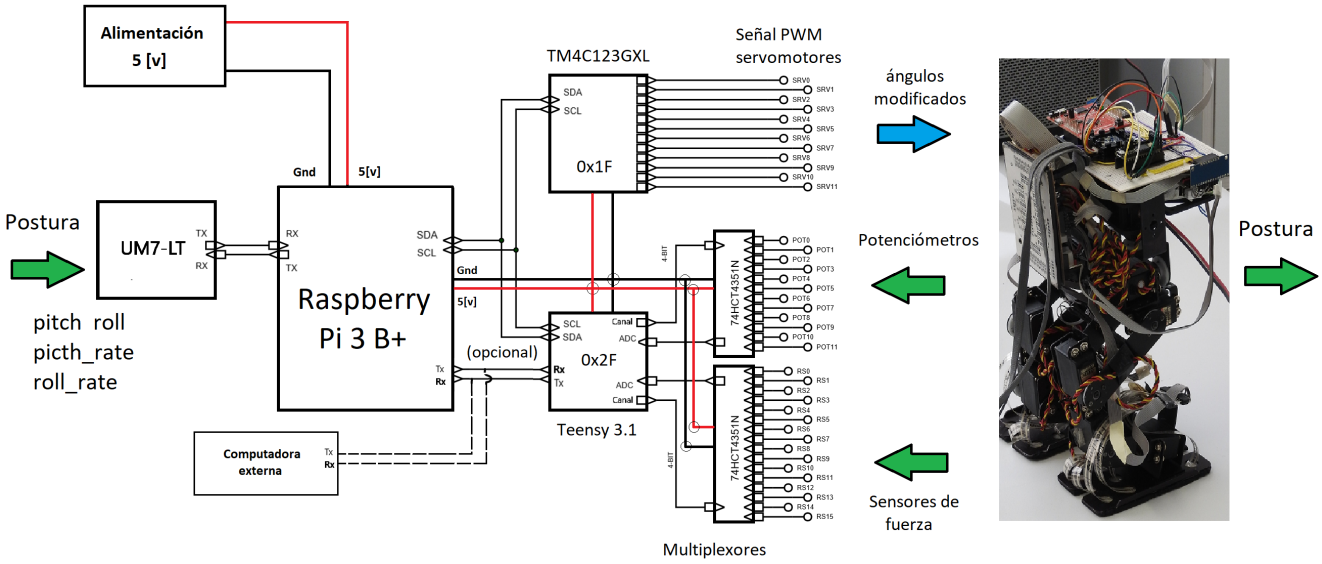


Figura 5.5: Conexiones principales del sistema embebido

Todas las conexiones visualizadas en la Figura 5.5 fueron alambradas en una tarjeta perforada tipo *steren 150-cc*. En esta tarjeta perforada se alambraron las lineas de comunicación, la tierra y el voltaje necesario para cada caso. En cuestiones simples, existen dos partes principales del sistema: la unidad central de procesamiento y las tarjetas esclavas dedicadas a la adquisición y la actuación, dichas tarjetas estarán sobre la tarjeta perforada.

### 5.2.5 Esquema conceptual del sistema embebido

Como complemento al diagrama de la Figura 5.5, se describe conceptualmente el flujo de información en la Figura 5.6.

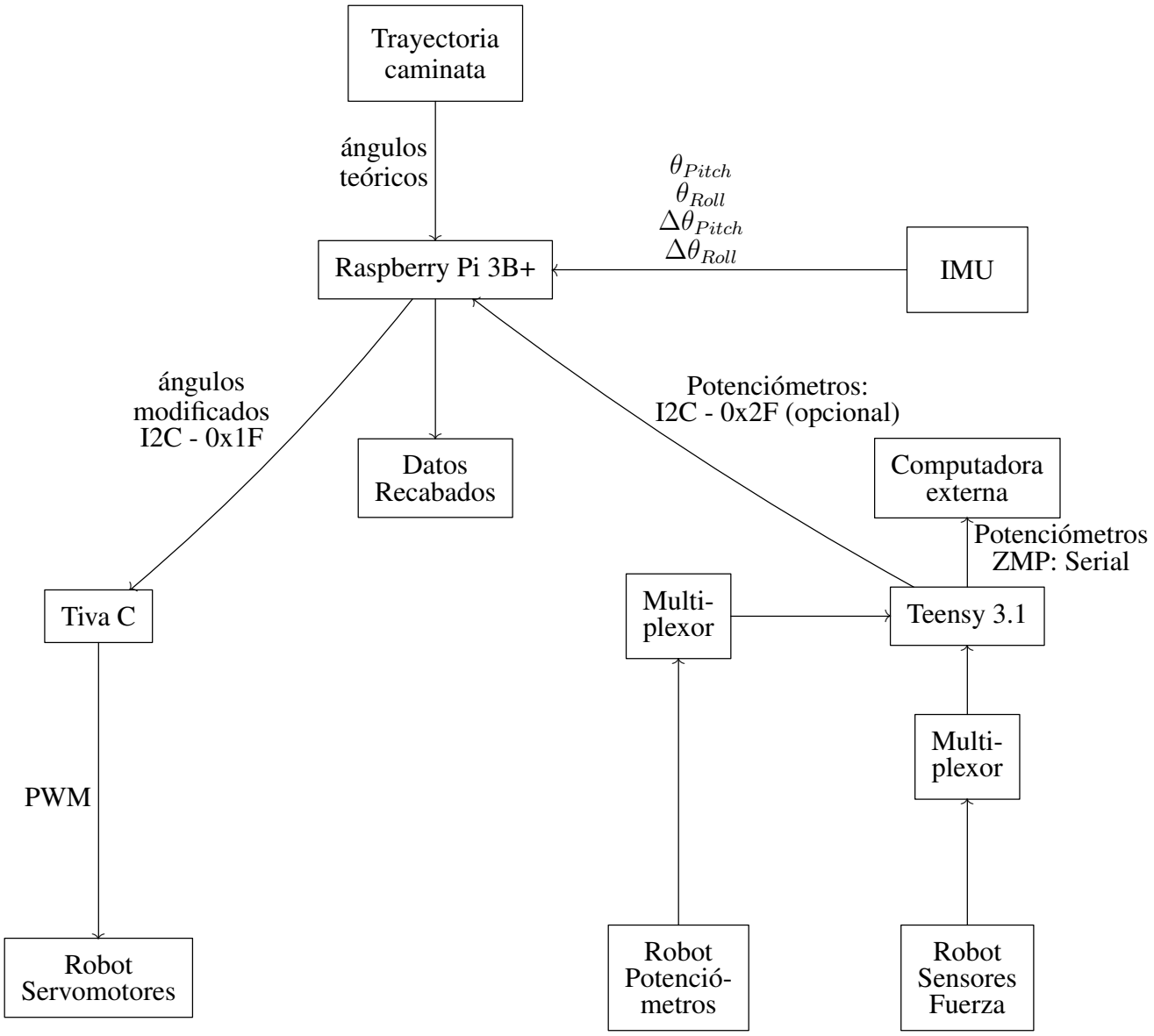


Figura 5.6: Esquema conceptual del sistema embebido

## 5.3 Implementación del esquema de control

### 5.3.1 Configuración preliminar

En la Figura 5.7 a) se muestra el circuito montado en el robot. Se optó por colocarlo en la parte trasera de la cadera. El sistema fue probado con cada una de las trayectorias de caminata, las primeras pruebas tuvieron lugar a plena capacidad de procesamiento de la Raspberry. El procesamiento fue rápido y tomo 5 segundos en completar una trayectoria; sin embargo, esta rapidez le imposibilitó al robot caminar por lo que, se ajusto el tiempo de envío a uno lo suficientemente grande como para hacer una caminata en lazo abierto sin complicaciones. Comparado con el envío de trayectorias en [29], que tardaba cerca de un minuto y medio en completar el envío de una sola trayectoria al robot, se obtuvo una reducción en tiempo de transmisión de 18 veces aproximadamente.

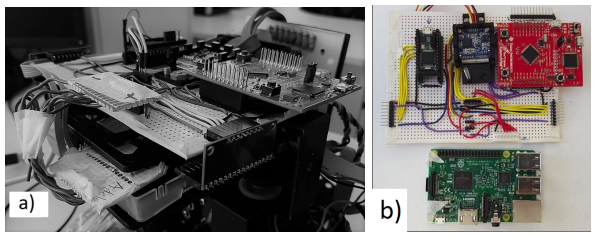


Figura 5.7: a) Circuito montado y armado sobre el robot bipedo b) Vista general del circuito

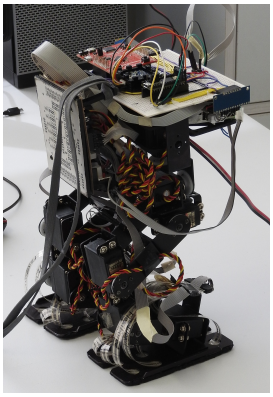

Figura 5.8: Robot bípedo con el sistema embebido

En la Figura 5.8 se muestra la pseudo autonomía de la que goza el robot, ya que está ligado permanentemente a la fuente de energía. sin embargo no le imposibilita ninguna estrategia de control. El sistema incrementa el peso de la cadera, por lo que en ocasiones en lazo abierto el robot tiende a caer; sin embargo, esta perturbación se ve compensada en lazo cerrado.

Se colocó el sensor inercial (IMU) lo más próximo al centro de masa. En los costados se colocaron los multiplexores. Los cables de las señales de PWM para los servomotores estarán justo en el lado superior derecho de la tarjeta, como se puede ver en la figura 5.7 b) para minimizar la distancia entre el sistema y la conexión del robot.

Para un correcto funcionamiento del protocolo $I^2C$, es necesario hacer pruebas preliminares de conexión. En la Figura 5.9, se muestra la detección de los bus por parte de la Raspberry; esta evaluación se hizo a través de la instrucción *sudo i2cdetect -y 1*, la cual hace un barrido de todos los puertos disponibles a conexión y muestra aquellos que estén ocupados por alguna dirección. De esta forma se hace muy claro y evidente la conexión establecida.

Las direcciones asignadas en el protocolo $I^2C$, para cada tarjeta son las siguientes:

- TM4C123GXL: **0x1F**, Teensy 3.1: **0x2F**

En la Figura 5.10, se muestra el envío de ángulos por medio de la Raspberry hacia la tarjeta Tiva-C driver de los servomotores; para que el robot pueda seguir las trayectorias el tiempo de envío quedó totalmente a merced del tiempo de retraso. En la misma Figura se observa el orden de envió, empezando por el servomotor 12 (se ocupa la misma convención de nombres que en la Figura 3.1) hasta el servomotor 61.

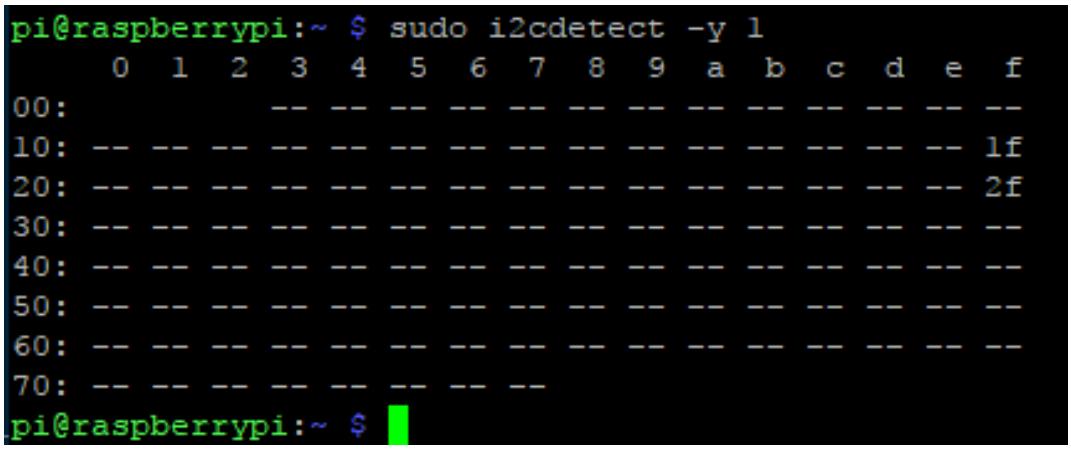


Figura 5.9: Prueba de conexión I2C entre Raspberry y los microcontroladores esclavos

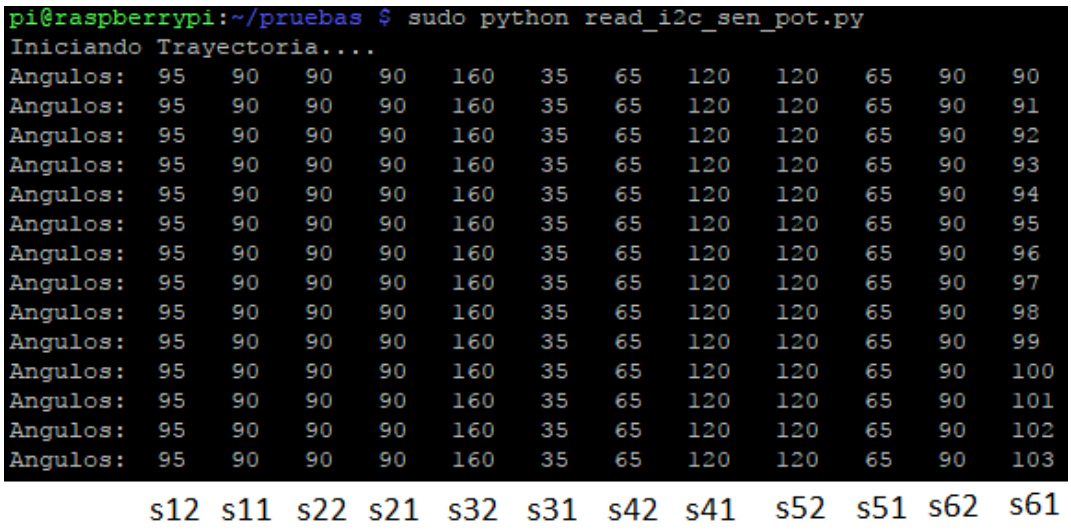


Figura 5.10: Prueba de conexión entre Raspberry y la tarjeta driver de los servomotores

### 5.3.2 Interfaz gráfica

La interfaz gráfica facilita el manejo del robot bípedo, esta permite la selección entre: una caminata en lazo abierto, un control de postura y el control de marcha o caminata. De esta forma se le permite al usuario del robot controlarlo de manera intuitiva. La interfaz fue diseñada mediante el uso de un módulo de python llamado *tkinter*, éste permite la creación de ventanas, etiquetas, botones, etc.

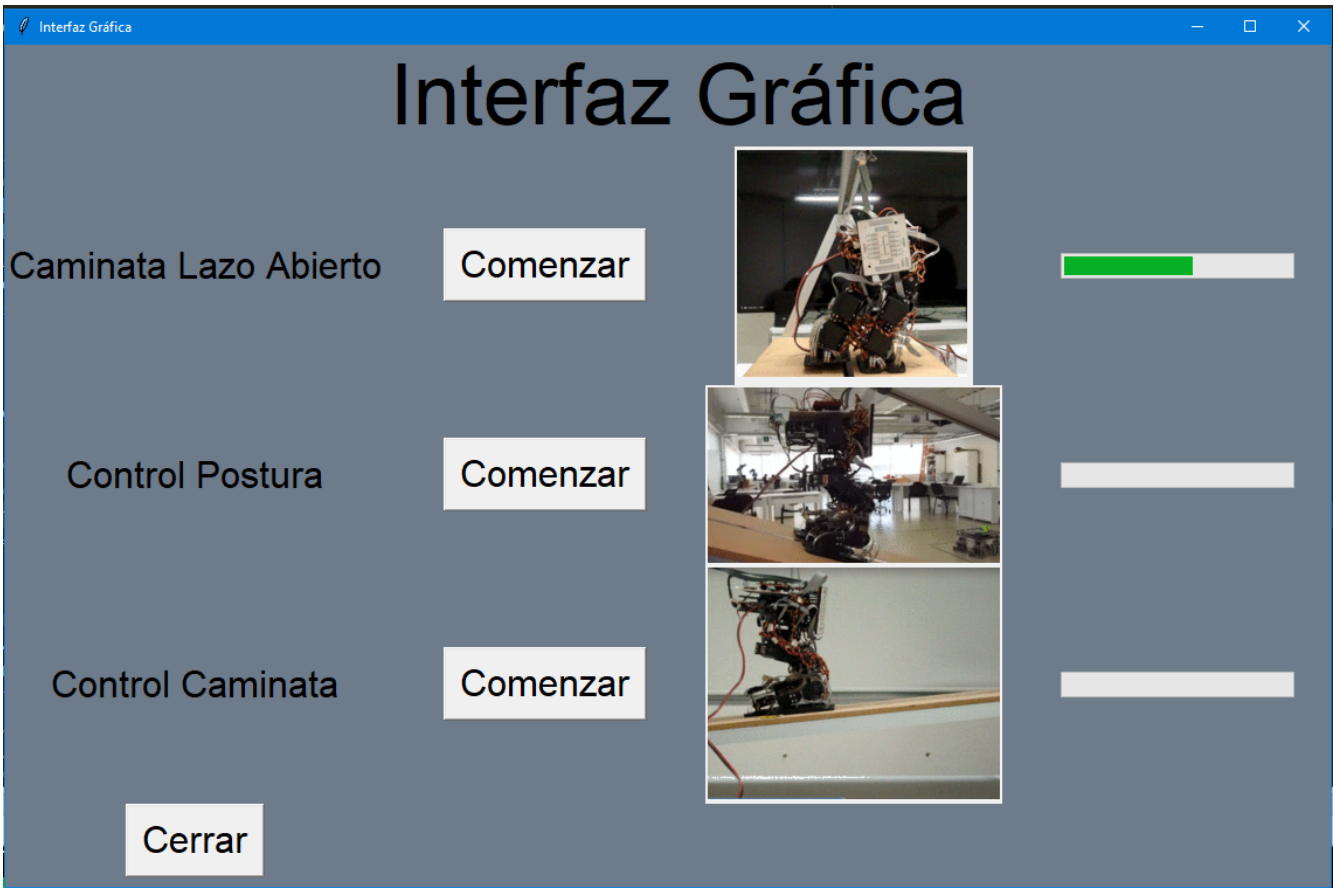


Figura 5.11: Interfaz gráfica (Elaborada en colaboración con Enrique y Diana en [43] [42] )

#### 5.3.2.1. Funcionamiento

El protocolo de comunicación utilizado entre el sistema embebido y la computadora externa fue TCP/IP, el cual es utilizado para el envío de datos de manera segura entre un cliente y un servidor. Se crearon casos lógicos para decidir entre las tres diferentes estrategias de control disponibles. Como se muestra en la Figura 5.12, el cliente se encuentra en una computadora externa, mientras que el servidor se aloja en el robot. El cliente le envía una señal de inicio al servidor y este actúa según el caso lógico seleccionado.

Una de las ventajas del protocolo TCP/IP es que puede ser utilizado mediante una red inalámbrica o alámbrica. Las ventajas del Wi-Fi es la distancia de operación ya que, no se requiere estar a lado de la

computadora externa dónde se encuentra el cliente y la distancia estará limitada al alcance de la señal emitida por el router o repetidor. Por otro lado la comunicación mediante Wi-Fi es lenta si se compara con la conexión física usando el cable Ethernet, la cual se verá restringido por la longitud del mismo.

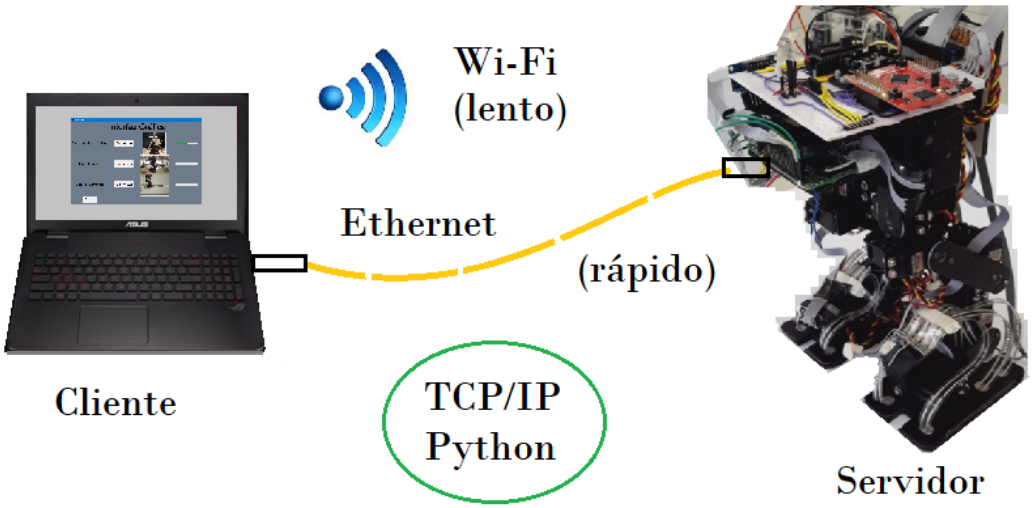


Figura 5.12: interfaz

### 5.3.3 Esquema conceptual del algoritmo de control

En las Secciones 5.3.3.1 y 5.3.3.2, se observa el diagrama de flujo perteneciente a los controladores difusos previamente diseñados y se muestran conceptualmente los algoritmos seguido para el control de postura y marcha.

#### 5.3.3.1. Control de postura

El control de postura inicia definiendo las variables lingüísticas para los controladores difusos. Habiendo definido las variables lingüísticas se prosigue con un bucle infinito de tipo *while* o finito de tipo *for*. El bucle será detenido solo en el caso de terminar con el intervalo (bucle tipo for), una interrupción por teclado o el fallo de alguno de los componentes.

En el bucle, se inicia el contador de tiempo (tiempo de ejecución) y se hace la lectura de la IMU. Con los datos obtenidos por la IMU y el tiempo transcurrido se procede al calculo de la suma del error, en dirección Pitch y Roll. Posteriormente se genera la fusificación de los datos de entrada, para después hacer la inferencia lógica, obtener una función agregada, con la cual se puede aplicar el método del centroide para defusificación con el objetivo de calcular las ganancias y las salidas de control. Al terminar, estas ganancias se introducen al controlador PID-ASLD. Las salidas para el control difuso de postura se suman al control PID-ASLD.

