Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN

Escuela Politécnica superior de Jaén

Trabajo Fin de Grado

CONTROL DE UN ROBOT HEXÁPODO

Alumno: Miguel Gómez Díaz

Tutor: Prof. Dña. Silvia María Satorres Martínez

Dpto: Ingeniería Electrónica y Automática

Abril, 2019

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Ingeniería Electrónica y Automática.

Doña Silvia María Satorres Martínez , tutora del Proyecto Fin de Carrera titulado: Control de un Robot Hexápodo, que presenta Miguel Gómez Díaz, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Marzo de 2019

El alumno: Los tutores:

MIGUEL GÓMEZ DÍAZ SILVIA M. SATORRES MARTÍNEZ

Índice

RESUMEN xiii
ABSTRACT xiii
1 INTRODUCCIÓN 1
1.1 Justificación 1
1.2 Objetivos del proyecto 2
1.3 Metodología 2
1.4 Estructura de la memoria 3
2. ESTADO DEL ARTE 5
2.1 Por qué robots hexápodos 5
2.1.1 Movimiento omnidireccional 5
2.1.2 Modos de caminar 7
2.1.3 Estabilidad 7
2.1.4 Utilización de las patas 9
2.2 Inspiración biológica 9
2.3 Actualidad de los robots hexápodos 10
2.3.1 Según el tipo de desempeño 11
2.3.1.1 Hobbistas 11
2.3.1.2 Enseñanza y divulgación robótica 12
2.3.1.3 Entornos peligrosos 13
2.3.1.4 Industria 15
2.3.2 Investigaciones actuales 15
3. HARDWARE 17
3.1. Hardware mecánico 17
3.1.1 Estudio de mercado 17
3.1.1.1 Búsqueda de piezas 18
3.1.1.2 Búsqueda de servomotores 22
3.1.1.3 Búsqueda de kits completos 28
3.1.2 Selección de materiales 31
3.1.2.1 Elección del pack 31
3.1.2.2 Modelo 2 33
3.2. Hardware electrónico 34

3.2.1 Controlador principal 34
3.2.1.1 Arduino 34
3.2.1.1.1 Características Atmel 328p 36
3.2.1.2 IDE Eclipse 37
3.2.2 Controlador de servomotores 38
3.2.2.1 Comunicación i2c 40
3.2.3 Mando Controlador 41
3.2.3.1 Mando PS2 por cable 41
3.2.3.2 Control sonoro 42
3.2.4 Control del encendido 43
3.2.5 Control de batería 46
3.2.5.1 Divisor de Tensión 47
3.2.5.2 LED RGB 48
3.2.6 Shield Arduino 49
3.2.6.1 Proceso de creación 50
3.2.6.2 Resultados 54
4. SOFTWARE 56
4.1. Situación espacial del robot 56
4.1.1 Patas y cuerpo 56
4.1.2 Resolución de ángulos en código C++ 59
4.2 Control de servomotores 60
4.2.1 Control por PWM 60
4.2.2 Calibración de posición 61
4.3 Desplazamiento del Robot 63
4.3.1 Aplicación cinemática inversa 64
4.3.1.1 Resolución ángulo teta 1 65
4.3.1.2 Resolución ángulo teta 2 69
4.3.1.3 Resolución ángulo teta 3 71
4.3.2 Matrices de transformación 73
4.3.3 Trayectorias 75
4.3.3.1 Fases de desplazamiento 75
4.3.3.1.1 Reposo 76
4.3.3.1.2 Avance 76

4.3.3.1.3 Empuje total ... 77
4.3.3.1.4 Avance total ... 77
4.3.3.2 Fases de giro ... 78
4.3.3.2.1 Reposo ... 78
4.3.3.2.2 Avance ... 79
4.3.3.2.3 Empuje total ... 79
4.3.3.2.4 Avance total ... 80
4.3.3.3 Parametrización ... 80
4.3.3.3.1 Interpolación de posiciones de servomotores ... 84
4.3.3.3.2 Velocidad límite ... 85
4.3.3.4 Secuencias ... 87
4.3.3.4.1 Ripple ... 87
4.3.3.4.2 Tripod ... 89
4.3.3.4.3 Wave ... 91
4.4 Movimiento del cuerpo ... 93
4.5 Estados predefinidos ... 95
4.6 Comandos del mando ... 95
4.6.1 Modo Desplazamiento robot ... 96
4.6.2 Modo control del cuerpo ... 97
4.7 Flujograma del código ... 98
4.7.1 Inicio del robot ... 98
4.7.2 Bucle Loop ... 99
4.7.3 Cinemática ... 100
5. RESULTADOS ... 102
6. PRESUPUESTO ... 108
7. CONCLUSIONES Y LÍNEAS DE FUTURO ... 111
8. BIBLIOGRAFÍA ... 114
ANEXO 1: CINEMÁTICA ... 116
1. Cinemática en robótica ... 116
1.1 Matemáticas para la localización espacial ... 116
1.1.1 Representación de la posición ... 116
1.1.1.1 Sistema cartesiano de referencia ... 117
1.1.2 Representación de la orientación ... 117

1.1.2.1 Matrices de rotación 117
1.1.2.2 Composición de rotaciones 119
1.1.3 Matrices de transformación homogénea 120
1.1.3.1 Coordenadas y matrices homogéneas 120
1.1.3.2 Traslación y rotación 121
1.1.3.3 Rotación seguida de translación 122
1.1.3.4 Composición de matrices homogéneas 123
1.2 Cinemática inversa 124
1.2.1 Distintos métodos de resolución 125
1.2.1.1 Método geométrico 125
1.2.1.2 Resolución a partir de las matrices de transformación homogénea 125
1.2.1.3 Desacoplamiento cinemático 125
2. Trayectorias 126
2.1 Trayectorias punto a punto 126
2.1.1 Movimiento eje a eje 126
2.1.2 Movimiento simultáneo de ejes 127
2.2 Trayectorias coordinadas o isocronas 127
2.3 Trayectorias continuas 128
ANEXO 2: SERVOMOTORES 130
2 Partes de un servomotor 130
2.1 Tipos de servomotores 131
2.1.1 Dimensiones y potencia 132
2.1.2 Servos analógicos y digitales 133
2.2 Control de servomotor 134
2.2.1 Modulación por ancho de pulso (PWM) 134
2.2.2 Flujograma de control 135
2.2.3 Puente H 136
ANEXO 3: ESQUEMÁ ELECTRÓNICO SHIELD PCB 140

Índice de Figuras

Figura 1: Movimiento omnidireccional 6
Figura 2: Obstáculos para un robot móvil 7
Figura 3: Ventaja de desplazamiento en terrenos irregulares 7
Figura 4: Robot dinámicamente estable 8
Figura 5: Robot estáticamente estable 8
Figura 6: Baricentro centrado con COM 8
Figura 7: Baricentro descentrado com COM 8
Figura 8: Nomenclatura de una pata 10
Figura 9: Distintos robots hexápodos hobbystas 11
Figura 10: Hexy robot 12
Figura 11: Vorpal The Hexapod 12
Figura 12: Asterisk agarrando un objeto 13
Figura 13: Asterisk trepando 13
Figura 14: Robot LAURON 14
Figura 15: SILO-6 14
Figura 16: Timberjack Walking Machine 15
Figura 17: PhantomX AX Metal Hexapod Mark III Kit 19
Figura 18: PhantomX AX Metal Hexapod Mark III 19
Figura 19: Hexapod 3DOF Phoenix Lynxmotion 20
Figura 20: Estructura aluminio Robo-Soul 21
Figura 21: Robo-Soul 18 DOF Silver Aluminium Hexapod 21
Figura 22: Robot robosoul CR-6 Hexapod 22
Figura 23: Hitec servo 23
Figura 24: GoTeck 24
Figura 25: Longruner 25
Figura 26: Lobot servo 26
Figura 27: Servo Dynamixel 27
Figura 28: Hexapod Phoenix Lynxmotion 28
Figura 29: Smaring Hexapod Robot 29
Figura 30: Robo soul LDX-218 30
Figura 31: Smaring Hexapod Robot 32
Figura 32: Robo soul LDX-218 32
Figura 33: Robo-soul 18 LDX-218 33
Figura 34: Arduino Uno R3 35
Figura 35:IDE oficial de Arduino 35
Figura 36: Eclipse para Arduino con plugin Sloeber 37
Figura 37: Controlador PCA 9685 38
Figura 38: Pads soldables 39
Figura 39: Conexionado comunicación i2C 39
Figura 40: Protocolo I2C 40
Figura 41: Conexión en serie de controladoras 41
Figura 42: Mando PS2 por cable 41
Figura 43: Patillaje conector PS2 42
Figura 44: Esquema buzzer 42
Figura 45: Buzzer 42