Durante y al finalizar el control de postura se recolectan datos. El tamaño de la adquisición de datos

dependerá de la dimensión del vector de datos recabados (ver Apéndice **??**). Al concluir la recolección de datos el CPU envía los nuevos ángulos a la tarjeta driver cerrando el lazo de control.

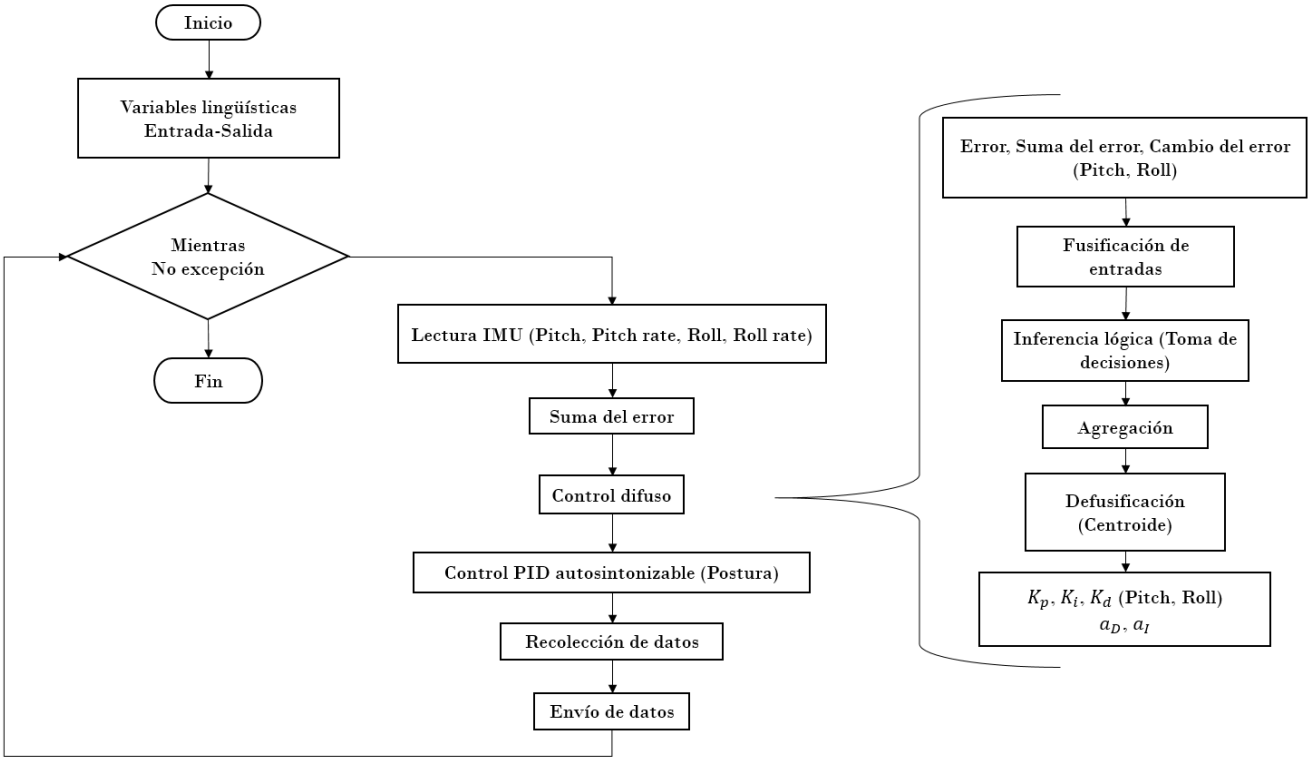


Figura 5.13: Esquema conceptual del control de postura

#### 5.3.3.2. Control de marcha

Para el control de marcha del robot bípedo se utiliza el control de postura, es decir utiliza el mismo flujo de procesos (variables lingüísticas, lectura IMU, calculo de la suma del error, etc.). Sin embargo, el control de marcha sólo utiliza el control PID-ASLD, dejando de lado el control difuso de postura en dirección Roll. El objetivo principal de éste controlador es la modificación de trayectorias previamente definidas por lo que, al iniciar el control de marcha es necesario definir que tipo de trayectoria se requiere.

La modificación de la trayectoria se hace actualizando los ángulos para cada servomotor y las acciones de control. Como se vió en el diseño de los controladores difusos, la modificación se hace adicionando al ángulo de la trayectoria las acciones de control. El ángulo modificado se adapta a la taryectoria; es decir, que no se modifica la forma de la trayectoria sino más bien la traslada; Se darán mas detalles en el Capítulo 6.

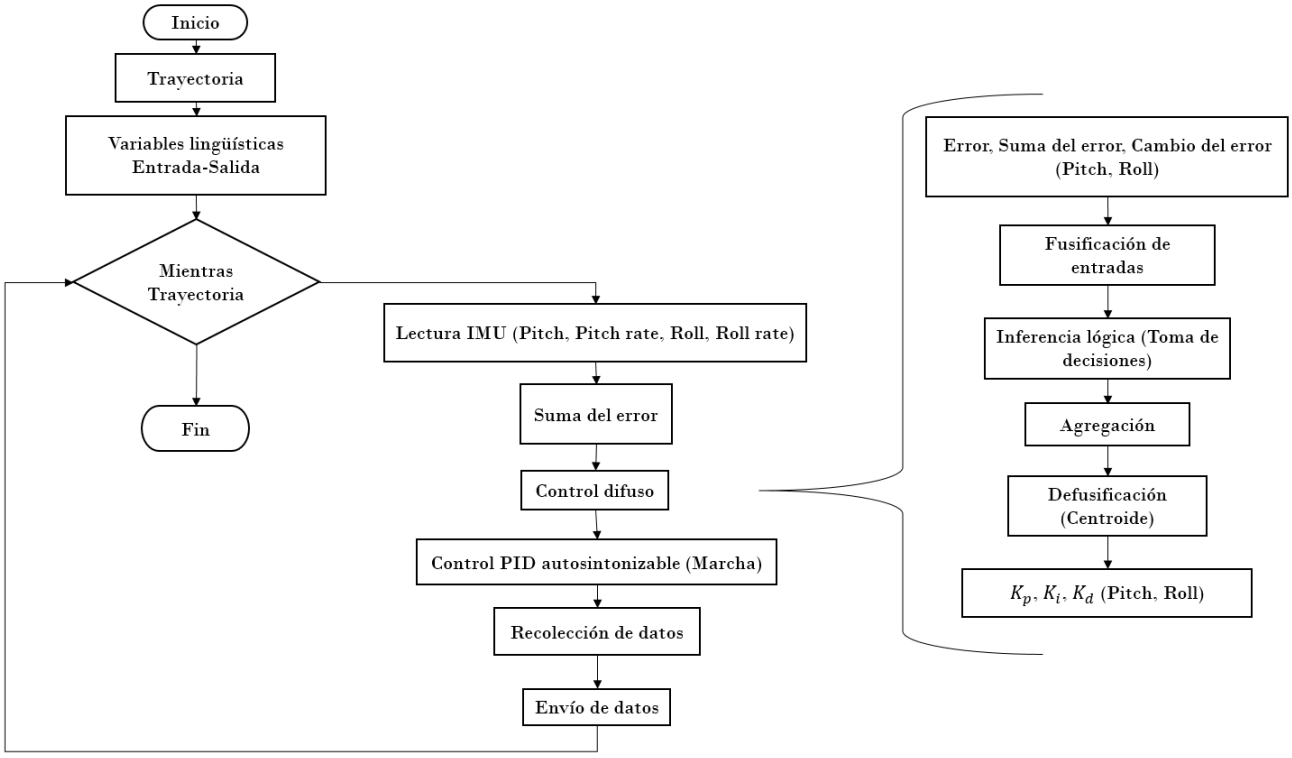


Figura 5.14: Esquema conceptual del control de marcha

# Capítulo 6

# Análisis de resultados

## 6.1 Control de postura

El control de postura se probó poniendo al robot sobre un terreno plano que incrementa gradualmente su pendiente hasta $15°$. La evolución de las ganancias sintonizadas se puede ver en la Figura 6.1, donde se aprecia como $K_p$ y $K_d$ aumentan con respecto a la magnitud del error, que aumenta en el tiempo ya que la pendiente se incrementa después del primer segundo y se mantiene constante a partir del quinto. Para la ganancia $K_i$, ésta se hace pequeña cuando el error incrementa, esto se hace para evitar la saturación de la acción de control en dirección Pitch, en caso de que la suma del error sea demasiado grande.

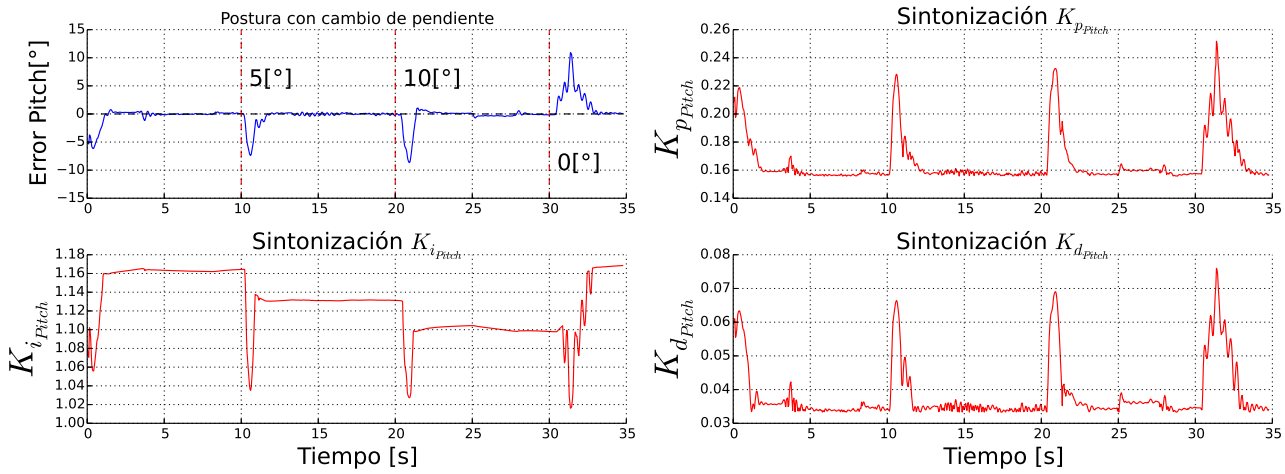


Figura 6.1: Control de postura: PID-ASLD versus PID tradicional

La Figura 6.2 muestra un compendio de imagenes que representan el control de postura en pleno funcionamiento sobre la plataforma de pruebas del robot bípedo. Las imagenes oscilan el ángulo de la plataforma entre 0 y 20 grados.

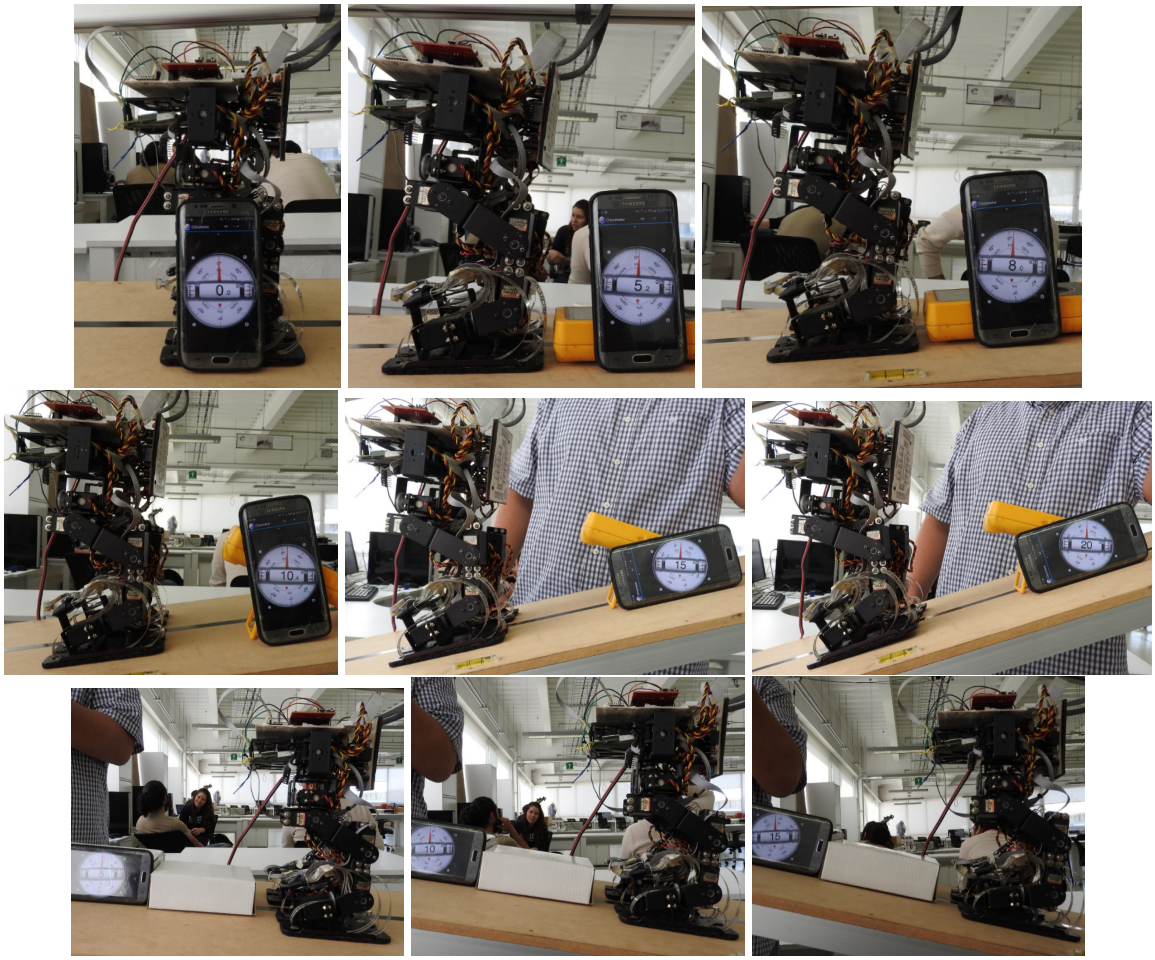

Figura 6.2: Control de postura sobre la plataforma de pruebas a diferentes grados de pendiente

En la Figura 6.3 a), se muestra la comparación entre un controlador PID tradicional y un controlador PID-ASLD. Se observa un comportamiento similar entre los dos controladores; sin embargo, se observa una leve mejora en el tiempo de asentamiento del controlador PID-ASLD. El PID tradicional se diseñó a partir del promedio de las ganancias $K_p$, $K_i$ y $K_d$ registradas en previos experimentos con el control PID autosintonizable, teniendo un valor de: $K_p = 0.247953$, $K_i = 0.804010$ y $K_d = 0.056908$, estas ganancias son un resultado útil desde el punto de vista del procesamiento, ya que en el caso de no poder implementar un PID con ganancias variables, se podrían sintonizar ganancias estáticas que proporcionen un buen desempeño al controlador de manera relativamente sencilla. En las imágenes 6.3 b), 6.3 c) y 6.3 d), se muestra el comportamiento de las ganancias sintonizadas para el control PID-ASLD. En este experimento se colocó al robot sobre una plataforma de pruebas y después de un segundo, la plataforma fue levantada hasta 15 grados, finalmente, el robot se adaptó a la pendiente.

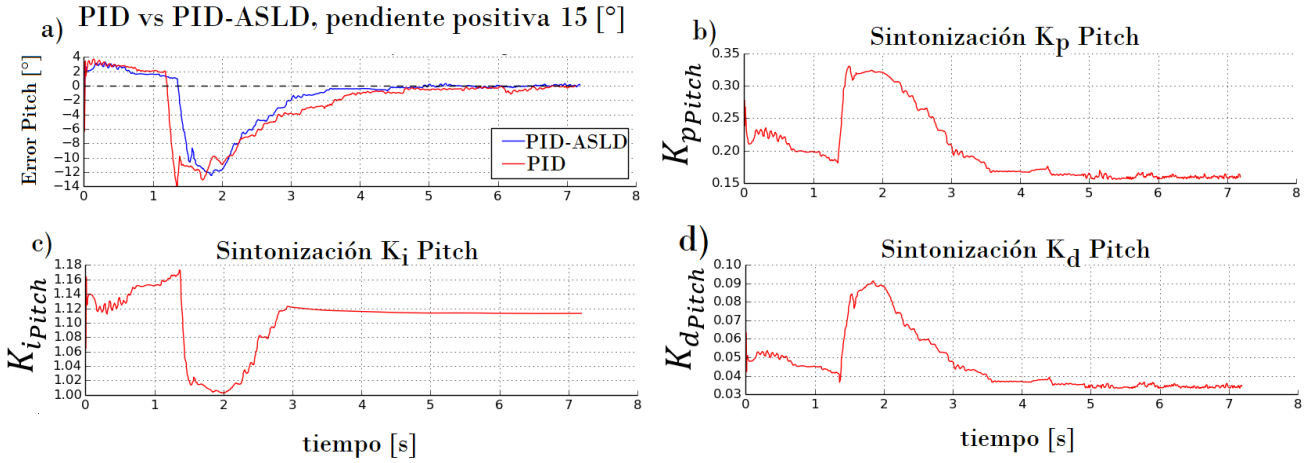


Figura 6.3: Control de postura: PID-ASLD versus PID tradicional

## 6.2 Control de marcha

En la secuencia de imágenes de la Figura 6.4, se observa el funcionamiento del control de marcha sobre la plataforma de pruebas del robot bípedo. La plataforma tenía 8 grados de inclinación positiva. Este experimento tuvo como objetivo mover la plataforma del robot bípedo con una inclinación arbitraria, y observar su comportamiento. Se determinó que el control de marcha tiene buen desempeño ante el cambio de pendiente [1]. En comparación a otro tipo de robots bípedos (humanoides, por ejemplo), este robot tiene una posición del CoM baja, esto permite que el control de marcha funcione con un desempeño aceptable con una pendiente de $\pm 10^{\circ}$ en dirección Pitch y $\pm 5^{\circ}$ en dirección Roll.

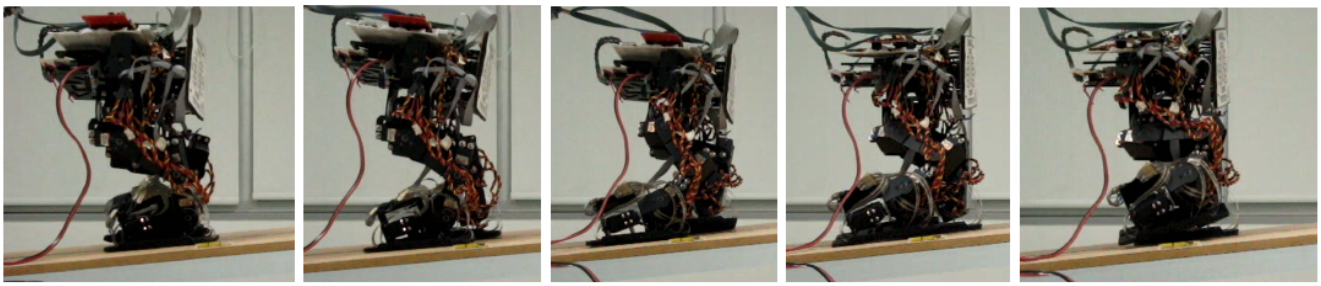

Figura 6.4: Control de marcha implementado en el robot bípedo

En el conjunto de gráficas de la Figura 6.5, se proporciona el comportamiento del error, del cambio del error en dirección Pitch, cuando el robot realiza una caminata sobre una superficie inclinada. En la parte inferior de la Figura se encuentran las ganancias $K_p$ y $K_d$, ambas dependen del error y del cambio del error, por lo que su comportamiento puede ser comparado con éstos. Se observa también el comportamiento de las ganancias conforme el avance del robot bípedo en un ciclo de marcha.

Se observa en la Figura 6.5 a), que en el intervalo de cero a cuatro segundos, la postura del robot es estabilizada por el control de postura, para que enseguida el ciclo de marcha inicie. Al desarrollarse la marcha, el robot cuenta con una oscilación entre $\pm 9^{\circ}$ a la referencia. Se puede observar una periodicidad en la gráfica, que es fruto del ciclo de marcha.

La tendencia cíclica de la Figura 6.5 a) se ve remarcada en las ganancia $K_p$ y $K_d$, de las Figuras 6.5 b) y d) , respectivamente. El cambio del error se destaca $K_p$ y $K_d$; cuando su cambio es grande, por ejemplo, justo antes del segundo ocho donde $|eD| > 20 \left[\frac{^{\circ}}{seg}\right]$, $K_p$ y $K_d$, también son grandes.

[1] Para ver al robot durante el control de marcha ver: video 1 https://www.youtube.com/watch?v=TPskJykRhyY, o video 2 https://www.youtube.com/watch?v=MEbZS5IdaTk

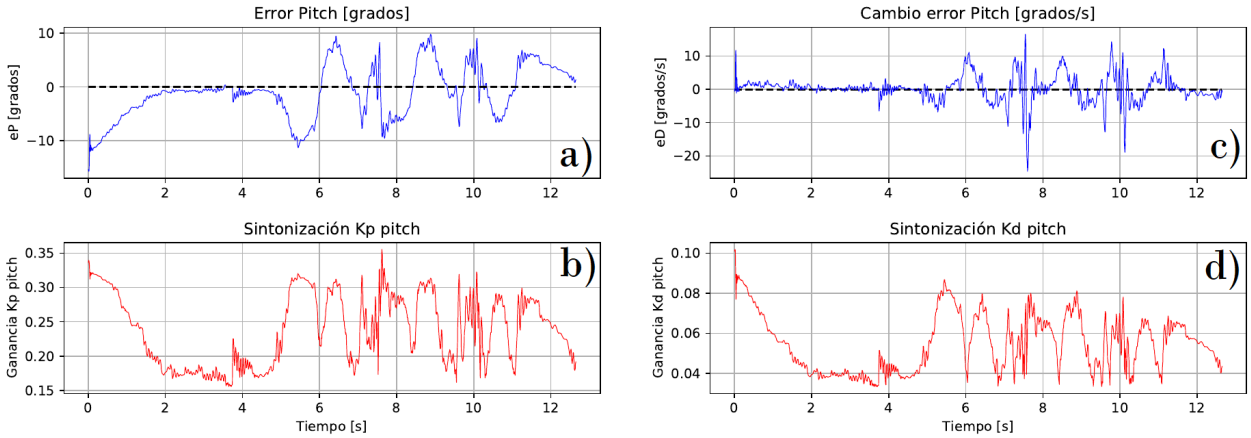


Figura 6.5: Respuesta del error y cambio del error en dirección Pitch. Comportamiento de las ganancias $K_p$ y $K_d$ en dirección Pitch

En general, la Figura 6.6, destaca el comportamiento de la ganancia Ki, la cual depende del error y de la suma del error. La Figura 6.6 b) describe la suma del error producto de la caminata del robot bípedo, mientras la estabilización tiene lugar la suma del error tiene un descenso suave. La suma del error durante la marcha oscila entre $\pm 20[^\circ]$, lo cual es consistente cuando el robot camina con pendiente positiva ya que, su centro de masa estará desplazado ligeramente hacía atrás causando una suma negativa en eI Pitch.

La ganancia Ki al depender de eP (Figura 6.6 a)), muestra comportamiento periódico en los mismos puntos de periodicidad del error; Sin embargo, al depender de la suma del error su comportamiento no es estrepitoso ya que, el valor de $K_i]$ oscila entre 0.9 y 1.1, lo cual, deja entrever la asociación con eI, resaltando que la definición de eI tiene un universo de discurso entre -300 y 300 su oscilación es relativamente pequeña.

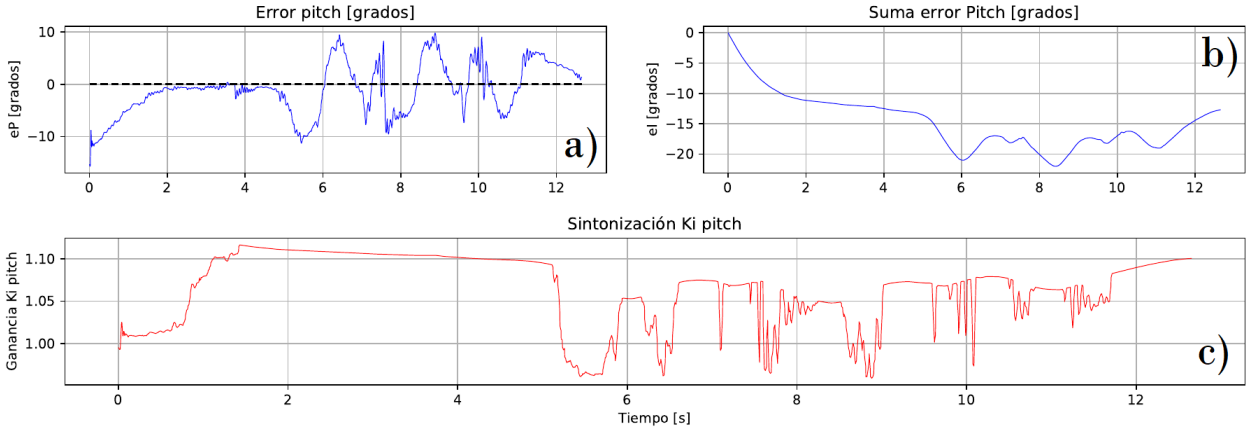


Figura 6.6: Respuesta del error y suma del error en dirección Pitch. Comportamiento de la ganancia $K_i$ en dirección Pitch

### 6.2.1 Modificación de trayectorias

Para cada articulación del robot bípedo existe una trayectoria teórica, por tanto, existen 12 trayectorias teóricas que el robot usa para poder caminar en lazo abierto, sólo 10 de las 12 serán modificadas con el objetivo de compensar las perturbaciones externas. Las trayectorias teóricas de cada articulación no cambiaran su forma, sino más bien se modificarán mediante un corrimiento (offset), que se ve reflejado, en cada una de las trayectorias, ya sea en dirección Pitch o Roll.

La Figura 6.7 muestra una comparación durante un ciclo de marcha del robot bípedo Scout entre las trayectorias teóricas y las trayectorias modificadas que el control de marcha proporciona. En negro se muestran las trayectorias teóricas y de color rojo las trayectorias modificadas. Dado que los servomotores solo trabajan en cierto intervalo, existen dos tipos de saturación: en $0°$ y $180°$.

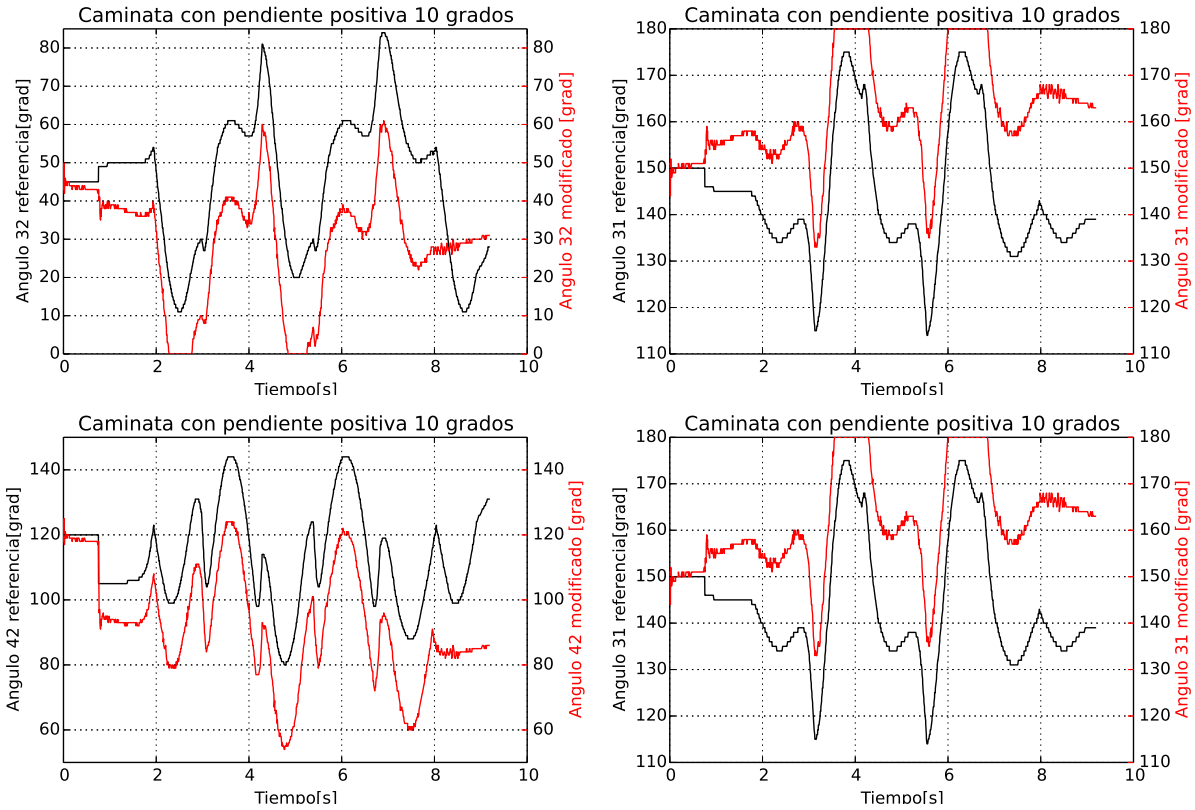


Figura 6.7: Modificación de trayectorias

### 6.2.2 ZMP - Marcha

Para fines visuales, los datos antes recopilados fueron graficados utilizando la forma del pie del robot. La posición del ZMP teórico está representado por un circulo de color rojo y las posiciones obtenidas durante la marcha por medio de asteriscos (*) de color negro. La planta del pie se detalla de color azul y la zona de seguridad se muestra de color rojo. Se destaca la oscilación que hay hacia una posición positiva y negativa en ambos pies durante la caminata; sin embargo, el pie derecho muestra oscilaciones del ZMP en la zona de seguridad, esto es debido al ruido interno del sistema, ya que, físicamente el robot camina de manera estable.

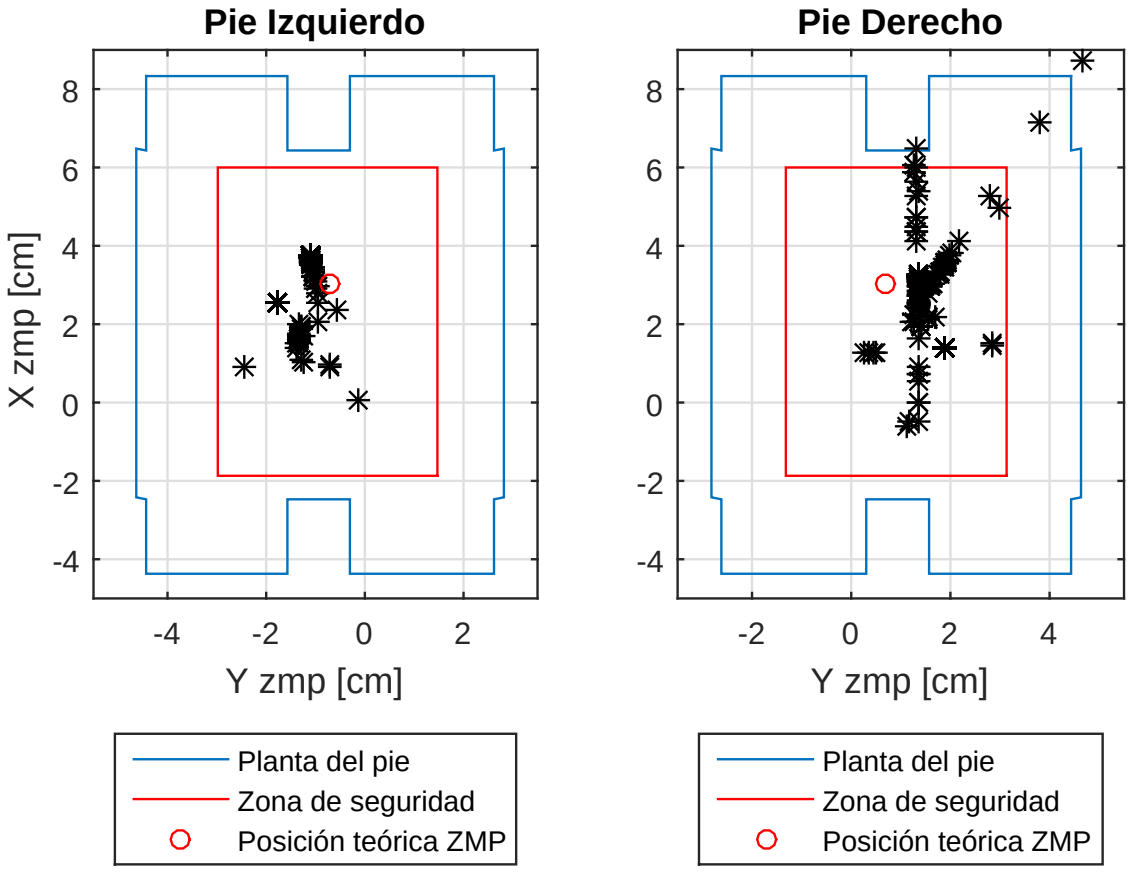


Figura 6.8: Gráfica de la posición del ZMP durante el control de marcha

Con el objetivo de observar la estabilidad del robot cuando se aplica el control de marcha, se decidió adquirir las coordenadas del ZMP durante 3 ciclos de marcha y 40 segundos de duración aproximadamente a $5^{\circ}$ positivos. En la Figura 6.9, se muestra de color rojo, los límites del polígono de soporte del pie izquierdo y derecho, ideales para que el robot no tienda a caer, se le puede considerar una zona de segura, en la cual, el ZMP puede contenerse.

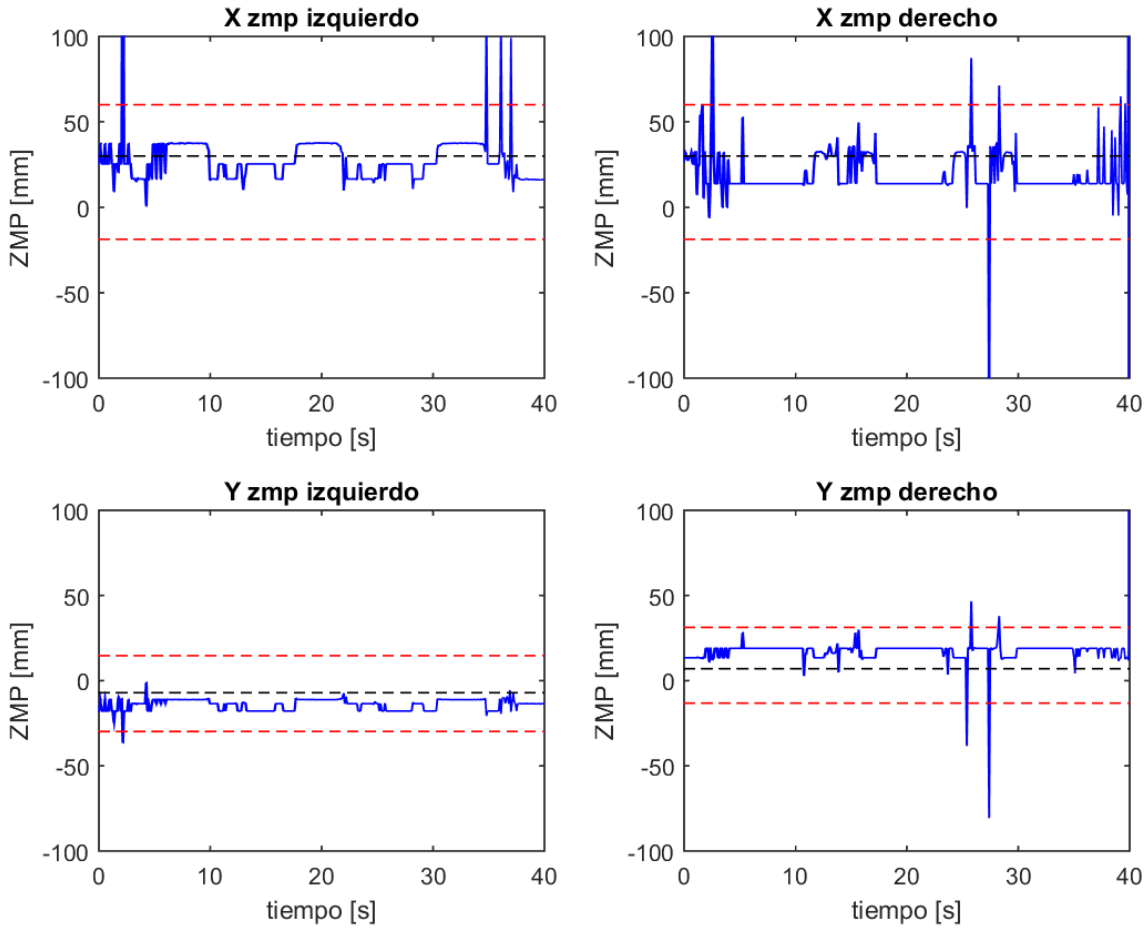


Figura 6.9: ZMP durante la marcha del robot bípedo

# Capítulo 7

# Conclusiones y trabajo a futuro

## 7.1 Conclusiones

El sistema embebido para el robot bípedo mejoró significativamente el procesamiento de datos, permitiendo la implementación de un controlador para la marcha y postura. El sistema embebido servirá para futuros controladores, dado que la unidad de procesamiento (Raspberry) permite una amplia gama de módulos. La separación de procesamiento entre tarjetas de adquisición y driver de los servomotores, permitió un procesamiento eficiente de la información adquirida ya sea por los sensores de fuerza o potenciómetros del robot o la aplicación de las leyes de control para el robot bípedo. Este mismo sistema puede ser extrapolado a cualquier otro robot. En general, el sistema embebido demostró tener un mejor desempeño que el sistema utilizado en trabajos previos: [29] y [30].

La interfaz gráfica permite una selección entre tres diferentes estrategias de control de manera visual, permitiendo al usuario probar al menos tres estrategias; sin embargo, ninguna de ellas se puede alterar de manera directa con la interfaz gráfica.

A través de la experimentación se comparó un controlador PID tradicional y el controlador PID-ASLD diseñado en el Capítulo 4. Éste último demostró tener un mejor desempeño en el tiempo de asentamiento para la postura del robot bípedo. El PID-ASLD tuvo robustez ante cambios de pendiente en contraste con el control PID tradicional.

El control de postura (CDP + PID-ASLD) resultó tener una gran resistencia a los cambios de pendiente, permitiendo que el robot se llegara a inclinar hasta $20^\circ$ en dirección Pitch y $10^\circ$ en dirección Roll.

El control de marcha (PID-ASLD) basado en el control de postura, estableció que la modificación de las trayectorias previamente calculadas para el robot bípedo dan lugar a una marcha estable. La modificación de las trayectorias no destruye su forma o configuración inicial, sino más bien desplaza dicho comportamiento. La estabilidad de la locomoción bípeda durante la aplicación del control de marcha fue probada por medio del criterio del ZMP; por éste se determinó que la marcha es estable.

## 7.2 Trabajo a futuro

### 7.2.1 Generación de trayectorias basadas en inteligencia artificial (Redes-neuronales, lógica difusa, máquinas de aprendizaje)

La generación de trayectorias para el robot bípedo podría adoptarse desde una perspectiva inteligente, pudiendo pensar en la implementación de un generador interno que adapte las trayectorias según el caso. Si se pudiera diseñar un generador lo suficientemente rápido, sería complemento ideal de la estrategia de control de marcha. La Figura 7.1, muestra una secuencia de poses utilizadas para el entrenamiento de un robo bípedo con la finalidad de obtener un generador trayectorias adaptables al terreno utilizando maquinas de aprendizaje (Deep Learning) [19]. El mejoramiento de la generación de trayectorias es un punto clave para nuevas estrategias de control.

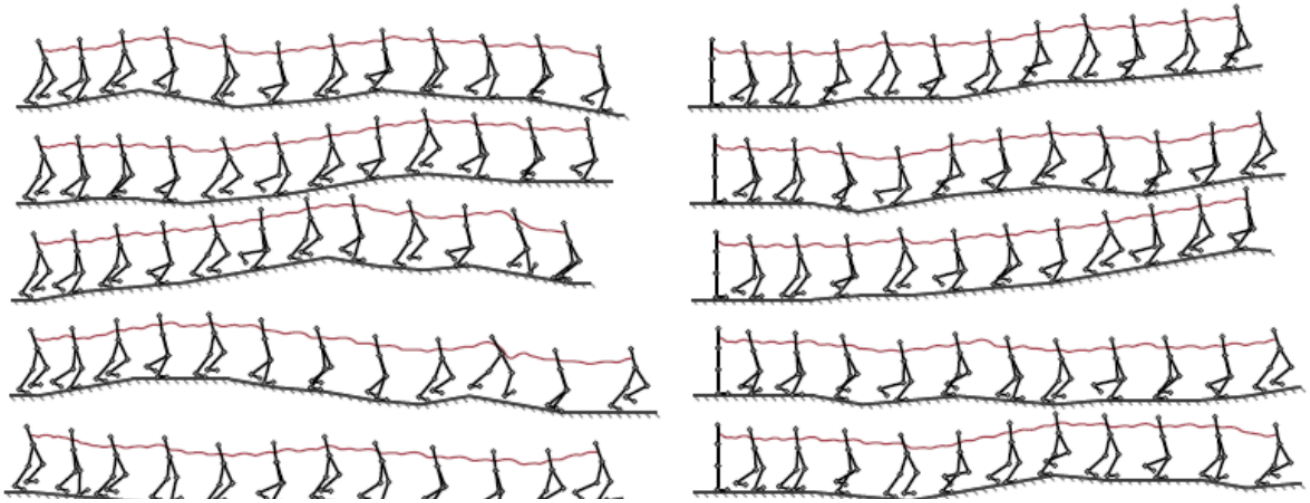

Figura 7.1: Diagrama de bloques propuesto para el robot bípedo Scout

En 2004 **Kuo-Yang Tu y Tsu-Tian Lee** presentaron *DESIGN OF A MULTI-LAYER FUZZY LOGIC CONTROLLER USING POLE ASSIGNMENT FOR BIPEDAL WALKING AT VARYING SPEEDS* [46], donde desarrollaron un controlador difuso multi-capa (MLFLC, Multi-Layer Fuzzy Logic Controller ) capaz de variar la velocidad de caminata, mediante la modificación de la distancia entre cada paso o zancada que da el robot bípedo. Se concluyó que MLFLC genera una variación de velocidad estable y aplicable a la evasión de obstáculos de un robot bípedo. Se propone seguir esta linea de investigación y diseñar un controlador multi-capa que le permita al robot bípedo Scout tener una marcha ajustable.

### 7.2.2 Unificación de controladores

Para obtener un controlador completo del robot bípedo Scout, se debe unificar los controladores basados en el ZMP y el CoM, además se puede utilizar un magnetómetro para el control de dirección y evitar que el robot camine de manera errónea en el caso de una linea recta. En la Figura 7.2 se muestra la unificación propuesta de los controladores.

El control orientado al ZMP tendría como objetivo la adaptación de la pisada al cambio de pendiente o cambio irregular del suelo. El control de el centro de masa conservará siempre una postura y caminata estable en dirección Pitch y Roll. Por último el control de dirección está pensado para rectificar el sentido de caminata del robot, en el caso de que el robot caminara en dirección errónea.

Además si la generación de trayectorias fuera adaptable como se propone en la sección 7.2.1, la marcha se puede hacer aún más robusta. Existe otro tipo de controladores orientados al control de vibraciones o al control de par. Estos controladores serían de difícil implementación en el robot bípedo Scout, por que los servomotores no puede ser controlados directamente mediante corriente.