Figura 46: Mosfet de canal P 43
Figura 47: Características Mosfet IRF 5305P 43
Figura 48: Mosfet desactivado 44
Figura 49: Mosfect activado 44
Figura 50: Batería lipo 7.4V HRB 46
Figura 51: Cargador-balanceador de baterías Lipo 46
Figura 52: Esquema medida voltaje 47
Figura 53: Led RGB Batería 49
Figura 54: Proceso de desarrollo de PCB 50
Figura 55: Conector PS2 50
Figura 56: Huellas creadas para PCB 51
Figura 57: Lyout antes de enrutar 52
Figura 58: Lyout después de enrutar 53
Figura 59: Lyout finalizado 53
Figura 60: Renderizado en 3D 54
Figura 61: Shield PCB fabricada 54
Figura 62: Resultado final PCB 54
Figura 63: Esquema simplificado de una pata 56
Figura 64: Cuerpo robot parte trasera 57
Figura 65: Disposición de patas 57
Figura 66: Ángulos según función atan2 60
Figura 67: Giro vs ancho de pulso 61
Figura 68: Detalle engranaje servo 61
Figura 69: Ejemplo de trayectoria 63
Figura 70: Relación entre cinemática directa e inversa 64
Figura 71: Resolución θ1 Pata 1 A 65
Figura 72: Resolución θ2 Pata 1 B 65
Figura 73: Resolución θ1 Pata 2 A 66
Figura 74: Resolución θ1 Pata 2 B 66
Figura 75: Resolución θ1 Pata 3 A 67
Figura 76: Resolución θ1 Pata 3 B 67
Figura 77: Resolución θ1 Pata 4 68
Figura 78: Resolución θ1 Pata 5 68
Figura 79: Resolución θ1 Pata 6 69
Figura 80: Longitud característica X1 69
Figura 81: Diagrama lateral pata 70
Figura 82: Resolución θ2 70
Figura 83: Ángulo real fémur 71
Figura 84: Resolución θ3 71
Figura 85: Ángulo real tibia 72
Figura 86: Ángulo tibia adicional 72
Figura 87: Aplicación de Matriz de transformación homogénea 74
Figura 88: Fases de la trayectoria 75
Figura 89: Estado de reposo 76
Figura 90: Mitad de avance 76
Figura 91: Empuje del robot 77
Figura 92: Avance completo 77