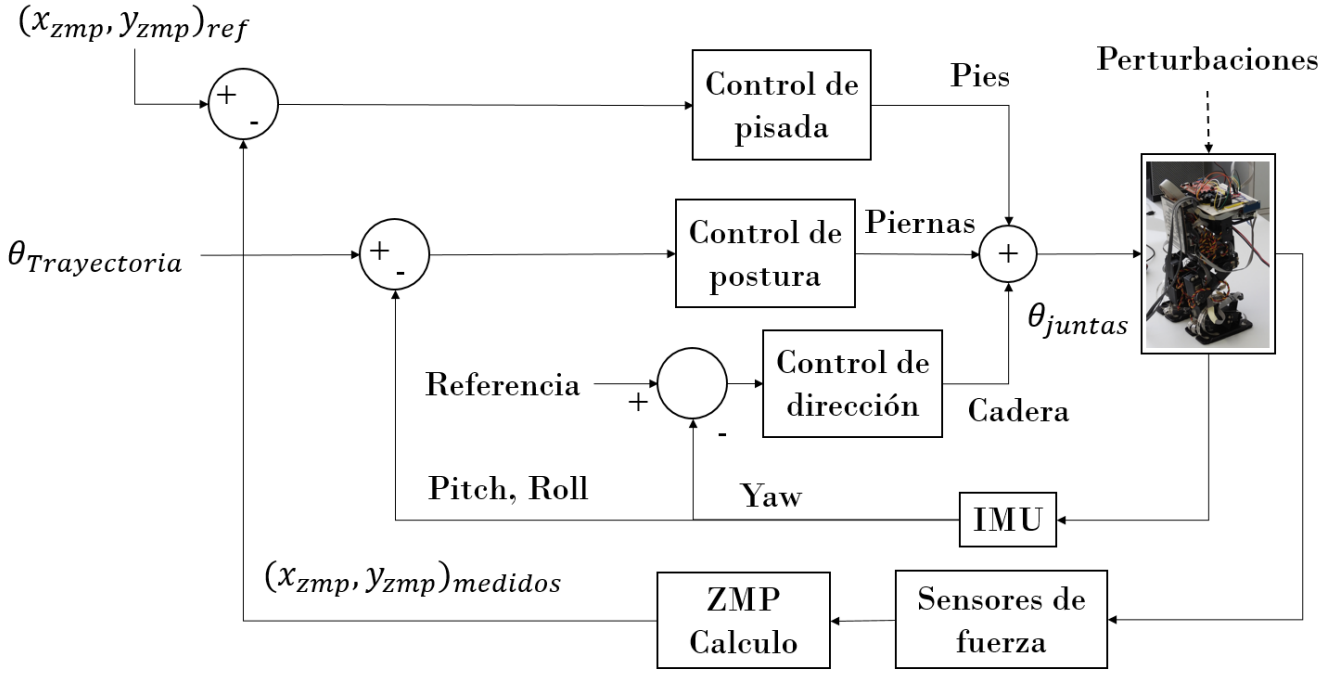


Figura 7.2: Diagrama de bloques propuesto para el robot bípedo Scout

# Apéndice A

# Operaciones entre conjuntos difusos

Al igual que en la teoría clásica de conjuntos, existen dos principales operaciones que serán utilizadas para los conjuntos difusos: la unión y la intersección. Estas dos operaciones serán la base de la inferencia lógica; es decir, la toma de decisiones de los sistemas difusos [53] [18].

## A.1 Unión

**Conjuntos Clásicos**

Sea un conjunto A y un conjunto B, ambos contenidos en un universo $U$. La unión se define como $C = A \cup B$, dónde C será el conjunto resultante de la operación. En la Figura A.1 a), se puede observar gráficamente ésta operación.

**Conjuntos Difusos**

En los conjuntos difusos, la unión fue establecida por Zadeh en [51]. Dado un conjunto A y un conjunto B, cada uno definido por su función de membresia, se puede hacer la unión mediante el máximo de los valores entre cada una de las funciones de membresía. Matemáticamente se define como: $\mu_C(x) = \mu_A(x) \vee \mu_B = max(\mu_A(x), \mu_B(x))$, esta operación se muestra en la Figura A.2 a).

## A.2 Intersección

**Conjuntos Clásicos**

Sea un conjunto A y un conjunto B, ambos contenidos en un universo $U$. La intersección se define como $C = A \cap B$, dónde C será el conjunto resultante de la operación. En la Figura A.1 b), se observa gráficamente la intersección.

**Conjuntos Difusos**

Al igual que en la operación unión, la intersección fue elaborada por Zadeh en [51].Dado un conjunto A y

un conjunto B, cada uno definido por su función de membresia, se puede hacer la intersección mediante el mínimo de los valores entre cada una de las funciones de membresía. Matemáticamente se define como: $\mu_C(x) = \mu_A(x) \wedge \mu_B = min(\mu_A(x), \mu_B(x))$, esta operación se muestra en la Figura A.2 b).

## A.3 Complemento

**Conjuntos Clásicos**

Dado un universo $U$ y un conjunto A, el complemento se define como todos aquellos valores que no pertenecen al conjunto A. $\neg A = \{x \mid x \notin A\}$. En la Figura A.1 c) , se observa el complemento de un conjunto clásico.

**Conjuntos Difusos**

Dado un conjunto difuso A, con un universo de discurso X, y una función de pertenencia $\mu_A(x)$, su complemento se define como: $\mu_{\neg A}(x) = 1 - \mu_A(x)$, $\neg A = \{x, \mu_{\neg A}(x)\}$. En la Figura A.3, se observa el complemento de un conjunto difuso.

En conclusión, se observa una similitud con las operaciones de conjuntos, ya que la teoría de conjuntos difusos tiende a ser la generalización de los conjunto clásicos o nítidos.

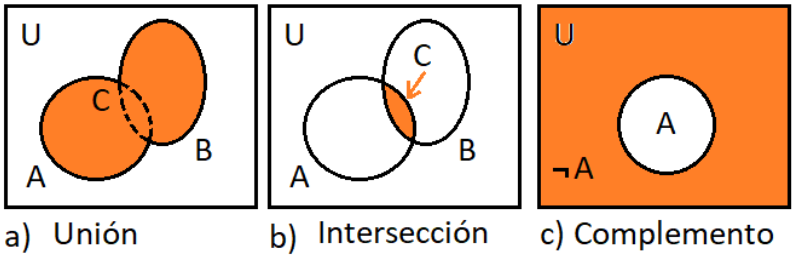


Figura A.1: Operaciones básicas entre conjunto clásicos: a) unión, b) intersección y c) complemento.

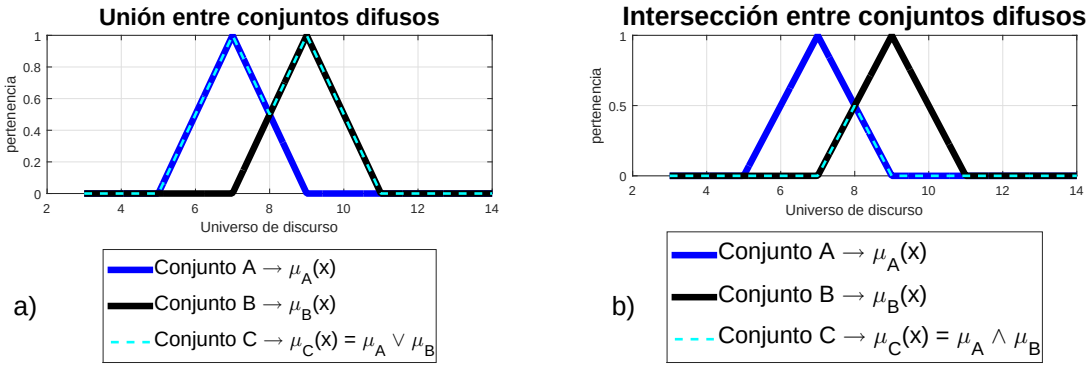


Figura A.2: a) Unión entre conjuntos difusos, b) intersección entre conjuntos difusos

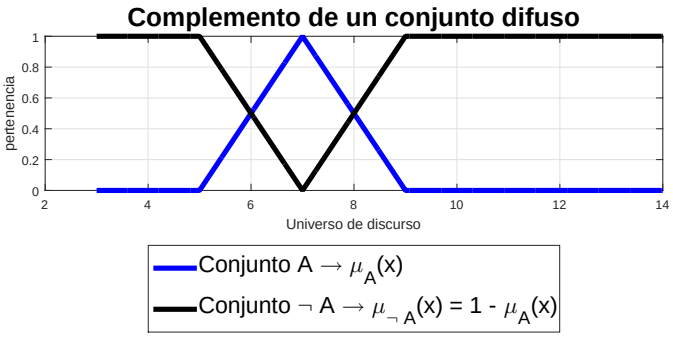


Figura A.3: Complemento de los conjuntos difusos

## A.4 Propiedades de las operaciones conjuntos difusos

### Conjuntos Clásicos

En la Tabla A.1, se muestran las propiedades básicas de las operaciones, unión e intersección, dado un conjunto A, perteneciente a un universo $U$ para los conjuntos clásicos o nítidos.

Tabla A.1: Propiedades de las operaciones en los conjuntos clásicos[18]

| **Propiedad** | **Unión** | **Intersección** |
|---|---|---|
| Conmutativa | $A \cup B = B \cup A$ | $A \cap B = B \cap A$ |
| Asociativa | $A \cup (B \cup C) = (A \cup B) \cup C$ | $A \cap (B \cap C) = (A \cap B) \cap C$ |
| Distributiva | $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ |
| Identidad | $A \cup \emptyset = A$ | $A \cap \emptyset = \emptyset$ |
| | $A \cup U = U$ | $A \cap U = A$ |
| Transitiva | si $A \subseteq B$ y $B \subseteq C$ entonces $A \subseteq C$ | |
| Idempotencia | $A \cup A = A$ | $A \cap A = A$ |
| Complementariedad | $A \cup \neg A = U$ | $A \cap \neg A = \emptyset$ |
| Involutiva | $\neg(\neg A) = A$ | |

### Conjuntos Difusos

Los conjuntos difusos cumplen, en su mayoría, con las propiedades de las operaciones de los conjuntos clásicos; sin embargo, muestra una seria diferencia en la propiedad de complementariedad, dónde se comprueba que en los conjuntos difusos no es posible esta propiedad, debido al grado de pertenencia cambiante de las funciones de membresía. Enseguida, en la Tabla A.2, se muestran las propiedades de las operaciones para los conjuntos difusos .

Tabla A.2: Propiedades de las operaciones en los conjuntos difusos[18]

| **Propiedad** | **Unión** | **Intersección** |
|---|---|---|
| Conmutativa | $\mu_A \vee \mu_B = \mu_B \vee \mu_A$ | $\mu_A \wedge \mu_B = \mu_B \wedge \mu_A$ |
| Asociativa | $\mu_A \vee (\mu_B \vee \mu_C) = (\mu_A \vee \mu_B) \vee \mu_C$ | $\mu_A \wedge (\mu_B \wedge \mu_C) = (\mu_A \wedge \mu_B) \wedge \mu_C$ |
| Distributiva | $\mu_A \vee (\mu_B \wedge \mu_C) = (\mu_A \vee \mu_B) \wedge (\mu_A \vee \mu_C)$ | $\mu_A \wedge (\mu_B \vee \mu_C) = (\mu_A \wedge \mu_B) \vee (\mu_A \wedge \mu_C)$ |
| Identidad | $\mu_A \vee \emptyset = \mu_A$ | $\mu_A \wedge \emptyset = \mu_A$ |
| | $\mu_A \vee X = X$ | $\mu_A \wedge X = \mu_A$ |
| Transitiva | si $\mu_A \subseteq \mu_B$, y $\mu_B \subseteq \mu_C$ entonces $\mu_A \subseteq \mu_C$ | |
| Idempotencia | $\mu_A \vee \mu_A = \mu_A$ | $\mu_A \wedge \mu_A = \mu_A$ |
| Complementariedad | $\mu_A \vee \neg\mu_A \neq X$ (no se cumple) | $\mu_A \wedge \neg\mu_A \neq \emptyset$ (no se cumple) |
| Involutiva | $\neg(\neg\mu_A) = \mu_A$ | |

# Apéndice B

# Relación entre conjuntos

### Conjuntos Clásicos

Una relación es una correspondencia de un elemento contenido en un conjunto X a otro elemento de un segundo conjunto Y. En la teoría clásica de conjuntos se llaman a las relaciones entre dos conjuntos, producto cartesiano, teniendo una forma: $\mathcal{R} = X \times Y = \{(x, y) \mid x \in X, y \in Y\}$. Al conjunto X se le conoce usualmente como dominio, y al conjunto Y co-dominio o imagen. El producto cartesiano permite ordenar en pares, elemento del conjunto X, junto un elemento del conjunto Y. Una de las relaciones mas utilizadas es la función. La función es aquella relación que considera un comportamiento unívoco; es decir, que a cada elemento del dominio sólo corresponda uno del co-dominio o imagen [18].

Sea un conjunto $A = \{1, 2, 3, 4, 5\}$ y un conjunto $B = \{v, w, x, y, z\}$, su producto cartesiano quedará definido como:(Ecuación B.1, TablaB.1).

La Tabla B.1 representa a la relación $A \times B$ (producto cartesiano), también muestra la pertenencia de cada uno de los pares ordenados. Como se sabe la pertenencia en los conjuntos clásicos solo puede oscilar entre 0 y 1.

$$\begin{aligned}\mathcal{R}_{\mathcal{AB}} = A \times B = \{&(1, v), (1, w), (1, x), (1, y), (1, z),\\ &(2, v), (2, w), (2, x), (2, y), (2, z),\\ &(3, v), (3, w), (3, x), (3, y), (3, z),\\ &(4, v), (4, w), (4, x), (4, y), (4, z),\\ &(5, v), (5, w), (5, x), (5, y), (5, z)\}\end{aligned} \tag{B.1}$$

Tabla B.1: Producto cartesiano $\mathcal{R}_{\mathcal{AB}} = A \times B$

| **Conjunto B** Imagen | | | | | |
|---|---|---|---|---|---|
| z | 1 | 1 | 1 | 1 | 1 |
| y | 1 | 1 | 1 | 1 | 1 |
| x | 1 | 1 | 1 | 1 | 1 |
| w | 1 | 1 | 1 | 1 | 1 |
| v | 1 | 1 | 1 | 1 | 1 |
| | 1 | 2 | 3 | 4 | 5 |
| | **Conjunto A** Dominio | | | | |

**Conjuntos Difusos**

Una relación difusa es similar a una relación clásica o nítida, la diferencia consiste en el grado de pertenencia, el cual varía en los conjuntos difusos, Zadeh([51]) definió una relación difusa como:

$$\mathcal{R} = \{\frac{\mu_R(x,y)}{(x,y)} \mid (x,y) \in X \times Y\} = min(\mu_X(x), \mu_Y(y)) = \mu_X(x) \wedge \mu_Y(y) \tag{B.2}$$

Es importante recordar la notación previamente definida, es decir el simbolo de fracción fungira simplemente para hacer relación (en éste caso) entre valores del universo de discurso y el grado de pertenencia que tienen.

Una relación difusa puede ser visualizada como una superficie, donde el eje X y el eje Y, se ven sustituidos por el universo de discurso X y el universo de discurso Y, respectivamente, y el eje Z por la pertenencia resultante a la relación.

Dado un conjunto $A = \{\frac{0.1}{1} + \frac{0.4}{2} + \frac{0.5}{3} + \frac{0.5}{4} + \frac{0.8}{5}\}$ y un conjunto $B = \{\frac{0.5}{1} + \frac{0.2}{2} + \frac{0.7}{3} + \frac{0.7}{4} + \frac{0.1}{5}\}$, donde el conjunto A tiene un universo de discurso $x \in [1,5]$, y el conjunto B tiene un universo de discurso $y \in [1,5]$, se obtiene $\mathcal{R}_{\mathcal{AB}} = A \times B$. En la Tabla B.2 y en las ecuaciones B.3 y B.4, se muestra el resultado.

Ejemplificando algunos casos, se tiene que:

Para $x = 1$ y $y = 1$:

$\mu_R(1,1) = min(\mu_A(1), \mu_B(1)) = min(0.1, 0.5) = 0.1$

Para $x = 4$ y $y = 3$:

$\mu_R(4,3) = min(\mu_A(4), \mu_B(3)) = min(0.5, 0.7) = 0.5$

Para $x = 2$ y $y = 5$:

$\mu_R(2,5) = min(\mu_A(2), \mu_B(5)) = min(0.2, 0.1) = 0.1$

$$\begin{aligned}\mathcal{R}_{\mathcal{AB}} = A \times B = \{&\mu_{\mathcal{R}_{\mathcal{AB}}}(1,1), \mu_{\mathcal{R}_{\mathcal{AB}}}(1,2), \mu_{\mathcal{R}_{\mathcal{AB}}}(1,3), \mu_{\mathcal{R}_{\mathcal{AB}}}(1,4), \mu_{\mathcal{R}_{\mathcal{AB}}}(1,5),\\ &\mu_{\mathcal{R}_{\mathcal{AB}}}(2,1), \mu_{\mathcal{R}_{\mathcal{AB}}}(2,2), \mu_{\mathcal{R}_{\mathcal{AB}}}(2,3), \mu_{\mathcal{R}_{\mathcal{AB}}}(2,4), \mu_{\mathcal{R}_{\mathcal{AB}}}(2,5),\\ &\mu_{\mathcal{R}_{\mathcal{AB}}}(3,1), \mu_{\mathcal{R}_{\mathcal{AB}}}(3,2), \mu_{\mathcal{R}_{\mathcal{AB}}}(3,3), \mu_{\mathcal{R}_{\mathcal{AB}}}(3,4), \mu_{\mathcal{R}_{\mathcal{AB}}}(3,5),\\ &\mu_{\mathcal{R}_{\mathcal{AB}}}(4,1), \mu_{\mathcal{R}_{\mathcal{AB}}}(4,2), \mu_{\mathcal{R}_{\mathcal{AB}}}(4,3), \mu_{\mathcal{R}_{\mathcal{AB}}}(4,4), \mu_{\mathcal{R}_{\mathcal{AB}}}(4,5),\\ &\mu_{\mathcal{R}_{\mathcal{AB}}}(5,1), \mu_{\mathcal{R}_{\mathcal{AB}}}(5,2), \mu_{\mathcal{R}_{\mathcal{AB}}}(5,3), \mu_{\mathcal{R}_{\mathcal{AB}}}(5,4), \mu_{\mathcal{R}_{\mathcal{AB}}}(5,5)\}\end{aligned} \tag{B.3}$$

$$\begin{aligned}\mathcal{R}_{\mathcal{AB}} = A \times B = \{&\mu_A(1) \wedge \mu_B(1), \mu_A(1) \wedge \mu_B(2), \mu_A(1) \wedge \mu_B(3), \mu_A(1) \wedge \mu_B(4), \mu_A(1) \wedge \mu_B(5),\\ &\mu_A(2) \wedge \mu_B(1), \mu_A(2) \wedge \mu_B(2), \mu_A(2) \wedge \mu_B(3), \mu_A(2) \wedge \mu_B(4), \mu_A(2) \wedge \mu_B(5),\\ &\mu_A(3) \wedge \mu_B(1), \mu_A(3) \wedge \mu_B(2), \mu_A(3) \wedge \mu_B(3), \mu_A(3) \wedge \mu_B(4), \mu_A(3) \wedge \mu_B(5),\\ &\mu_A(4) \wedge \mu_B(1), \mu_A(4) \wedge \mu_B(2), \mu_A(4) \wedge \mu_B(3), \mu_A(4) \wedge \mu_B(4), \mu_A(4) \wedge \mu_B(5),\\ &\mu_A(5) \wedge \mu_B(1), \mu_A(5) \wedge \mu_B(2), \mu_A(5) \wedge \mu_B(3), \mu_A(5) \wedge \mu_B(4), \mu_A(5) \wedge \mu_B(5)\}\end{aligned} \tag{B.4}$$

Tabla B.2: Producto cartesiano $\mathcal{R}_{\mathcal{AB}} = A \times B$

| Imagen **Conjunto B** | | | | | |
|---|---|---|---|---|---|
| 5 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 4 | 0.1 | 0.4 | 0.5 | 0.5 | 0.7 |
| 3 | 0.1 | 0.4 | 0.5 | 0.5 | 0.7 |
| 2 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 |
| 1 | 0.1 | 0.4 | 0.5 | 0.5 | 0.5 |
| | 1 | 2 | 3 | 4 | 5 |
| | | | | | **Conjunto A** Dominio |

Cabe mencionar que una relación hereda las operaciones y propiedades de los conjuntos difusos, es decir, la unión, la intersección y el complemento con sus distintas propiedades.

## B.1 Composición de relaciones

Es posible hacer composición de relaciones, ya que se cumple con distributividad y asociatividad al haber heredado todas las propiedades de los conjuntos difusos. Teniendo dos relaciones: una R y otra S, definidas en los conjuntos A,B y C, donde $\mathcal{R} \subseteq A \times B \rightarrow (x, y)$ y $\mathcal{S} \subseteq B \times C \rightarrow (y, z)$.

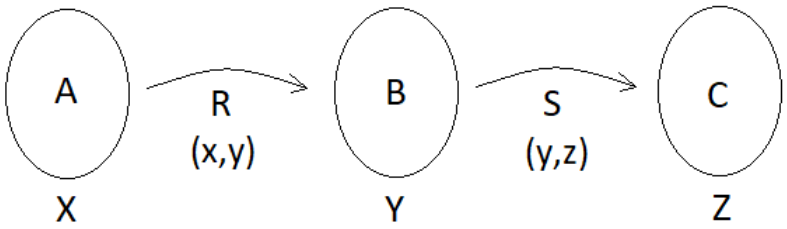


Figura B.1: Representación esquemática de la composición en conjuntos

Existen dos tipos de composiciones: la composición **max-min** y la composición **max-producto**. El uso de dichas composiciones estará delimitado por el sistema difuso en cuestión. Cabe mencionar que las composiciones son aplicables a los conjuntos clásicos y difusos, debido a que los conjuntos clásicos son un caso especial de los conjuntos difusos. En el caso del presente trabajo la composición **max-min** tomará importancia en el diseño de los controladores difusos, se define como:

$$\mathcal{T} = \bigvee_{y} \{\mu_{\mathcal{R}}(x, y) \wedge \mu_{\mathcal{S}}(y, z)\} = \mathcal{R} \circ \mathcal{S} \tag{B.5}$$

# Apéndice C

# Operadores difusos generalizados

La generalización de las operaciones realizadas en primera instancia por Zadeh en [51], permitió la elaboración de una lógica difusa más compleja a través de operadores.

Existen dos principales operadores difusos, también conocidos *normas triangulares (normas - T) y normas - S (co-normas -T)*. Básicamente son relaciones que mapean de un conjunto difuso a otro conjunto difuso con cierta operación intermedia, esta operación intermedia es definida por el tipo de norma, la norma-T es exclusiva de la intersección y la norma-S, de la unión. Las normas-S, son las normas duales de las normas-T, por lo que sus operaciones son inversas. En resumen las normas-T y S, son generalizaciones de la intersección y la unión, respectivamente.

La norma-T, se define como:

$$T : [0,1] \times [0,1] \rightarrow [0,1] \tag{C.1}$$

A través de un operador binario $\triangle$, la norma-T, se define como:

$$\mu_{A \wedge B}(x,y) = T(\mu_A(x), \mu_B(y)) = \mu_A(x) \triangle \mu_B(y) \tag{C.2}$$

## C.1 Normas-T

Los tipos de normas-T, más importantes son:

- Mínimo: $T_{min} = min(x,y)$
- Producto:$T_p = x * y$
- Lukasiewicz: $T_L = max(0, x + y - 1)$

- Producto drástico: $T_{pd} = \begin{cases} x & si, \quad y = 1 \\ y & si, \quad x = 1 \\ 0 & otro \quad caso \end{cases}$

La norma-S, se define como:

$$S : [0,1] \times [0,1] \rightarrow [0,1] \tag{C.3}$$

## C.2 Normas-S

Los tipos de normas-S, más importantes son:

- Máximo: $S_{max} = max(x, y)$
- Suma algebraica:$S_{sa} = x + y - xy$
- Lukasiewicz: $S_L = min(0, x + y - 1)$
- Suma drástica:$S_{sd} = \begin{cases} x & si, \quad y = 0 \\ y & si, \quad x = 0 \\ 1 & otro \quad caso \end{cases}$

## C.3 Propiedades

Las normas-T y S, cumplen con las siguientes propiedades:

Tabla C.1: Propiedades de las normas - T, y normas - S

| **Propiedad** | **Norma-T** | **Norma-S** |
|---|---|---|
| Acotada | $T(0,0) = 0$ | $S(1,1) = 1$ |
| Monotónica | $T(a,b) \leq T(c,d)$ | $S(a,b) \geq S(c,d)$ |
| | si, $a \leq c$ y $b \leq d$ | si, $a \geq c$ y $b \geq d$ |
| Conmutativa | $T(a,b) = T(b,a)$ | $S(a,b) = S(b,a)$ |
| Asociativa | $T(a,T(b,c)) = T(T(a,b),c)$ | $S(a,S(b,c)) = S(S(a,b),c)$ |

Estas normas toman importancia en la inferencia lógica de un sistema difuso ya que pueden modificar el comportamiento del sistema, haciéndolo susceptible a ciertos valores de entrada. En conclusión dependiendo de la norma -T o norma -S , que se elija, se definirá la sensibilidad del sistema difuso diseñado, en el presente trabajo se usaron las normas: $T_{min}$ y $S_{max}$.

# Apéndice D

# Diseño de un controlador difuso tipo mamdani

Con el fin de ejemplificar cada una de las partes de un controlador difuso de tipo Mamdani, se utilizara el siguiente caso: se asume que se requiere controlar la posición de un motor para esto, la posición real de su flecha es medida y comparada mediante un potenciómetro y un amplificador operacional diferencial, respectivamente, obteniendo así el error de posición. Se considera también que un voltaje negativo aplicado al motor derivará en un sentido de giro anti-horario, mientras que un voltaje positivo en un sentido horario.

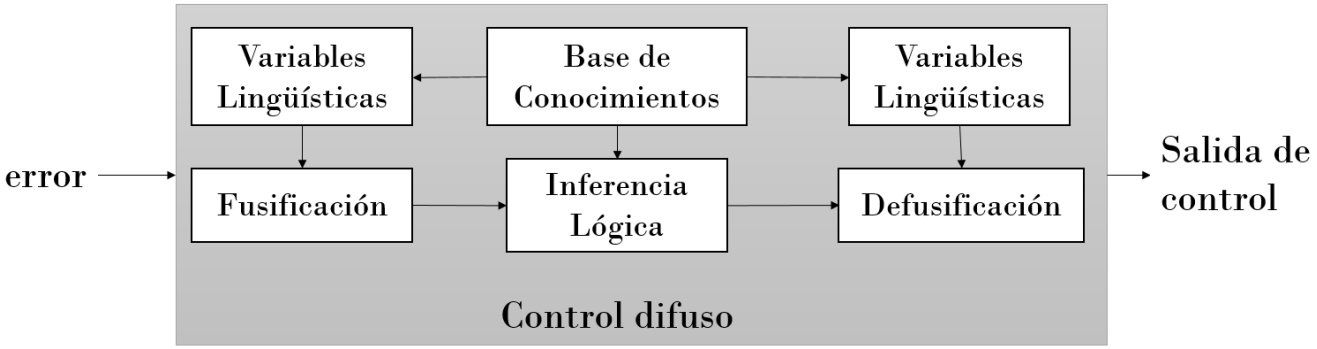


Figura D.1: Esquema conceptual interno de un controlador difuso de tipo Mamdani

## D.1 Variables lingüísticas

Para un controlador difuso de tipo Mamdani, la correcta asignación de las variables lingüísticas es esencial para la correcta fusificación del o de los datos de entrada. Ya que está basado en el comportamiento del error; el error como tal es la variable lingüística de entrada por defecto. Sin embargo, pueden existir variantes según el controlador, por ejemplo, el cambio del error o la suma del error. Por otro lado se tienen las variables lingüísticas para la salida de control, estas variables deben ser asignadas según la planta. Por

ejemplo en el caso del motor, se le podrá controlar mediante el voltaje por lo que su variable lingüística de salida será el voltaje por defecto.

Considerando el caso de estudio del control de posición de un motor, las variables lingüísticas de entrada y salida pueden ser asignadas como:

**Entrada - Error** (v,X,$T_v$,M)

**v:** Error

**X:** Suponiendo que el motor solo se puede mover 90 grados, el error puede ser considerado como:

$[-90°, 90°]$

$T_v = \{$Negativo Grande, Negativo Pequeño, Cero, Positivo Pequeño, Positivo Grande $\}$

$T_v = \{$NG,NP,CE,PP,PG$\}$

$M(NG) = trapmf([-90, -80, -60, -30])$

$M(NP) = trimf([-60, -30, 0)$

$M(CE) = trimf([-30, 0, 30])$

$M(PP) = trimf([0, 30, 60])$

$M(PG) = trapmf([30, 60, 80, 90])$

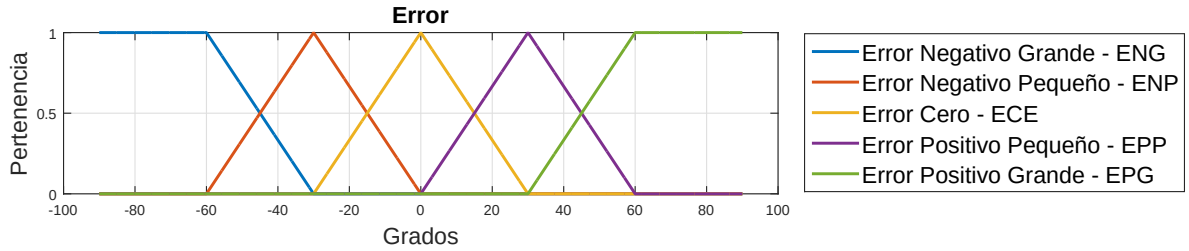


Figura D.2: Variable lingüística de entrada: Error

**Salida - Voltaje** (v,X,$T_v$,M)

**v:** Voltaje

**X:** Suponiendo que el motor solo puede soportar hasta 15 o -15 [v], el intervalo de operación queda como: [-15,15] volts

$T_v = \{$Negativo Grande,Negativo Pequeño,Cero,Positivo Pequeño,Positivo Grande $\}$

$T_v = \{$NG,NP,CE,PP,PG$\}$

$M(NG) = trapmf([-15, -15, -10, -5])$

$M(NP) = trimf([-10, -5, 0)$

$M(CE) = trimf([-5, 0, 5])$

$M(PP) = trimf([0, 5, 10])$

$M(PG) = trapmf([5, 10, 15, 15])$

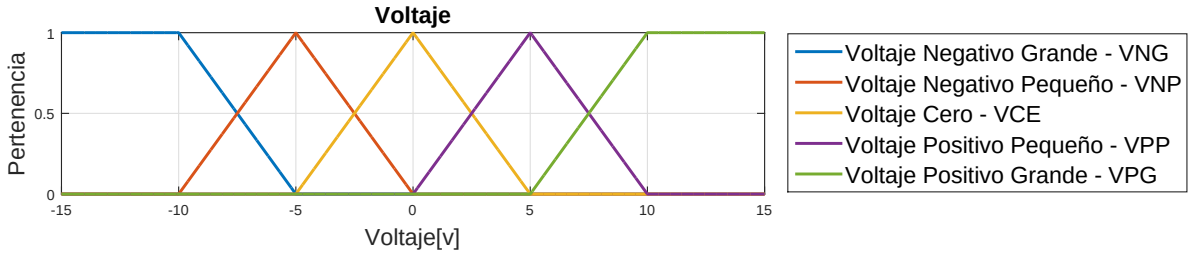


Figura D.3: Variable lingüística de salida: Voltaje

## D.2 Interfaz de fusificación

La interfaz de fusificación es la encargada de trasladar valores reales, nítidos o exactos e interpretarlos de manera difusa. Este proceso se hace considerando al dato de entrada como un conjunto con una función de membresía de tipo singleton, ver Sección 1.1.2. La intersección entre el conjunto(s) entre el valor de entrada y el valor lingüístico en cuestión dentro del controlador darán como resultado la fusificación inmediata del dato real para su posterior procesamiento dentro del controlador. Esto se puede ejemplificar mediante la Figura D.4, donde un valor real de entrada a la variable linguistica del error, es transformado a un conjunto singleton, lo cual ocasiona la intersección con dos conjuntos difusos: Error Cero y Error Pequeño Positivo. Los dos valores de la intersección, son la pertenencia a cada uno de esos conjuntos, esto inmediatamente repercute en la inferencia lógica cuando se genera la conclusión de cada regla.

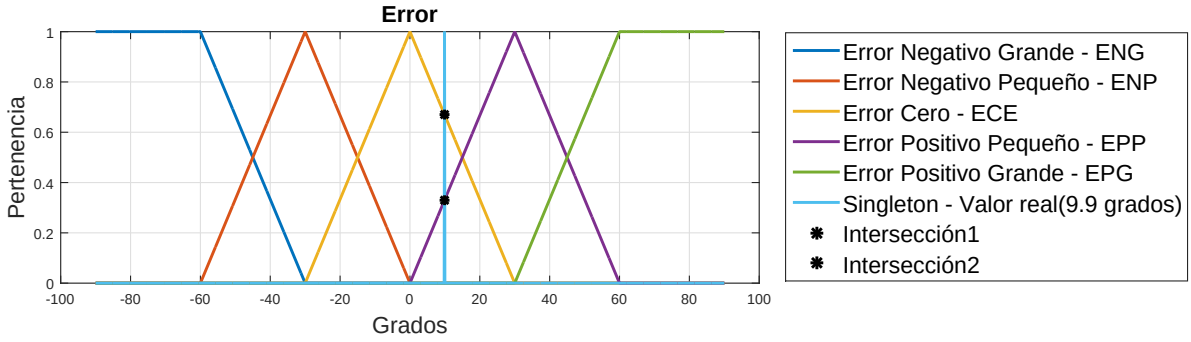


Figura D.4: Fusificación

## D.3 Base de conocimientos

La base de conocimientos es la parte más importante para el comportamiento del controlador difuso. Dado que este controlador se basa en el comportamiento del error, se debe exponer el conocimiento del

agente experto. El agente experto debe ser capaz de describir la relación entrada-salida y ser minucioso en las reglas de control. Las reglas de control normalmente pueden ser visualizadas mediante tablas de verdad, como se muestra en la tabla D.1.

En la tabla D.1, se muestra la relación entrada-salida que debe haber para controlar la posición de un motor.

Cuando el error sea Negativo Grande, la acción de control deberá ser un voltaje Positivo Grande y dado un error Negativo Pequeño, se le corresponderá con un voltaje Positivo Pequeño; suponiendo un error cercano a Cero, entonces se tendrá un voltaje cercano a Cero y así sucesivamente se construyen las reglas de control para la planta en cuestión. La relación entrada-salida puede ser mucho más compleja según el sistema a controlar.

Tabla D.1: Tabla de verdad Implicación difusa

| **Error** | Correspondencia | **Voltaje** |
|---|---|---|
| ENG | $\Longrightarrow$ | VPG |
| ENP | $\Longrightarrow$ | VPP |
| ECE | $\Longrightarrow$ | VCE |
| EPP | $\Longrightarrow$ | VNP |
| EPG | $\Longrightarrow$ | VNG |

## D.4 Inferencia lógica

La inferencia del controlador difuso tipo Mamdani se hace generalizando el Modus ponens difuso. A continuación se presenta el Modus ponens usado en la Sección 2.5.3, con el fin de contrastar la configuración con el Modus ponens extendido para el controlador difuso.

Regla - premisa 1 **Si** $x$ es $A$, **entonces** $y$ es $B$

Hecho - premisa 2 $x$ es $A'$

---

Conclusión $\therefore y$ es $B'$

La generalización considera que cada uno de los Modus ponens dará como resultado una conclusión para cada regla, por lo que se utiliza la unión entre conclusiones para obtener una **conclusión general**, tal y como se muestra a continuación:

Hecho - premisa 1 $x$ es $A'$

Regla1 - premisa 2 **Si** $x$ es $A_1$, **entonces** $y$ es $B_1$ Conclusión $\therefore y$ es $B'_1$
Regla2 - premisa 3 **Si** $x$ es $A_2$, **entonces** $y$ es $B_2$ Conclusión $\therefore y$ es $B'_2$
Regla3 - premisa 4 **Si** $x$ es $A_3$, **entonces** $y$ es $B_3$ Conclusión $\therefore y$ es $B'_3$
$\vdots$ $\vdots$ $\vdots$ $\vdots$
Regla i - premisa i+1 **Si** $x$ es $A_i$, **entonces** $y$ es $B_i$ Conclusión $\therefore y$ es $B'_i$

---

**Conclusión general** $\therefore y$ es $B' = B'_1 \cup B'_2 \cup B'_3 ... \cup B'_i$

Matemáticamente se puede reescribir como se muestra enseguida:

Hecho - premisa 2 $x_0 \implies \mu_{A'}(x_0)$
Regla - premisa 1 $\mu_{B'_1}(y) = \mu_{A_1}(x_0) \wedge \mu_B(y) = T_{min}(\mu_{A_1}(x_0), \mu_{B_1}(y))$
Regla - premisa 1 $\mu_{B'_2}(y) = \mu_{A_2}(x_0) \wedge \mu_B(y) = T_{min}(\mu_{A_2}(x_0), \mu_{B_2}(y))$
Regla - premisa 1 $\mu_{B'_3}(y) = \mu_{A_3}(x_0) \wedge \mu_B(y) = T_{min}(\mu_{A_3}(x_0), \mu_{B_3}(y))$
$\vdots$ $\vdots$ $\vdots$
Regla i - premisa i+1 $\mu_{B'_i}(y) = \mu_{A_i}(x_0) \wedge \mu_B(y) = T_{min}(\mu_{A_i}(x_0), \mu_{B_i}(y))$

---

**Conclusión general** $\therefore \mu_{B'}(y) = \mu_{B'_1}(y) \vee \mu_{B'_2}(y) \vee \mu_{B'_3}(y), ... \vee \mu_{B'_i}(y)$

Siguiendo el caso de estudio del control de posición del motor. Cuando se fusifique el valor real $9.9°$ de error obtenido en el ejemplo, se observará que este intersecta con dos conjuntos difusos, en mayor o menor medida cada uno le otorgará un valor de pertenencia, posteriormente se procede a interpretar la base de conocimientos, utilizando reglas de inferencia, utilizando el modus ponens difuso, en este caso particular se tiene que el conjunto ECE, está asociado a VCE, por lo que en el punto de intersección se hace un corte. De igual forma para el conjunto EPP, asociado al conjunto de salida VNP.

## D.5 Interfaz de defusificación

La interfaz de defusificación consiste en calcular la salida de control para el controlador difuso, es decir el resultado después de haber hecho la inferencia lógica o toma de decisiones.

### D.5.1 Agregación

Se le llama agregación a la unión entre conclusiones provenientes de las inferencias lógicas, podría considerarse una suma de conclusiones con base en la toma de decisiones. La agregación genera un función

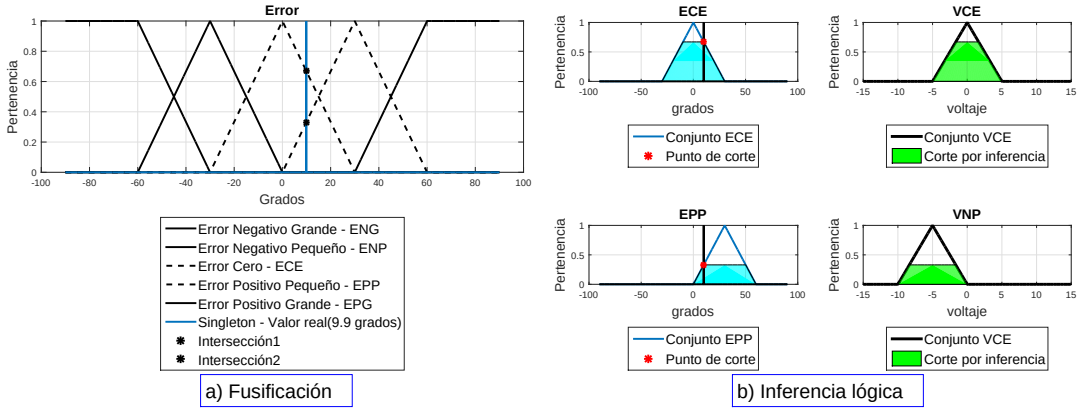


Figura D.5: Esquema conceptual: a) fusificación b) inferencia lógica

de membresía extendida, que recauda cada una de las funciones de membresía recortadas debidas a su inferencia. Para lograr ésto se aplica la unión entre conjuntos difusos, quedando como la ecuación D.1:

$$\mu_{B'}(y) = \mu_{B'_1}(y) \vee \mu_{B'_2}(y) \vee \mu_{B'_3}(y), ... \vee \mu_{B'_i}(y) \tag{D.1}$$

Se considera que al unir cada uno de los conjuntos difusos éstos captan la información de la inferencia lógica de manera general, sin embargo, son un conjunto de soluciones así que, a este conjunto se le debe interpretar d alguna forma para obtener un valor real.

La agregación puede ser vista gráficamente en la FiguraD.6.

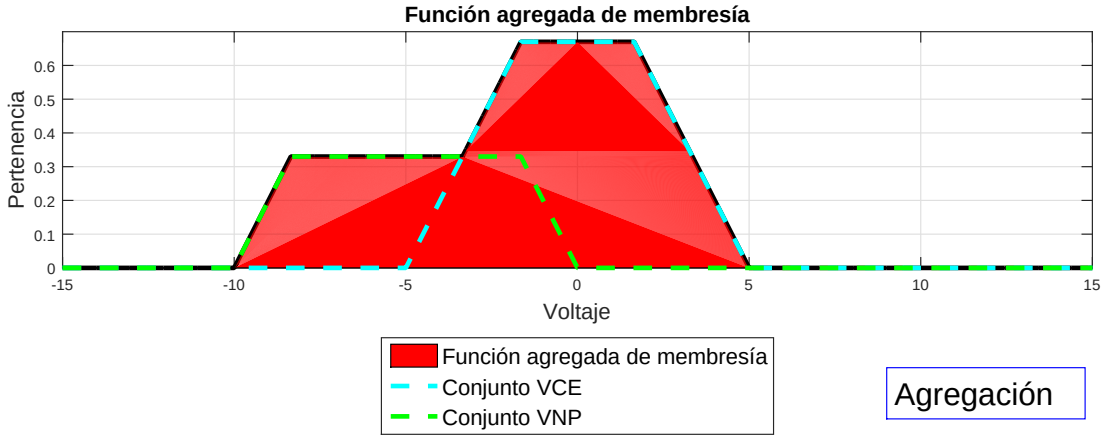


Figura D.6: Agregación de las funciones de membresía

### D.5.2 Métodos de interpretación

Una vez hecha la agregación queda obtener el valor real de salida. Existen varios métodos de interpretación, tales como: centroide, maximo, minimo. Sin embargo, en un controlador de tipo Mamdani el más utilizado es el método del centroide, ya que nos ayuda a obtener resultados más adecuados para el uso del control en lazo cerrado.