Figura 93: Reposo 78
Figura 94: Mitad de avance 79
Figura 95: Empuje del cuerpo 79
Figura 96: Avance completo 80
Figura 97: Pasos en una trayectoria simple 81
Figura 98: Parámetros para fase de empuje y avance2 82
Figura 99: Grupos de patas ripple 87
Figura 100: Patrón Ripple 88
Figura 101: Secuencia Ripple 6 pasos 88
Figura 102: Grupos de patas tripod 89
Figura 103: Patrón tripod 89
Figura 104:Secuencia Tripod 6 pasos 90
Figura 105: Grupos de patas wave 91
Figura 106: Patrón Wave 91
Figura 107: Secuencia Wave 12 pasos 92
Figura 108: Traslación del cuerpo del robot 93
Figura 109: Rotación sobre el eje y 94
Figura 110: Rotación sobre el eje x 94
Figura 111: Rotación sobre el eje z 94
Figura 112: Configuración botones mando PS2 95
Figura 113: Flujograma del inicio del robot 98
Figura 114: Flujograma del bucle loop 99
Figura 115: Flujograma del cálculo de la cinemática 100
Figura 116: Altura máxima 102
Figura 117: Altura máxima de paso 103
Figura 118: Giro sobre eje Z 104
Figura 119: Giro sobre eje Y 104
Figura 120: Giro sobre eje X 105
Figura 121: Traslaciones del cuerpo 106
Figura 122: Relación entre cinemática directa e inversa 116
Figura 123: Vector en coordenadas cartesianas de 3 dimensiones 117
Figura 124: Ángulos de Euler: roll, pitch y yaw 119
Figura 125: Distintas soluciones geométricas para una misma orientación ... 125
Figura 126: Movimiento eje a eje 126
Figura 127: Movimiento simultaneo de ejes 127
Figura 128: Trayectoria coordinada 127
Figura 129: Trayectoria continua rectilínea 128
Figura 130: Componentes de un servomotor 130
Figura 131: Servo estándar 132
Figura 132: Fijaciones universales 132
Figura 133: Ancho de pulso 134
Figura 134: Orientación 134
Figura 135: Diagrama de control de un servomotor 135
Figura 136: Circuito puente H 136
Figura 137: Giro del motor al activar Mosfets alternos 136
Figura 138: Funcionamiento diodo flyback 137

Índice de Tablas

Tabla 1: Características servomotor HS-422 23
Tabla 2: Características servomotor GS-4060BB 24
Tabla 3: Características servomotor Longruner 25
Tabla 4: Características servomotor Lobot 26
Tabla 5: Características servomotor Dynamixel 27
Tabla 6: Características modelo Phoenix Lynxmotion 28
Tabla 7: Características Smaring Metal Hexapod Robot 29
Tabla 8: Características Robo-soul 18 LDX-218 30
Tabla 9: Elección de Modelo 32
Tabla 10: Características ATmega328 36
Tabla 11: Pines de comunicación mando-Arduino 42
Tabla 12: Tabla de verdad para activación mosfet 44
Tabla 13: Características de consumo de potencia IRF 5305P 45
Tabla 14: Características de batería LIPO 7.4V HRB 46
Tabla 15: Tabla puntos experimentales 48
Tabla 16: Distancias coxa 58
Tabla 17: Longitudes patas 58
Tabla 18: Coordenadas puntos coxa 58
Tabla 19: Coordenadas de los extremos de las patas 59
Tabla 20: Tabla de constantes PWM 63
Tabla 21: Parámetros de la trayectoria 1 81
Tabla 22: Secuencias Ripple 89
Tabla 23: Parámetros secuencias Tripod 90
Tabla 24: Parámetros secuencias Wave 92
Tabla 25: Comandos desplazamiento robot 96
Tabla 26: Comandos movimiento del cuerpo del robot 97
Tabla 27: Presupuesto PCB Shield Arduino 108
Tabla 28: Presupuesto componentes electrónicos 109
Tabla 29: Presupuesto hardware mecánico 109
Tabla 30: Presupuesto total 109

RESUMEN

El presente proyecto abarca tanto el montaje como el control de un robot hexápodo. Para ello se dispondrá de un kit básico de piezas más servomotores que conforman el robot. El control principal se realizará mediante Arduino, en concreto, un Arduino Uno R3. Se utilizarán, además, otros controladores externos específicos para los servomotores. El lenguaje utilizado para la programación ha sido C++.