A continuación se hace una breve descripción del método del centroide:

$$x_{centroide} = \frac{\sum_1^n (x_n)(\mu_{agregada}(x_n))}{\sum_1^n (\mu_{agregada}(x_n))} \tag{D.2}$$

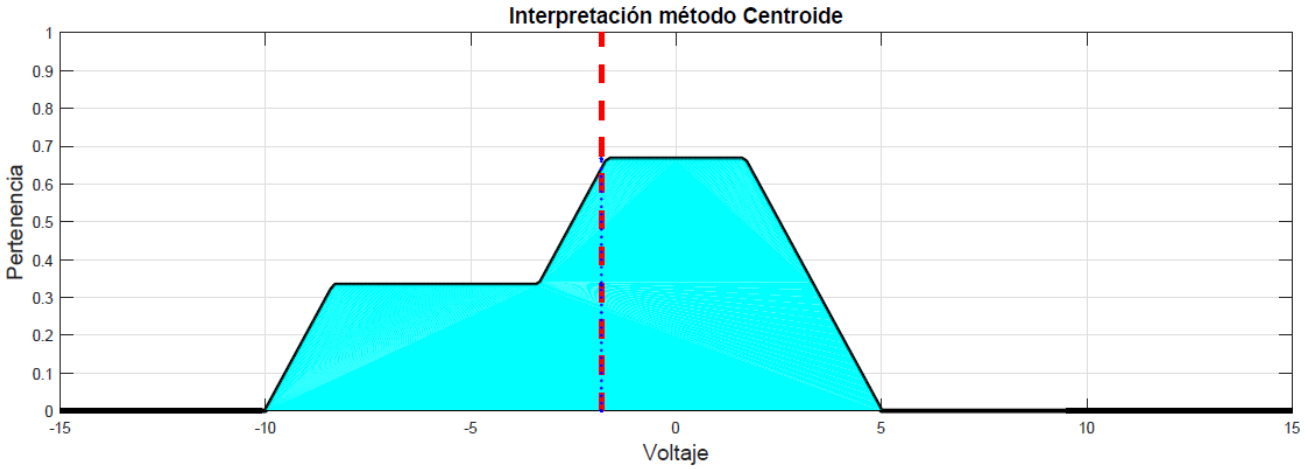


Figura D.7: Interpretación difusa, método del centroide

$$voltaje_{centroide} = \frac{\sum_{-15}^{15} (voltaje_n)(\mu_{agregada}(voltaje_n))}{\sum_{-15}^{15} (\mu_{agregada}(voltaje_n))} \tag{D.3}$$

## D.6 Sensibilidad

Como se comentó en el estado del arte en la Sección 1.1.2, es posible que los sistemas difusos se adapten a cualquier comportamiento matemático, pudiendo imitar a cualquier función a través de las reglas lingüísticas. La modificación de los conjuntos difusos a lo largo del universo de discurso dará como resultado una adaptación de las entradas y su posterior procesamiento, esto se puede visualizar como el incremento de la sensibilidad en el sistema difuso en cuestión.

En la Figura D.8 (a) se muestra un intervalo definido y constante entre cada conjunto difuso, esta configuración es una equivalencia a una función lineal. Por otro lado en la Figura D.8 (b) se observa un espacio irregular entre cada conjunto difuso, esto trae como consecuencia un comportamiento no lineal del sistema difuso. La configuración elaborada para el sistema difuso está totalmente ligada a la base de conocimientos, dado que estos intervalos se deben considerar conociendo al sistema a controlar [16].

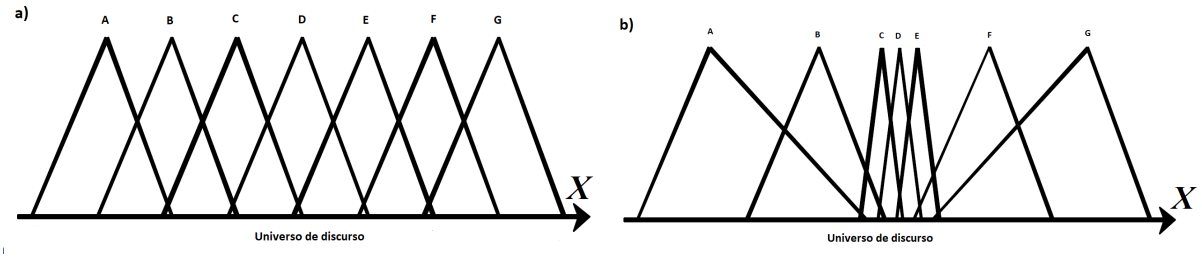


Figura D.8: a)Intervalo regular entre conjuntos difusos b)intervalo irregular entre conjuntos difusos

# Apéndice E

# Estabilidad de controladores difusos

La estabilidad es una parte inherente a los sistemas y se puede definir de muchas maneras y dependiendo de los casos se obtendrá una definición adecuada.

## E.1 Punto de equilibrio

Por definición física y matemática un punto de equilibrio es un punto tal que, todas las trayectorias del sistema sean cero en ese punto; es decir, que las derivadas de los estados del sistema valen cero evaluadas en ese punto. Se dice que el punto de equilibrio es estable si todas las trayectorias del sistema cercanas a éste punto convergen a él. Se dice que el punto de equilibrio es inestable si cualquier trayectoria cercana al punto divergen del mismo.

Cualquier sistema físico modelado por ecuaciones diferenciales tendrá punto o puntos de equilibrio, dependiendo del sistema y si es lineal o no lineal. Existen dos respuestas principales: la respuesta natural de los sistemas físicos es aquella que tiene lugar cuando las condiciones iniciales así como la entrada la sistema valen cero y la respuesta forzada, se observa cuando al sistema se le coloca alguna excitación y/o condiciones diferentes a cero.

## E.2 Controladores difusos

En la mayoría de los casos los controladores son utilizados para tres propósitos: generales, estabilización, regulación o seguimiento. Los controladores difusos serán diseñados según el objetivo. Las reglas elegidas para los controladores difusos deben seguir una tendencia estable.

Cuando una planta es modelada por medio de un sistema difuso del tipo Takagi-Sugeno, se puede conocer de manera aproximada el modelado de la planta y por tanto realizar análisis de estabilidad de manera directa a este modelado, por ejemplo, el método directo de Lyapunov. De otra forma un controlador difuso de tipo Mamdani carece de modelo, entonces solo puede considerarse el comportamiento deseado.

En la figura E.1 se muestra un esquema básico de selección para las reglas para el diseño de un controlador que otorgue al sistema utilizado un comportamiento asintóticamente estable; es decir, para llegar al punto de equilibrio.

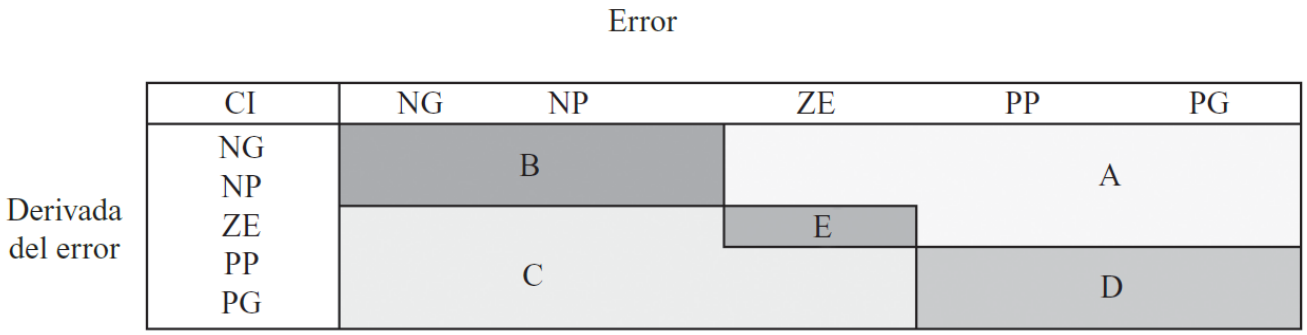


Figura E.1: Configuración de reglas para una respuesta asintóticamente estable [11]

A continuación en la figura E.2 se observa el comportamiento del sistema, cuando la selección de reglas fue según la Figura E.1. Cabe destacar que en general de ésta forma se elaboran los controladores difusos; sin embargo, el orden de las reglas puede ser modificado, pero la estructura del mismo se mantiene. En el borde de la figura E.2 se pueden apreciar los conjuntos difusos en el plano de fase por un costado, ejemplificando un comportamiento asintóticamente estable.

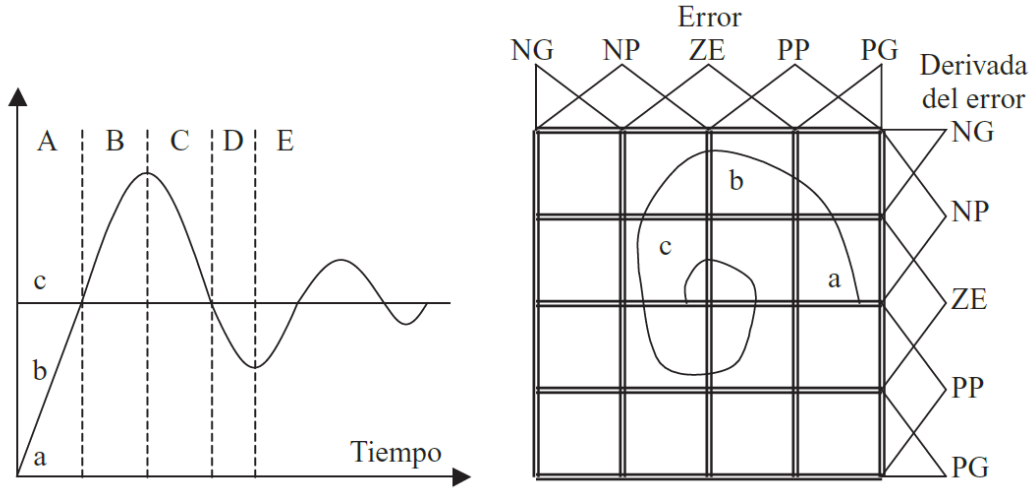


Figura E.2: Se muestra la curva de respuesta y plano de fase sobre el diseño de un controlador [11]

# Apéndice F

# Códigos

## F.1 Control de marcha - Python

Código en C empleado para el control de los servomotores del robot bípedo

```
import um7
import numpy as np
import skfuzzy as fuzz
import time
from time import time as Time_c
import smbus
import math

bus = smbus.SMBus(1)
address_tiva = 0x1F#from teensy 3.1
address_teensy = 0x2F #from tiva C

name = 's'
port = '/dev/serial0'
statevars = ['roll', 'pitch', 'yaw','roll_rate', 'pitch_rate']
IMU = um7.UM7(name, port, statevars, baud=115200)

datos_angulos = np.genfromtxt('/home/pi/pruebas/Tray2.txt', delimiter = '\t'
    )

dim_trayectoria = 1200
data = np.zeros((dim_trayectoria,12)) #trayectoria
a = 120

for n in xrange(0, dim_trayectoria - a):
```

```
25          data[n+a,0] = datos_angulos[n,0]
26          data[n+a,1] = datos_angulos[n,1]
27          data[n+a,2] = datos_angulos[n,2]
28          data[n+a,3] = datos_angulos[n,3]
29          data[n+a,4] = datos_angulos[n,4]
30          data[n+a,5] = datos_angulos[n,5]
31          data[n+a,6] = datos_angulos[n,6]
32          data[n+a,7] = datos_angulos[n,7]
33          data[n+a,8] = datos_angulos[n,8]
34          data[n+a,9] = datos_angulos[n,9]
35          data[n+a,10] = datos_angulos[n,10]
36          data[n+a,11] = datos_angulos[n,11]
37
38  datos_recabados = np.zeros((dim_trayectoria,25))
39
40  def readList_teensy():
41          bytes_i2c = bus.read_i2c_block_data(address_teensy,0,19)
42          return bytes_i2c
43
44  eP_pitch = np.arange(-90, 90,0.1)
45
46  eP_pitch_ENG = fuzz.trimf(eP_pitch, [-90,-90,-15])
47  eP_pitch_ENP = fuzz.trimf(eP_pitch,[-60,-15,0])
48  eP_pitch_ECC = fuzz.trapmf(eP_pitch,[-15,-5,5,15])
49  eP_pitch_EPP = fuzz.trimf(eP_pitch,[0,15,60])
50  eP_pitch_EPG = fuzz.trimf(eP_pitch, [15,90,90])
51
52  eI_pitch = np.arange(-200, 200,0.1)
53
54  eI_pitch_ENG = fuzz.trimf(eI_pitch, [-200,-200,-50])
55  eI_pitch_ENP = fuzz.trimf(eI_pitch,[-100,-50,0])
56  eI_pitch_ECC = fuzz.trimf(eI_pitch,[-50,0,50])
57  eI_pitch_EPP = fuzz.trimf(eI_pitch,[0,50,100])
58  eI_pitch_EPG = fuzz.trimf(eI_pitch, [50,200,200])
59  eD_pitch = np.arange(-300, 300,0.1)
60
61  eD_pitch_ENG = fuzz.trimf(eD_pitch, [-300,-300,-50])
62  eD_pitch_ENP = fuzz.trimf(eD_pitch,[-100,-50,0])
63  eD_pitch_ECC = fuzz.trapmf(eD_pitch,[-50,-5,5,50])
64  eD_pitch_EPP = fuzz.trimf(eD_pitch,[0,50,100])
65  eD_pitch_EPG = fuzz.trimf(eD_pitch, [50,300,300])
66
67  KP = np.arange(0,1.5,0.1)
68
69  KP_mf1 = fuzz.trapmf(KP, [0, 0,0.2,0.4])
70  KP_mf2 = fuzz.trimf(KP, [0.2, 0.4, 0.6])
```

```
71  KP_mf3 = fuzz.trimf(KP,[0.4,0.6,0.8])
72  KP_mf4 = fuzz.trimf(KP, [0.6, 0.9, 1.1])
73  KP_mf5 = fuzz.trapmf(KP, [0.9, 1.1, 1.5,1.5])
74
75  KI = np.arange(0,1.5,0.1)
76
77  KI_mf1 = fuzz.trapmf(KI, [0, 0,0.2,0.4])
78  KI_mf2 = fuzz.trimf(KI, [0.2, 0.4, 0.6])
79  KI_mf3 = fuzz.trimf(KI,[0.4,0.6,0.8])
80  KI_mf4 = fuzz.trimf(KI, [0.6, 0.9, 1.1])
81  KI_mf5 = fuzz.trapmf(KI, [0.9, 1.1, 1.5,1.5])
82
83  KD = np.arange(0,0.5,0.1)
84
85  KD_mf1 = fuzz.trimf(KD,[ 0,0,0.1])
86  KD_mf2 = fuzz.trimf(KD, [0, 0.1, 0.2])
87  KD_mf3 = fuzz.trimf(KD,[0.1,0.2,0.3])
88  KD_mf4 = fuzz.trimf(KD, [0.2, 0.3, 0.4])
89  KD_mf5 = fuzz.trapmf(KD, [0.3, 0.4, 0.5,0.5])
90
91  KP_roll = np.arange(0,1.5,0.1)
92
93  KP_roll_mf1 = fuzz.trapmf(KP_roll, [0, 0,0.1,0.2])
94  KP_roll_mf2 = fuzz.trimf(KP_roll, [0.1, 0.2, 0.4])
95  KP_roll_mf3 = fuzz.trimf(KP_roll,[0.2,0.4,0.8])
96  KP_roll_mf4 = fuzz.trimf(KP_roll, [0.6, 0.8, 1.1])
97  KP_roll_mf5 = fuzz.trapmf(KP_roll, [0.8, 1.1, 1.5,1.5])
98
99  KI_roll = np.arange(0,1.5,0.1)
100
101 KI_roll_mf1 = fuzz.trapmf(KI, [0, 0,0.2,0.4])
102 KI_roll_mf2 = fuzz.trimf(KI, [0.2, 0.4, 0.6])
103 KI_roll_mf3 = fuzz.trimf(KI,[0.4,0.6,0.8])
104 KI_roll_mf4 = fuzz.trimf(KI, [0.6, 0.9, 1.1])
105 KI_roll_mf5 = fuzz.trapmf(KI, [0.9, 1.1, 1.5,1.5])
106
107 KD_roll = np.arange(0,0.5,0.1)
108
109 KD_roll_mf1 = fuzz.trimf(KD,[ 0,0,0.05])
110 KD_roll_mf2 = fuzz.trimf(KD, [0.05, 0.1, 0.2])
111 KD_roll_mf3 = fuzz.trimf(KD,[0.1,0.2,0.3])
112 KD_roll_mf4 = fuzz.trimf(KD, [0.2, 0.3, 0.4])
113 KD_roll_mf5 = fuzz.trapmf(KD, [0.3, 0.4, 0.5,0.5])
114
115 Ang_RodillaD = np.arange(0,50,1)
116
```

```
117 Ang_RodillaD_mf1 = fuzz.trapmf(Ang_RodillaD,[0,0,5,10])
118 Ang_RodillaD_mf2 = fuzz.trimf(Ang_RodillaD,[5, 10,15])
119 Ang_RodillaD_mf3 = fuzz.trimf(Ang_RodillaD,[5,15,25])
120 Ang_RodillaD_mf4 = fuzz.trimf(Ang_RodillaD,[15,30,35])
121 Ang_RodillaD_mf5 = fuzz.trapmf(Ang_RodillaD,[30,35,50,50])
122
123 Ang_RodillaI = np.arange(0,50,1)
124
125 Ang_RodillaI_mf1 = fuzz.trapmf(Ang_RodillaI,[0,0,5,10])
126 Ang_RodillaI_mf2 = fuzz.trimf(Ang_RodillaI,[5, 10,15])
127 Ang_RodillaI_mf3 = fuzz.trimf(Ang_RodillaI,[5,15,25])
128 Ang_RodillaI_mf4 = fuzz.trimf(Ang_RodillaI,[15,30,35])
129 Ang_RodillaI_mf5 = fuzz.trapmf(Ang_RodillaI,[30,35,50,50])
130
131
132 def ErrorP_category(error_in):
133         eP_pitch_ENG_cat = fuzz.interp_membership(eP_pitch,eP_pitch_ENG,
               error_in)
134         eP_pitch_ENP_cat = fuzz.interp_membership(eP_pitch,eP_pitch_ENP,
               error_in)
135         eP_pitch_ECC_cat = fuzz.interp_membership(eP_pitch,eP_pitch_ECC,
               error_in)
136         eP_pitch_EPP_cat = fuzz.interp_membership(eP_pitch,eP_pitch_EPP,
               error_in)
137         eP_pitch_EPG_cat = fuzz.interp_membership(eP_pitch,eP_pitch_EPG,
               error_in)
138         return dict(NG = eP_pitch_ENG_cat, NP = eP_pitch_ENP_cat, CE =
               eP_pitch_ECC_cat,
139                                 PP = eP_pitch_EPP_cat, PG =
                                       eP_pitch_EPG_cat)
140
141 def ErrorI_category(error_in):
142         eI_pitch_ENG_cat = fuzz.interp_membership(eI_pitch,eI_pitch_ENG,
               error_in)
143         eI_pitch_ENP_cat = fuzz.interp_membership(eI_pitch,eI_pitch_ENP,
               error_in)
144         eI_pitch_ECC_cat = fuzz.interp_membership(eI_pitch,eI_pitch_ECC,
               error_in)
145         eI_pitch_EPP_cat = fuzz.interp_membership(eI_pitch,eI_pitch_EPP,
               error_in)
146         eI_pitch_EPG_cat = fuzz.interp_membership(eI_pitch,eI_pitch_EPG,
               error_in)
147         return dict(NG = eI_pitch_ENG_cat, NP = eI_pitch_ENP_cat, CE =
               eI_pitch_ECC_cat,
148                                 PP = eI_pitch_EPP_cat, PG =
                                       eI_pitch_EPG_cat)
```

```
149
150 def ErrorD_category(error_in):
151         eD_pitch_ENG_cat = fuzz.interp_membership(eD_pitch,eD_pitch_ENG,
                error_in)
152         eD_pitch_ENP_cat = fuzz.interp_membership(eD_pitch,eD_pitch_ENP,
                error_in)
153         eD_pitch_ECC_cat = fuzz.interp_membership(eD_pitch,eD_pitch_ECC,
                error_in)
154         eD_pitch_EPP_cat = fuzz.interp_membership(eD_pitch,eD_pitch_EPP,
                error_in)
155         eD_pitch_EPG_cat = fuzz.interp_membership(eD_pitch,eD_pitch_EPG,
                error_in)
156         return dict(NG = eD_pitch_ENG_cat, NP = eD_pitch_ENP_cat, CE =
                eD_pitch_ECC_cat,
157                                 PP = eD_pitch_EPP_cat, PG =
                                       eD_pitch_EPG_cat)
158
159 def Limites(angulo):
160         if angulo > 180:
161                 angulo = 180
162         if angulo < 0:
163                 angulo = 0
164         return angulo
165
166 ahora_t = 0
167 ultimo_t = 0
168 dt = 0
169 error_sum_pitch = 0
170 error_sum_roll = 0
171 contador = 0
172 tiempo = 0
173 suma_tiempo = 0
174 filter_rate_pitch = 0
175 filter_rate_roll = 0
176
177 try:
178
179         for i in xrange(0,dim_trayectoria):
180
181                                 ahora_t = Time_c()
182                                 IMU.catchsample()
183                                 error_roll = 0 - IMU.state['pitch']
184                                 error_pitch = 0 - IMU.state['roll']
185                                 error_rate_pitch = 0 - IMU.state['roll_rate'
                                        ]
186                                 error_rate_roll = 0 - IMU.state['roll_rate']
```

```
187                                 filter_rate_pitch = filter_rate_pitch +
                                       0.1*(error_rate_pitch - filter_rate_pitch
                                       )
188                                 filter_rate_roll = filter_rate_roll +0.1*(
                                       error_rate_roll - filter_rate_roll)
189                                 ultimo_t = Time_c()
190                                 dt = ultimo_t - ahora_t
191                                 suma_tiempo  += dt
192                                 error_sum_pitch += (error_pitch*dt)
193                                 error_sum_roll += (error_roll*dt)
194                                 #categorias errores pitch
195                                 eP_rule = ErrorP_category(error_pitch)
196                                 eI_rule = ErrorI_category(error_sum_pitch)
197                                 eD_rule = ErrorD_category(filter_rate_pitch)
198                                 #categorias error roll
199                                 eP_roll_rule = ErrorP_category(error_roll)
200                                 eI_roll_rule = ErrorI_category(
                                       error_sum_roll)
201                                 eD_roll_rule = ErrorD_category(
                                       filter_rate_roll)
202
203                                 regla1_KP = np.fmax(eP_rule['NG'],eD_rule['
                                       NG'])
204                                 regla2_KP = np.fmax(eP_rule['NP'],eD_rule['
                                       NP'])
205                                 regla3_KP = np.fmax(eP_rule['CE'],eD_rule['
                                       CE'])
206                                 regla4_KP = np.fmax(eP_rule['PP'],eD_rule['
                                       PP'])
207                                 regla5_KP = np.fmax(eP_rule['PG'],eD_rule['
                                       PG'])
208
209                                 regla1_KI = np.fmax(eP_rule['NG'],eI_rule['
                                       NG'])
210                                 regla2_KI = np.fmax(eP_rule['NP'],eI_rule['
                                       NP'])
211                                 regla3_KI = np.fmax(eP_rule['CE'],eI_rule['
                                       CE'])
212                                 regla4_KI = np.fmax(eP_rule['PP'],eI_rule['
                                       PP'])
213                                 regla5_KI = np.fmax(eP_rule['PG'],eI_rule['
                                       PG'])
214
215                                 regla1_KD = np.fmax(eP_rule['NG'],eD_rule['
                                       NG'])
216                                 regla2_KD = eP_rule['NP']
```

```
217                                 regla3_KD = eP_rule['CE']
218                                 regla4_KD = eP_rule['PP']
219                                 regla5_KD = np.fmax(eP_rule['PG'],eD_rule['
                                        PG'])
220
221                                 regla1_KP_roll = np.fmax(eP_roll_rule['NG'],
                                        eD_roll_rule['NG'])
222                                 regla2_KP_roll = np.fmax(eP_roll_rule['NP'],
                                        eD_roll_rule['NP'])
223                                 regla3_KP_roll = np.fmax(eP_roll_rule['CE'],
                                        eD_roll_rule['CE'])
224                                 regla4_KP_roll = np.fmax(eP_roll_rule['PP'],
                                        eD_roll_rule['PP'])
225                                 regla5_KP_roll = np.fmax(eP_roll_rule['PG'],
                                        eD_roll_rule['PG'])
226
227                                 regla1_KI_roll = np.fmax(eP_roll_rule['NG'],
                                        eI_roll_rule['NG'])
228                                 regla2_KI_roll = np.fmax(eP_roll_rule['NP'],
                                        eI_roll_rule['NP'])
229                                 regla3_KI_roll = np.fmax(eP_roll_rule['CE'],
                                        eI_roll_rule['CE'])
230                                 regla4_KI_roll = np.fmax(eP_roll_rule['PP'],
                                        eI_roll_rule['PP'])
231                                 regla5_KI_roll = np.fmax(eP_roll_rule['PG'],
                                        eI_roll_rule['PG'])
232
233                                 regla1_KD_roll = np.fmax(eP_roll_rule['NG'],
                                        eD_roll_rule['NG'])
234                                 regla2_KD_roll = eP_roll_rule['NP']
235                                 regla3_KD_roll = eP_roll_rule['CE']
236                                 regla4_KD_roll = eP_roll_rule['PP']
237                                 regla5_KD_roll = np.fmax(eP_roll_rule['PG'],
                                        eD_roll_rule['PG'])
238
239                                 #ANGULO RODILLA
240                                 regla1_ang_rodillaD = eP_roll_rule['NG']
241                                 regla2_ang_rodillaD = eP_roll_rule['NP']
242                                 regla3_ang_rodillaD = eP_roll_rule['CE']
243                                 regla4_ang_rodillaD = eP_roll_rule['PP']
244                                 regla5_ang_rodillaD = eP_roll_rule['PG']
245
246                                 #ANGULO RODILLA
247                                 regla1_ang_rodillaI = eP_roll_rule['NG']
248                                 regla2_ang_rodillaI = eP_roll_rule['NP']
249                                 regla3_ang_rodillaI = eP_roll_rule['CE']
```

```
250                                 regla4_ang_rodillaI = eP_roll_rule['PP']
251                                 regla5_ang_rodillaI = eP_roll_rule['PG']
252
253                                 #Implementacion de reglas KP,KI,KD (pitch y
                                       roll), a_D y a_I
254                                 regla1_KP_act = np.fmin(regla1_KP,KP_mf5)
255                                 regla2_KP_act = np.fmin(regla2_KP,KP_mf3)
256                                 regla3_KP_act = np.fmin(regla3_KP,KP_mf1)
257                                 regla4_KP_act = np.fmin(regla4_KP,KP_mf3)
258                                 regla5_KP_act = np.fmin(regla5_KP,KP_mf5)
259
260                                 regla1_KI_act = np.fmin(regla1_KI,KI_mf1)
261                                 regla2_KI_act = np.fmin(regla2_KI,KI_mf3)
262                                 regla3_KI_act = np.fmin(regla3_KI,KI_mf4)
263                                 regla4_KI_act = np.fmin(regla4_KI,KI_mf3)
264                                 regla5_KI_act = np.fmin(regla5_KI,KI_mf1)
265
266                                 regla1_KD_act = np.fmin(regla1_KD,KD_mf4)
267                                 regla2_KD_act = np.fmin(regla2_KD,KD_mf2)
268                                 regla3_KD_act = np.fmin(regla3_KD,KD_mf1)
269                                 regla4_KD_act = np.fmin(regla4_KD,KD_mf2)
270                                 regla5_KD_act = np.fmin(regla5_KD,KD_mf4)
271
272                                 regla1_KP_roll_act = np.fmin(regla1_KP_roll,
                                       KP_roll_mf2)
273                                 regla2_KP_roll_act = np.fmin(regla2_KP_roll,
                                       KP_roll_mf1)
274                                 regla3_KP_roll_act = np.fmin(regla3_KP_roll,
                                       KP_roll_mf1)
275                                 regla4_KP_roll_act = np.fmin(regla4_KP_roll,
                                       KP_roll_mf1)
276                                 regla5_KP_roll_act = np.fmin(regla5_KP_roll,
                                       KP_roll_mf2)
277
278                                 regla1_KI_roll_act = np.fmin(regla1_KI_roll,
                                       KI_roll_mf1)
279                                 regla2_KI_roll_act = np.fmin(regla2_KI_roll,
                                       KI_roll_mf2)
280                                 regla3_KI_roll_act = np.fmin(regla3_KI_roll,
                                       KI_roll_mf3)
281                                 regla4_KI_roll_act = np.fmin(regla4_KI_roll,
                                       KI_roll_mf2)
282                                 regla5_KI_roll_act = np.fmin(regla5_KI_roll,
                                       KI_roll_mf1)
283
284                                 regla1_KD_roll_act = np.fmin(regla1_KD_roll,
```

```
                                    KD_roll_mf2)
285                                 regla2_KD_roll_act = np.fmin(regla2_KD_roll,
                                    KD_roll_mf1)
286                                 regla3_KD_roll_act = np.fmin(regla3_KD_roll,
                                    KD_roll_mf1)
287                                 regla4_KD_roll_act = np.fmin(regla4_KD_roll,
                                    KD_roll_mf1)
288                                 regla5_KD_roll_act = np.fmin(regla5_KD_roll,
                                    KD_roll_mf2)
289
290                                 regla1_ang_rodillaD_act = np.fmin(
                                    regla1_ang_rodillaD,Ang_RodillaD_mf5)
291                                 regla2_ang_rodillaD_act = np.fmin(
                                    regla2_ang_rodillaD,Ang_RodillaD_mf3)
292                                 regla3_ang_rodillaD_act = np.fmin(
                                    regla3_ang_rodillaD,Ang_RodillaD_mf1)
293                                 regla4_ang_rodillaD_act = np.fmin(
                                    regla4_ang_rodillaD,Ang_RodillaD_mf3)
294                                 regla5_ang_rodillaD_act = np.fmin(
                                    regla5_ang_rodillaD,Ang_RodillaD_mf5)
295
296                                 regla1_ang_rodillaI_act = np.fmin(
                                    regla1_ang_rodillaI,Ang_RodillaI_mf5)
297                                 regla2_ang_rodillaI_act = np.fmin(
                                    regla2_ang_rodillaI,Ang_RodillaI_mf3)
298                                 regla3_ang_rodillaI_act = np.fmin(
                                    regla3_ang_rodillaI,Ang_RodillaI_mf1)
299                                 regla4_ang_rodillaI_act = np.fmin(
                                    regla4_ang_rodillaI,Ang_RodillaI_mf3)
300                                 regla5_ang_rodillaI_act = np.fmin(
                                    regla5_ang_rodillaI,Ang_RodillaI_mf5)
301
302                                 SumaMembresiasKP = np.fmax(regla1_KP_act,np.
                                    fmax(regla2_KP_act,np.fmax(regla3_KP_act,
                                    np.fmax(regla4_KP_act,regla5_KP_act))))
303                                 SumaMembresiasKI =  np.fmax(regla1_KI_act,np
                                    .fmax(regla2_KI_act,np.fmax(regla3_KI_act
                                    ,np.fmax(regla4_KI_act,regla5_KI_act))))
304                                 SumaMembresiasKD = np.fmax(regla1_KD_act,np.
                                    fmax(regla2_KD_act,np.fmax(regla3_KD_act,
                                    np.fmax(regla4_KD_act,regla5_KD_act))))
305
306                                 SumaMembresiasKP_roll = np.fmax(
                                    regla1_KP_roll_act,np.fmax(
                                    regla2_KP_roll_act,np.fmax(
                                    regla3_KP_roll_act,np.fmax(
```

```
                                    regla4_KP_roll_act,regla5_KP_roll_act))))
307                                 SumaMembresiasKI_roll = np.fmax(
                                    regla1_KI_roll_act,np.fmax(
                                    regla2_KI_roll_act,np.fmax(
                                    regla3_KI_roll_act,np.fmax(
                                    regla4_KI_roll_act,regla5_KI_roll_act))))
308                                 SumaMembresiasKD_roll = np.fmax(
                                    regla1_KD_roll_act,np.fmax(
                                    regla2_KD_roll_act,np.fmax(
                                    regla3_KD_roll_act,np.fmax(
                                    regla4_KD_roll_act,regla5_KD_roll_act))))
309
310                                 SumaMembresias_a_D = np.fmax(
                                    regla1_ang_rodillaD_act,np.fmax(
                                    regla2_ang_rodillaD_act,np.fmax(
                                    regla3_ang_rodillaD_act,np.fmax(
                                    regla4_ang_rodillaD_act,
                                    regla5_ang_rodillaD_act))))
311                                 SumaMembresias_a_I = np.fmax(
                                    regla1_ang_rodillaI_act,np.fmax(
                                    regla2_ang_rodillaI_act,np.fmax(
                                    regla3_ang_rodillaI_act,np.fmax(
                                    regla4_ang_rodillaI_act,
                                    regla5_ang_rodillaI_act))))
312
313                                 #defuzzification, resultado:
314                                 Kp_pitch = fuzz.centroid(KP,
                                    SumaMembresiasKP)
315                                 Ki_pitch = fuzz.centroid(KI,
                                    SumaMembresiasKI)
316                                 Kd_pitch = fuzz.centroid(KD,
                                    SumaMembresiasKD)
317
318                                 Kp_roll = fuzz.centroid(KP_roll,
                                    SumaMembresiasKP_roll)
319                                 Ki_roll = fuzz.centroid(KI_roll,
                                    SumaMembresiasKI_roll)
320                                 Kd_roll = fuzz.centroid(KD_roll,
                                    SumaMembresiasKD_roll)
321
322
323                                 a_D = fuzz.centroid(Ang_RodillaD,
                                    SumaMembresias_a_D)
324                                 a_I = fuzz.centroid(Ang_RodillaI,
                                    SumaMembresias_a_I)
325
```

```
326                                     #ADQUISiCION DE DATOS, GRAFICAS
327                                     datos_recabados[i,0] = suma_tiempo
328                                     datos_recabados[i,1] = error_pitch
329                                     datos_recabados[i,2] = error_roll
330                                     datos_recabados[i,3] = error_rate_pitch
331                                     datos_recabados[i,4] = error_rate_roll
332                                     datos_recabados[i,5] = filter_rate_pitch
333                                     datos_recabados[i,6] = filter_rate_roll
334                                     datos_recabados[i,7] = error_sum_pitch
335                                     datos_recabados[i,8] = error_sum_roll
336                                     datos_recabados[i,9] = Kp_pitch
337                                     datos_recabados[i,10] = Ki_pitch
338                                     datos_recabados[i,11] = Kd_pitch
339                                     datos_recabados[i,12] = Kp_roll
340                                     datos_recabados[i,13] = Ki_roll
341                                     datos_recabados[i,14] = Kd_roll
342
343                                     if (i < a + 1):
344                                             servo11 = 95
345                                             servo12 = 90
346                                             servo21 = Limites(int(90 + Kp_roll*
                                                error_roll + Ki_roll*
                                                error_sum_roll + Kd_roll*
                                                error_rate_roll))
347                                             servo22 = Limites(int(90 + Kp_roll*
                                                error_roll + Ki_roll*
                                                error_sum_roll + Kd_roll*
                                                error_rate_roll))
348                                             servo31 = Limites(int(150 - Kp_pitch
                                                *error_pitch - Ki_pitch*
                                                error_sum_pitch - Kd_pitch*
                                                error_rate_pitch - a_D))
349                                             servo32 = Limites(int(45 + Kp_pitch*
                                                error_pitch + Ki_pitch*
                                                error_sum_pitch + Kd_pitch*
                                                error_rate_pitch +  a_I))
350                                             servo41 = Limites(int(65 - Kp_pitch*
                                                error_pitch - Ki_pitch*
                                                error_sum_pitch - Kd_pitch*
                                                error_rate_pitch - a_D))
351                                             servo42 = Limites(int(120 + Kp_pitch
                                                *error_pitch + Ki_pitch*
                                                error_sum_pitch + Kd_pitch*
                                                error_rate_pitch + a_I))
352                                             servo51 = Limites(int(120 - Kp_pitch
                                                *error_pitch - Ki_pitch*
```

```
                                        error_sum_pitch - Kd_pitch*
                                        error_rate_pitch - a_D))
353                                  servo52 = Limites(int(65 + Kp_pitch*
                                        error_pitch + Ki_pitch*
                                        error_sum_pitch + Kd_pitch*
                                        error_rate_pitch + a_I))
354                                  servo61 = Limites(int(90 + Kp_roll*
                                        error_roll+ Ki_roll*
                                        error_sum_roll + Kd_roll*
                                        error_rate_roll))
355                                  servo62 = Limites(int(95 + Kp_roll*
                                        error_roll+ Ki_roll*
                                        error_sum_roll + Kd_roll*
                                        error_rate_roll))
356                              if(i > a ):
357                                  servo12 = int(data[i,0])
358                                  servo11 = int(data[i,1])
359                                  servo22 = Limites(int(data[i,2] +
                                        Kp_roll*error_roll + Ki_roll*
                                        error_sum_roll + Kd_roll*
                                        error_rate_roll))
360                                  servo21 = Limites(int(data[i,3] +
                                        Kp_roll*error_roll + Ki_roll*
                                        error_sum_roll + Kd_roll*
                                        error_rate_roll))
361                                  servo32 = Limites(int(data[i,4] +
                                        Kp_pitch*error_pitch + Ki_pitch*
                                        error_sum_pitch + Kd_pitch*
                                        error_rate_pitch ))
362                                  servo31 = Limites(int(data[i,5] -
                                        Kp_pitch*error_pitch - Ki_pitch*
                                        error_sum_pitch - Kd_pitch*
                                        error_rate_pitch ))
363                                  servo42 = Limites(int(data[i,6] +
                                        Kp_pitch*error_pitch + Ki_pitch*
                                        error_sum_pitch + Kd_pitch*
                                        error_rate_pitch ))
364                                  servo41 = Limites(int(data[i,7] -
                                        Kp_pitch*error_pitch - Ki_pitch*
                                        error_sum_pitch - Kd_pitch*
                                        error_rate_pitch ))
365                                  servo52 = Limites(int(data[i,8] +
                                        Kp_pitch*error_pitch + Ki_pitch*
                                        error_sum_pitch + Kd_pitch*
                                        error_rate_pitch ))
366                                  servo51 = Limites(int(data[i,9] -
```

```
                                                Kp_pitch*error_pitch - Ki_pitch*
                                                error_sum_pitch - Kd_pitch*
                                                error_rate_pitch ))
367                                           servo62 = Limites(int(data[i,10] +
                                                Kp_roll*error_roll+ Ki_roll*
                                                error_sum_roll + Kd_roll*
                                                error_rate_roll))
368                                           servo61 = Limites(int(data[i,11] +
                                                Kp_roll*error_roll+ Ki_roll*
                                                error_sum_roll + Kd_roll*
                                                error_rate_roll))
369
370                                   datos_recabados[i,15] = servo22
371                                   datos_recabados[i,16] = servo21
372                                   datos_recabados[i,17] = servo32
373                                   datos_recabados[i,18] = servo31
374                                   datos_recabados[i,19] = servo42
375                                   datos_recabados[i,20] = servo41
376                                   datos_recabados[i,21] = servo52
377                                   datos_recabados[i,22] = servo51
378                                   datos_recabados[i,23] = servo62
379                                   datos_recabados[i,24] = servo61
380
381                                   bus.write_i2c_block_data(address_tiva,1,[0,
                                        servo11,servo12,servo21,servo22,servo31,
                                        servo32,servo41,servo42,servo51,servo52,
                                        servo61,servo62])
382
383                                   print error_pitch," Kp_pitch: ",Kp_pitch
384           np.savetxt('datosPrueba_caminata_10_grados.txt',datos_recabados,fmt
                 = '%10.5f',delimiter = '\t')
385   except KeyboardInterrupt:
386           print " "
387           print "Termino Control"
```