El objetivo es desarrollar unos algoritmos de control del movimiento lo suficientemente logrados que doten de gran libertad de movimiento al robot, de manera que se pueda tener un control omnidireccional del mismo a través de un mando controlador.

El desarrollo de este proyecto ha implicado una investigación en la robótica y en el ámbito de la inspiración biológica que tienen éstos de los animales, mostrando qué ventajas suponen este tipo de robots frente a otros.

Finalmente, el resultado del proyecto podrá utilizarse como material en clases docentes y como ejemplos de aplicación de la robótica en jornadas de puertas abiertas y talleres. Igualmente, puede aplicarse para incentivar el acercamiento a la robótica y la ingeniería en general.

ABSTRACT

The present project covers the assembly and the control of a hexapod robot. For this, we will have a basic kit which includes 18 servomotors and parts for the assembly. The main control is implemented with Arduino, specifically, an Arduino Uno R3. However, to control that amount of servomotors we will have to use two external controllers specifically to servomotors. The language used for programming has been C++.

The objective is to develop control algorithms that provide the robot an omnidirectional movement. Also, it will be controlled through a controller.

The development of this project has involved research into robotics. Also the biological inspiration that robots have of animals, showing the advantages of this type of robots over others.

Finally, the result of the project can be used as material in teaching classes and as examples of application of robotics in days of open doors and visits. Likewise, it can be applied to encourage the approach to robotics and engineering in general.

1 INTRODUCCIÓN

Este Trabajo de Fin de Grado (TFG) tiene como objetivo el montaje y el control de un robot hexápodo. Para ello, se buscará plasmar todos los conocimientos y habilidades adquiridas durante el transcurso del grado en Ingeniería Electrónica Industrial. El controlador se basará en un microcontrolador Atmega 328p, que se programa mediante el lenguaje C++ y además forma parte de la plataforma Arduino (una plataforma de bajo coste y de código libre), la cual proporciona muchas facilidades de desarrollo.

Este proyecto servirá como material docente para clases, además de ejemplo de desarrollo en Jornadas de puertas abiertas y talleres.

Además, el presente TFG sirve de complemento al TFG “Sistema de navegación autónoma de robot hexápodo” del alumno José Antonio Guirado Cárdenas, que se centra en la navegación autónoma del robot a través de un sistema de visión por computador. Con estos proyectos se pretende dotar al robot de funcionalidad para moverse y además hacerlo inteligentemente para navegar en entornos parcialmente estructurados con y sin la presencia de obstáculos en los mismos.

1.1 Justificación

La motivación para la creación de este proyecto viene dada por la inquietud surgida a partir de la asignatura de robótica industrial. De todos los tipos de robots estudiados, los robots inspirados en animales suscitan un gran interés, ya que viendo como los animales han evolucionado para solucionar un problema u obstáculo, como pueda ser el hábitat y adaptarse para sobrevivir en él, es interesante ver cómo podemos hacer que los robots adquieran estas capacidades. De esta manera podemos inspirarnos en ellos y copiar o adaptar una o varias soluciones para solventar o mejorar la actuación del robot al problema en cuestión.

Se busca hacer una ampliación en el estudio de robots hexápodos, su morfología y movimiento. Por lo tanto se busca una adaptación robótica de las patas de los animales hexápodos y posteriormente el control de las mismas, así como las distintas formas de andar de animales hexápodos.

También busca aplicar y desarrollar todos los conocimientos aprendidos durante el transcurso de la carrera, tales como la capacidad de resolver problemas de manera autónoma, la mejora de las habilidades de programación en microcontroladores en lenguaje C++, o en el desarrollo y montaje de hardware electrónico.

Como proyecto de divulgación tecnológica y científica los dos trabajos de fin de grado anteriormente indicados, “Control de Robot Hexápodo” y “Navegación Autónoma de Robot Hexápodo”, tienen la finalidad de ser mostrados en posibles jornadas de puertas abiertas y talleres, ya que se le dotará al robot de distintas funcionalidades que pondrán de manifiesto algunas de las competencias que han sido adquiridas por los autores de los TFG a lo largo de la titulación realizada.

1.2 Objetivos del proyecto

Programar el Arduino en lenguaje C++ para proveer a los servos de capacidad para moverse conjuntamente, de forma ordenada e inteligente, permitiendo realizar movimientos en cada una de las extremidades. Dotar al robot de algoritmos de control que permitan un movimiento omnidireccional en entornos estructurados para realizar distintos modos de caminar estables y eficientes.

El proyecto no contempla la creación de las piezas que forman el robot debido a que ya había otro proyecto que abordó ese tema, con impresión 3D, por lo que en este caso las piezas se adquieren ya confeccionadas.

1.3 Metodología

El punto de partida es un estudio del estado del arte en materia de robots hexápodos creados anteriormente. En este sentido, analizamos las piezas que componen el esqueleto del robot, qué tipo de sistema de control utilizaban, los servomotores que empleaban y las ventajas e inconvenientes que mostraban cada solución. Posteriormente, se realiza una búsqueda de los componentes anteriormente citados, más atractivos y baratos para el objetivo que buscamos.

Una vez que tenemos todos los elementos del proyecto se procede al estudio en profundidad de la cinemática, programación y electrónica necesarias para dotar al robot de movimiento en cada una de sus patas, con la meta de moverse de forma coordinada para que el robot ande.

El siguiente paso es desarrollar la programación necesaria para dotar de movimiento a las patas y el montaje de las mismas; una vez se ha completado este paso se procede a hacer pruebas para verificar el movimiento y la correcta correlación entre lo esperado en los cálculos y lo obtenido en la realidad.

Una vez se han alcanzados los anteriores puntos de manera satisfactoria se procede a implementar el mando para variar los parámetros del robot a nuestro antojo y verificar su funcionamiento.

Por último se desarrollan los sistemas para el movimiento del cuerpo y los distintos modos de caminar y se implementan al robot para poder manejarlos con el mando.

Para finalizar, se recopila toda la información buscada y todo el contenido desarrollado, para realizar la memoria necesaria para finalizar el TFG y su posterior defensa ante el tribunal.

1.4 Estructura de la memoria

La estructura del proyecto está compuesta por siete capítulos, los cuales se detallan a continuación:

- En el capítulo 1 se realiza una introducción al proyecto, en la cual se expone la motivación del desarrollo de este proyecto, así como los objetivos a alcanzar en el mismo y la metodología para alcanzarlos.

- En el capítulo 2 se muestra el estado del arte respecto a los robots hexápodos, mostrando las motivaciones del desarrollo de este tipo de robots, la inspiración tomada de los animales y los robots hexápodos que hay actualmente.

- El capítulo 3 se explican las partes que forman en robot en sí, el hardware mecánico por un lado y el electrónico por otro.

- En el capítulo 4 se exponen todas las consideraciones a la hora de desarrollar el control del robot y posteriormente se explica su desarrollo e implementación en Arduino.

- En el capítulo 5 se muestran los resultados obtenidos con respecto al movimiento del robot.

- El capítulo 6 muestra el presupuesto total del robot, diferenciando entre partes mecánicas y electrónicas.

- En el capítulo 7 se presentan las conclusiones personales del proyecto y propuestas de mejora para el futuro.

- En el capítulo 8 se muestra la bibliografía recopilada en el estudio de este proyecto.

Finalmente tenemos tres anexos.