## F.2 Tarjeta de adquisición - C++ (Processing)

Código en C empleado para el control de los servomotores del robot bípedo

```
1   int numerosBinarios [16][4] =   {
2                                 {0,0,0,0},
3                                 {0,0,0,1},
4                                 {0,0,1,0},
```

```
5                                   {0,0,1,1},
6                                   {0,1,0,0},
7                                   {0,1,0,1},
8                                   {0,1,1,0},
9                                   {0,1,1,1},
10                                  {1,0,0,0},
11                                  {1,0,0,1},
12                                  {1,0,1,0},
13                                  {1,0,1,1},
14                                  {1,1,0,0},
15                                  {1,1,0,1},
16                                  {1,1,1,0},
17                                  {1,1,1,1}
18                                  }; //Matriz de 16x4 que contiene los numeros
                                       binarios,
19                                  //para los canales de seleccion de los
                                       sensores de fuerza.
20
21  float conversionIzquierdo[9][5] = {
22                                      {+0.3027,-1.9701,+4.8688,-2.8849,+0.0019},
23                                      {+0.3168,-2.0886,+5.1297,-3.3509,+0.0003},
24                                      {+0.3145,-1.9947,+4.6649,-2.4090,+0.0028},
25                                      {+0.2052,-1.1116,+2.0559,+0.6049,+0.0117},
26                                      {+0.0000,+0.0000,+0.0000,+0.0000,+0.0000},
27                                      {+0.2463,-1.3114,+2.6863,-0.1932,+0.0026},
28                                      {+0.2541,-1.6868,+4.1244,-2.2417,+0.0005},
29                                      {+0.1297,-0.5123,+0.6333,+1.8661,+0.0095},
30                                      {+0.1849,-0.9479,+1.7859,+0.3678,+0.0058},
31                                      };//Matriz 9x5 que ayudara en la
                                           conversion de tension a fueza.
32
33  float conversionDerecho[9][5] = {
34                                     {+0.1762,-0.9540,+1.8915,+0.3617,+0.0052},
35                                     {+0.2178,-1.1379,+2.4998,-0.5883,+0.0043},
36                                     {+0.1852,-0.9604,+1.9626,+0.2848,-0.0004},
37                                     {+0.2837,-1.7585,+3.9822,-1.7387,+0.0009},
38                                     {+0.0000,+0.0000,+0.0000,+0.0000,+0.0000},
39                                     {+0.2291,-1.4029,+3.2722,-1.0680,+0.0051},
40                                     {+0.1117,-0.3331,+0.0902,+2.2172,+0.0002},
41                                     {+0.2599,-1.6504,+3.9333,-2.0373,+0.0007},
42                                     {+0.1683,-0.9886,+2.4650,-0.8432,+0.0022},
43                                     };//Matriz 9x5 que ayudara en la conversion
                                        de tension a fueza.
44
45  float vP_11 = 0, vP_12 = 0,vP_21 = 0, vP_22 = 0, vP_31 = 0, vP_32 = 0, vP_41
        = 0, vP_42 = 0, vP_51 = 0, vP_52 = 0, vP_61 = 0, vP_62 = 0;
```

```
46
47  float pot = 0;
48
49  float posicionY[3]= {13.5,-7.5,-28.5};//mm
50  float posicionX[3]= {54.0,20.0,-13.0};//mm
51
52  float F1_i = 0, F2_i = 0, F3_i = 0, F4_i = 0, F5_i = 0, F6_i = 0;
53  float F1_d = 0, F2_d = 0, F3_d = 0, F4_d = 0, F5_d = 0, F6_d = 0;
54  //Posiciones (x,y) de los sensores de fuerza en los pies.
55
56  const int bit1 = 5 , bit2 = 4, bit3 = 3, bit4 = 2; //Pines que enviaran el
       numero en binario al multiplexor.
57  const int bit1_pot = 23 , bit2_pot = 22, bit3_pot = 12, bit4_pot = 20; //
       Pines que enviaran el numero en binario al multiplexor
58
59  float LecturaTension[18]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
60  float LecturaPot[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
61
62  float fuerzaDerecho[9] = {0,0,0,0,0,0,0,0,0};//Vector de fuerzas pie derecho
       .
63  float fuerzaIzquierdo[9]= {0,0,0,0,0,0,0,0,0};//Vector de fuerzas pie
       izquierdo.
64  float matrizDerecho[3][3];           //Matriz de fuerzas pie derecho.
65  float matrizIzquierdo[3][3];         //Matriz de fuerzas pie izquierdo.
66  float numeradorXd = 0, numeradorYd = 0, denominadord = 0;
67  float numeradorXi = 0, numeradorYi = 0, denominadori = 0;
68  float sensor,aux1,aux2,aux3,aux4, Xzmpi , Yzmpi, Xzmpd, Yzmpd;
69  int k;
70  float lecturaTensionDerecho[9] = {0,0,0,0,0,0,0,0,0};        //Vector de
       lecturas de los sensores pie derecho.
71  float lecturaTensionIzquierdo[9] = {0,0,0,0,0,0,0,0,0};//Vector de lecturas
       de los sensores pie izquierdo.
72
73  int pos_s11, pos_s12, pos_s21, pos_s22, pos_s31, pos_s32;
74  int pos_s41, pos_s42, pos_s51, pos_s52, pos_s61, pos_s62;
75
76  float tiempo_ini = 0, tiempo_fin = 0, delta = 0, tiempo_suma = 0;
77
78  void setup()
79  {
80
81    Serial.begin(115200);
82    analogReadRes(16);
83    pinMode(bit1, OUTPUT);
84    pinMode(bit2, OUTPUT);
85    pinMode(bit3, OUTPUT);
```

```
86    pinMode(bit3, OUTPUT);
87    pinMode(bit1_pot, OUTPUT);
88    pinMode(bit2_pot, OUTPUT);
89    pinMode(bit3_pot, OUTPUT);
90    pinMode(bit4_pot, OUTPUT);
91    pinMode(13, OUTPUT);
92    digitalWrite(13,1);
93    //pinMode(EN,OUTPUT);
94  }
95
96  void loop()
97  {
98        numeradorYi = 0;
99        numeradorXi = 0;
100       denominadori = 0;
101       numeradorYd = 0;
102       numeradorXd = 0;
103       denominadord = 0;
104
105 k=0;
106 for( int j=0 ; j<16 ; j++)
107 {
108       enviarBinarioS(numerosBinarios[j][0],numerosBinarios[j][1],
            numerosBinarios[j][2],numerosBinarios[j][3]);//envia el respectivo
            canal seleccionado
109       sensor=analogRead(A2);
110       if (sensor < 80){ sensor = 0; }
111       if(j<8) { lecturaTensionDerecho[k] = sensor*(3.3 / 4095); }
112       else {    lecturaTensionIzquierdo[k-9] = sensor*(3.3 / 4095); }
113       if (k==3 || k==12){ k++; }
114       k++;
115       delayMicroseconds(100);
116       enviarBinarioP(numerosBinarios[j][0],numerosBinarios[j][1],
            numerosBinarios[j][2],numerosBinarios[j][3]);
117       pot = analogRead(A1);
118       LecturaPot[j] = pot*(3.3 / 65535.0);
119       delayMicroseconds(100);
120 }
121
122 pos_s11 = int(-72.6442*LecturaPot[0] + 207.5405);
123 pos_s12 = int(-74.1056*LecturaPot[6] + 199.9111);
124 pos_s21 = int(-74.7734*LecturaPot[1] + 203.2582);
125 pos_s22 = int(-73.3606*LecturaPot[7] + 203.6811);
126 pos_s31 = int(-74.0196*LecturaPot[2] + 195.9776);
127 pos_s32 = int(-75.2146*LecturaPot[8] + 200.7762);
128 pos_s41 = int(-73.3096*LecturaPot[3] + 203.425);
```

```
129  pos_s42 = int(-73.9639*LecturaPot[9] + 204.5746);
130  pos_s51 = int(-73.7304*LecturaPot[4] + 204.8248);
131  pos_s52 = int(-76.5749*LecturaPot[10] + 198.678);
132  pos_s61 = int(-74.4718*LecturaPot[5] + 210.3748);
133  pos_s62 = int(-73.9041*LecturaPot[11] + 204.943);
134
135  for(int i = 0; i < 9 ; i++)
136    {
137
138       fuerzaIzquierdo[i] = tensionFuerza(lecturaTensionIzquierdo[i],
          conversionIzquierdo[i][0],conversionIzquierdo[i][1],
139                            conversionIzquierdo[i][2],conversionIzquierdo[i
                               ][3],conversionIzquierdo[i][4]);
140
141       fuerzaDerecho[i] = tensionFuerza(lecturaTensionDerecho[i],
          conversionDerecho[i][0],conversionDerecho[i][1],
142                            conversionDerecho[i][2],conversionDerecho[i][3],
                               conversionDerecho[i][4]);
143    }
144    //Se convierten los datos de tension a fuerza del pie izquierdo; FI = {f1,
          f2,f3,f4,0,f5,f6,f7,f8}
145    //Se convierten los datos de tension a fuerza del pie derecho; FD = {f9,
          f10,f11,f12,0,f13,f14,f15,f6}
146
147    F1_i = fuerzaIzquierdo[7] + fuerzaIzquierdo[4] + fuerzaIzquierdo[2];
148    F2_i = fuerzaIzquierdo[6] + fuerzaIzquierdo[1];
149    F3_i = fuerzaIzquierdo[5] + fuerzaIzquierdo[3] + fuerzaIzquierdo[0];
150
151    F4_i = fuerzaIzquierdo[0] + fuerzaIzquierdo[1] + fuerzaIzquierdo[2];
152    F5_i = fuerzaIzquierdo[3] + fuerzaIzquierdo[5];
153    F6_i = fuerzaIzquierdo[6] + fuerzaIzquierdo[7] + fuerzaIzquierdo[8];
154
155    F1_d = fuerzaDerecho[7] + fuerzaDerecho[4] + fuerzaDerecho[2];
156    F2_d = fuerzaDerecho[6] + fuerzaDerecho[1];
157    F3_d = fuerzaDerecho[5] + fuerzaDerecho[3] + fuerzaDerecho[0];
158
159    F4_d = fuerzaDerecho[6] + fuerzaDerecho[7] + fuerzaDerecho[8];
160    F5_d = fuerzaDerecho[3] + fuerzaDerecho[5];
161    F6_d = fuerzaDerecho[0] + fuerzaDerecho[1] + fuerzaDerecho[2];
162
163
164    denominadori =  fuerzaIzquierdo[0] + fuerzaIzquierdo[1] + fuerzaIzquierdo
          [2] +
165                    fuerzaIzquierdo[3] + fuerzaIzquierdo[5] + fuerzaIzquierdo
                       [6] +
166                    fuerzaIzquierdo[7] + fuerzaIzquierdo[8];
```

```
167
168     denominadord =  fuerzaDerecho[0] + fuerzaDerecho[1] + fuerzaDerecho[2] +
169                     fuerzaDerecho[3] + fuerzaDerecho[5] + fuerzaDerecho[6] +
170                     fuerzaDerecho[7] + fuerzaDerecho[8];
171
172     numeradorXi = F1_i*posicionX[0] + F2_i*posicionX[1] - F3_i*posicionX[2];
173     numeradorYi = -F4_i*posicionY[0] + F5_i*posicionY[1] + F6_i*posicionY[2];
174
175     numeradorXd = F1_d*posicionX[0] + F2_d*posicionX[1] - F3_d*posicionX[2];
176     numeradorYd = F4_d*posicionY[0] - F5_d*posicionY[1] - F6_d*posicionY[2];
177
178   Xzmpi = numeradorXi / denominadori;
179   Yzmpi = numeradorYi / denominadori;
180   Xzmpd = numeradorXd / denominadord;
181   Yzmpd = numeradorYd / denominadord;
182
183     Serial.print(Xzmpi);//d1
184     Serial.print(",");
185     Serial.print(Yzmpi);//d2
186     Serial.print(",");
187     Serial.print(Xzmpd);//d3
188     Serial.print(",");
189     Serial.println(Yzmpd);//d4
190
191     delay(100);
192
193
194   }
195
196   void enviarBinarioS(boolean a, boolean b, boolean c,boolean d)
197   {
198             digitalWrite(bit1,a);
199             digitalWrite(bit2,b);
200             digitalWrite(bit3,c);
201             digitalWrite(bit4,d);
202   }
203
204   void enviarBinarioP(boolean a, boolean b, boolean c,boolean d)
205   {
206             digitalWrite(bit1_pot,a);
207             digitalWrite(bit2_pot,b);
208             digitalWrite(bit3_pot,c);
209             digitalWrite(bit4_pot,d);
210   }//Funcion encargada de escribir el numero en binario usando los pines 52,
        50 y 48.
211
```

```
212 float tensionFuerza(float tension, float a, float b, float c, float d, float
        e)
213 {
214    float Fuerza;
215          Fuerza = +a*tension*tension*tension*tension
216                   +b*tension*tension*tension
217                   +c*tension*tension
218                   +d*tension
219                   +e;
220   return Fuerza;
221 }
```

## F.3 Tarjeta driver - C++ (Processing)

Código en C empleado para el control de los servomotores del robot bípedo

```
1  #include <Servo.h>
2  #include <Wire.h>//Libreria que permite utilizar I2C (Two Wire Interface)
3  #define Esclavo 0x1F//Direccion establecida para la comunicacion, en
       especifico para
4                      //la tarjeta Tiva-C TM4C123G
5
6  Servo s62,s52,s42,s32,s22,s12,s61,s51,s41,s31,s21,s11;  // creo un objeto de
       la clase servo
7
8  byte AngulosRecibidos[13];//Vector que alojara a los angulos recibidos
9                            //provenientes de la Raspberry
10
11 void setup()
12 {
13   Serial.begin(9600);//Configuracion de Baudios
14   Wire.begin(Esclavo);//Se inicia la Comunicacion con el Maestro
15   Wire.onRequest(EnviarDatos);//Funcion utilizada cuando el Maestro pide
       datos (callback)
16   Wire.onReceive(RecibirDatos);//Funcion utilizada para recibir datos del
       Maestro
17
18   s61.attach(PF_2,900,2100);
19   s62.attach(PF_3,750,2250);//<--------750-2250 micro-sec servo hs 5485 mg
20
21   s51.attach(PB_3,900,2100);
22   s52.attach(PC_4,900,2100);
23
```

```
24    s41.attach(PC_5,900,2100);
25    s42.attach(PC_6,900,2100);
26
27    s31.attach(PC_7,900,2100);
28    s32.attach(PF_4,900,2100);
29
30    s21.attach(PA_2,900,2100);
31    s22.attach(PA_3,900,2100);
32
33    s11.attach(PA_4,900,2100);
34    s12.attach(PB_2,900,2100);
35
36  //CADERA
37    s11.write(90); //izquierda  YAW
38    s12.write(90);//derecha
39
40    s21.write(90); //izquierda  ROLL
41    s22.write(90);//derecha
42
43    s31.write(130); //izquierda  PITCH
44    s32.write(65); //derecha
45
46  //RODILLA
47    s41.write(65); //izquierda PITCH
48    s42.write(120); //derecha
49
50  //TOBILLO
51    s51.write(130); //izquierda  PITCH
52    s52.write(55); //65derecha
53
54    s61.write(87); //izquierda  ROLL
55    s62.write(87);  //derecha
56
57  }
58
59  void loop()
60  {
61
62  }
63  void RecibirDatos(int NumeroBytes)
64  {
65      while(Wire.available())//Mientras haya datos en la comunicacion
66      {
67        for (int NumeroDato = 0; NumeroDato < NumeroBytes; NumeroDato++)//Para
              cada dato Recibido se le asigna un numero
68        {
```

```
69            if ( NumeroDato < 14)//Si es menor a 13
70            {
71                 AngulosRecibidos[NumeroDato] = Wire.read();//Se guardan los
                      datos recibidos
72            }
73            else//De otra forma se desechan los datos
74            {
75                 Wire.read();
76            }
77
78          }
79
80       //CADERA
81       s11.write(AngulosRecibidos[2]); //izquierda  YAW
82       s12.write(AngulosRecibidos[3]);//derecha
83
84       s21.write(AngulosRecibidos[4]); //izquierda  ROLL
85       s22.write(AngulosRecibidos[5]);//derecha
86
87       s31.write(AngulosRecibidos[6]); //izquierda  PITCH
88       s32.write(AngulosRecibidos[7]); //derecha
89
90       //RODILLA
91       s41.write(AngulosRecibidos[8]); //izquierda PITCH
92       s42.write(AngulosRecibidos[9]); //derecha
93
94        //TOBILLO
95       s51.write(AngulosRecibidos[10]); //izquierda  PITCH
96       s52.write(AngulosRecibidos[11]); //derecha
97
98       s61.write(AngulosRecibidos[12]); //izquierda  ROLL
99       s62.write(AngulosRecibidos[13]);  //derecha
100      }
101 }
102
103 void EnviarDatos(void)
104   {
105     //Se utiliza para enviar datos al Maestro
106     //Codigo
107   }
```

# Bibliografía


[1] AROCHE, O. N. Modelo Cinemático y Dinámico de un robot bípedo con doce grados de libertad internos. Tesis, Universidad Nacional Autónoma de México, Ciudad de México, 2010.

[2] ARTBOT. Une étude de la marche dans le but de concevoir un robot bipède. Accedido en 12/01/2017 a http://sme-chinoises-euronext.typepad.fr/artbot/2014/08/une-%C3%A9tude-de-la-marche-dans-le-but-de-concevoir-un-automate-bip%C3%A8de.html, Agosto 2014.

[3] BATIKAN E. DEMIR, R. B., AND DURAN, F. real-time trajectory tracking of an unmanned aerial vehicle using a self-tuning fuzzy proportinal integral derivative controller. *International Journal of Micro Air Vehicles* (2016).

[4] BEGUM MUTLU, E. A. S., AND NEFESLIOGLU, H. A. A defuzzification-free hierarchical fuzzy system (df-hfs): Rockmass rating prediction. *Science Direct* (2016).

[5] BORELLI, G. De motu animalium. Accedido en 19/12/2016 a https://www.nlm.nih.gov/exhibition/horse/hor_borelli_mechanics2.html, 1680.

[6] BRADSHAW, R. R. H. J. M., AND FORD, K. M. Beyond Asimov: the three laws of responsible robotics. *IEEE INTELLIGENT SYSTEMS* (2009), 14–20.

[7] CHING-LONG, S. W. A., AND ZHU, G. Y. Fuzzy logic force control for a biped robot. *IEEE lnternational Symposium on Intelligent Control* (1991).

[8] CHRISTINE CHEVALLEREAU, G. B. G. A., AND AOUSTIN, Y. *Bipedal Robots: modeling, design and building walking robots*, 1 ed. ISTE, WILEY, 2009.

[9] CIENCIA, X. El origen de la palabra robot. Accedido en 20/12/2016 a https://www.xatakaciencia.com/robotica/el-origen-de-la-palabra-robot, Enero 2006.

[10] CRAIG, J. J. *Introduction to Robotics*, 3 ed. Pearson Prentice Hall, 2005.

[11] CRUZ, P. P. *Inteligencia artificial con aplicaciones a la ingeniería*, 1 ed. Alfaomega, Del Valle México D.F., Julio 2010.

[12] DEHGHANI, A., AND KHODADADI, H. Fuzzy logic self-tuning pid control for a single-link flexible joint robot manipulator in the presence of uncertainty. *15th International Conference on Control, Automation and Systems* (October 2015).

[13] DEKKER, M. H. P. Zero moment point method for stable biped walking. Internship report, Eidhoven, University of Technology, Netherlands, 2009.

[14] DERNNCOURT, F. Introduction to fuzzy logic. *Massachusetts Institute of Technology* (2013).

[15] GARRIDO, A. A brief history of fuzzy logic. *BRAINotes* (2012).

[16] GÓMEZ, J. G. Conjuntos y sistemas difusos (lógica difusa y aplicaciones). *Departamento de Lenguajes y Ciencias de la Computación Universidad de Málaga* (2005).

[17] GOGANI, A. M. O. F. M. B., AND POURGHOLI, M. Fuzzy gain scheduling of pid controller implemented on real time level control. *Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)* (2015).

[18] GOMEZ, E. Z. Curso de lógica difusa y controladores difusos - hackeando tec. Accedido en 10/03/2017 a https://www.youtube.com/channel/UCCFT304Nz5pSWktsiidUy0Q, 2017.

[19] HERRMANN, M. Deep learning and everything. Robotics and systems, University of Edinburgh, School of Informatics, Edinburgh, 2016.

[20] HOY, C. Sistema de músculos artificiales para los robots y las prótesis. Accedido en 21/12/2016 a http://computerhoy.com/noticias/life/sistema-musculos-artificiales-robots-protesis-48036, Julio 2016.

[21] J. A. DUNLOP, K. J. BURNHAM, D. J. G. J., AND KING, P. J. Application of fuzzy logic based self tuning control. *IEEE* (1995).

[22] JUNG-YUP KIM, I.-W. P., AND OH, J.-H. Experimental realization of dynamic walking of the biped humanoid robot khr-2 using zero moment point feedback and inertial measurement. *Advanced Robotics, Vol. 20, No. 6, pp. 707–736 (2006)* (2006).

[23] JUNG-YUP KIM, I.-W. P., AND OH, J.-H. Walking control algorithm of biped humanoid robot on uneven and inclined floor. *Journal of intelligent and Robotic Systems* (2007).

[24] LAS GAVETAS DE MI ESCRITORIO, D. Referencias anatómicas para describir el cuerpo. Accedido en 12/01/2017 a http://gavetasdemiescritorio.blogspot.mx/2013/03/referencias-anatomicas-para-describir.html, Marzo 2013.

[25] LEE, C. G., AND KIM, S. J. The development of an on-line self-tuning pid controller using fuzzy logic. *IEEE* (1995).

[26] LHAR, D. Saffir. Accedido en 20/12/2016 a http://www.electronicproducts.com/Electromechanical_Components/Motors_and_Controllers/Designing_a_firefighting_humanoid_robot.aspx, 2012.

[27] LUTENBERG, A., AND CRUZ, J. M. Introducción general a los sistemas embebidos. Simposio Argentino de Sistemas Embebidos, 2012.

[28] MAMDANI, E. H., AND BAAKLINI, N. Prescriptive method for deriving control policy in a fuzzy-logic controller. *IEEE* (1975).

[29] MORALES, M. F. M. Planeación de trayectorias de un robot Bípedo con un modelo parametrizado carro-mesa. Tesis, Universidad Nacional Autónoma de México, Ciudad de México, 2016.

[30] MÁRQUEZ, S. H. Renovación y mejoramiento de actuadores e instrumentación de un robot bípedo. Tesis, Universidad Nacional Autónoma de México, Ciudad de México, 2016.

[31] OF NAVAL RESEARCH, O. Shipboard autonomous fireighting robot (SAFFiR). Accedido en 21/12/2016 a http://www.onr.navy.mil/en/Media-Center/Fact-Sheets/Shipboard-Robot-Saffir.aspx, 2011.

[32] ORLICKÝ, J. P. L., AND DÚBRAVSKÁ, M. Self tuning fuzzy pid controller. *International Conference on Process Control* (June 2013).

[33] PALMA, R., AND STUTZ, C. Open loop dynamic trajectory generator for a fuzzy gain scheduler. *Science direct* (2003).

[34] PAPPIS, C. P., AND MAMDANI, E. H. A fuzzy logic controller for a trafc junction. *IEEE* (1977).

[35] RAVIRAJ, V. S. C., AND SEN, P. C. Comparative study of proportional-integral, sliding mode and fuzzy logic controllers for power converters. *IEEE* (1995).

[36] RAY, K. S., AND MAJUMDER, D. D. Fuzzy logic control of a nonlinear multivariable steam generating unit using decoupling theory. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS* (1984).

[37] RIOJA, U. A. J. Equilibrio dinámico de un robot bípedo utilizando el modelo simplificado carro-mesa. Tesis, Universidad Nacional Autónoma de México, Ciudad de México, 2014.

[38] SAAVEDRA, M. R. A. *Stable locomotion of humanoid robots based on mass concentrated model*. PhD thesis, Universidad Carlos III de Madrid, Leganés, October 2008.

[39] SADEGHIAN, R., AND MASOULE, M. T. An experimental study on the pid and fuzzy-pid controllers on a designed two-wheeled Self-Balancing autonomous robot. *4th International Conference on Control, Instrumentation, and Automation (ICCIA)* (2016).

[40] SALAS, F. G. Sistemas difusos jerárquicos para modelado y control. Tesis, Centro de investigación y de estudios avanzados del Instituto Politéctnico Nacional, Ciudad de México, 2005.

[41] SIMOES, M. G. Introduction to fuzzy control. *Colorado School of Mines*.

[42] SÁNCHEZ, A. E. Desarrollo de una interfaz de control para un robot bípedo tipo scout. Estancia Académica de verano, Universidad Nacional Autónoma de México, Facultad de ingeniería, 2017.

[43] SOLANO, D. Desarrollo de una interfaz de control para un robot bípedo tipo scout. Estancia Académica de verano, Universidad Nacional Autónoma de México, Facultad de ingeniería, 2017.

[44] SOTO, J. M. V. S. R., AND LLAMA, M. Stable fuzzy self tuning pid control of robot manipulators. *IEEE International Conference on Systems Man and Cybernetics* (October 2009).

[45] STENGELE, F. H. Diseño y construcción de prototipo neumático de prótesis de pierna humana. Tesis, Universidad de las Americas Puebla, Cholula Puebla, mayo 2008.

[46] TU, K.-Y., AND LEE, T.-T. Design of a multi-layer fuzzy logic controller using pole assignment for bipedal walking at varying speeds. *Journal of the Chinese Institute of Engineers* (2004), 55–68.

[47] VICTOR MANUEL HÉRNANDEZ GUZMÁN, RAMON SILVA ORTIGOZA, Y. R. V. C. S. *CONTROL AUTOMÁTICO Teoría de diseño, construcción de prototipos, modelado, identificación y pruebas experimentales*, 1 ed. Colección CIDETEC, 2013.

[48] WIKIPEDIA. Embedded system — wikipedia, the free encyclopedia. Accedido en 17/04/2017 a https://en.wikipedia.org/w/index.php?title=Embedded_system&oldid=775529063, 2017.

[49] WILMART, R. Theorical and practical definition of the zmp safety zone. Tesis, Faculté Polytechnique de Mons, Bruxelles, 2013.

[50] YESIL, E., AND GUZAY, C. A self tuning fuzzy pid controller design using gamma aggregation operator. *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (July 2014).

[51] ZADEH, L. A. Fuzzy sets. *INFORMATION AND CONTROL 8, 338, 353* (1965).

[52] ZADEH, L. A. Fuzzy algorithms. *INFORMATION AND CONTROL 12, 94, 102* (1968).

[53] ZADEH, L. A. Fuzzy logic and approximate reasoning. *Springer* (1975).

[54] ZADEH, L. A. Fuzzy logic—a personal perspective. *ScienceDirect* (2015).