- En el anexo 1 se profundiza en la cinemática estudiada para poder realizar este TFG
-
- En el anexo 2 se explica más en profundidad las partes y el funcionamiento de los servomotores.
-

- El anexo 3 muestra el diagrama electrónico de la PCB Shield fabricada.

2. ESTADO DEL ARTE

En este capítulo se examinará por qué los robots caminantes hexápodos son motivo de desarrollo, y se investigará sobre los sistemas biológicos de locomoción. Por último, se aportarán ejemplos de otros robots hexápodos creados con anterioridad.

2.1 Por qué robots hexápodos

Uno de los motivos del desarrollo de los robots caminantes es que pueden salvar obstáculos mayores que sus equivalentes con ruedas; esto es verdad hasta cierto punto, ya que estos últimos se pueden fabricar más grandes de manera que puedan solventarlos y tienen la ventaja de ser más simples que ellos. Sin embargo, los robots andantes gozan de mayor grado de movilidad y pueden operar en zonas estrechas e irregulares, donde no es posible usar un robot de mayores dimensiones, o de transportarlos hasta dicha zona.

A medida que las condiciones de trabajo y los entornos se vuelven cada vez más peligrosos, es extremadamente conveniente eliminar al humano de estas situaciones potencialmente peligrosas y reemplazarlo con un robot diseñado específicamente para realizar una tarea determinada. Ejemplos de estas zonas son: Cuevas, minas, desastres naturales, bosques, barrancos, otros planetas etc… Los robots hexápodos son perfectos para estos lugares debido a las ventajas en su movilidad.

2.1.1 Movimiento omnidireccional

Una de las ventajas clave que gozan los robots andantes sobre los demás es que pueden moverse en cualquier dirección y girar en cualquier ángulo; esta capacidad de movimiento omnidireccional (

Figura *1*) es la que premia a este tipo de robots frente al resto.

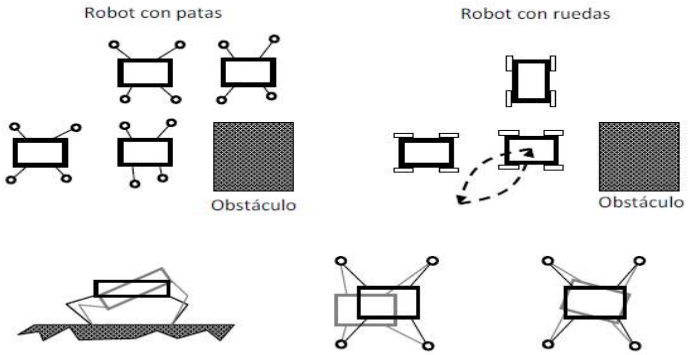


Figura 1: Movimiento omnidireccional

2.1.2 Modos de caminar

Otra de las ventajas es la capacidad de elección de varios modos de caminar y de poder levantar las patas del suelo en lugar de estar en continuo contacto con él. Este aspecto se ve beneficiado en ambientes sensibles, ya que es capaz de evitar pisar zonas peligrosas como agujeros, productos químicos, minas terrestres etc…

Usando esta precisión en los pasos, pueden lograr salvar puntos de apoyo para las patas que no son seguros, cuando están escalando o descendiendo desniveles, ventaja de la que no gozan otros tipos de robots como los que se desplazan mediante ruedas, como vemos en las siguientes figuras.

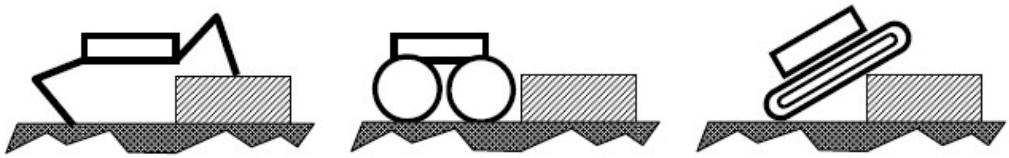


Figura 2: Obstáculos para un robot móvil

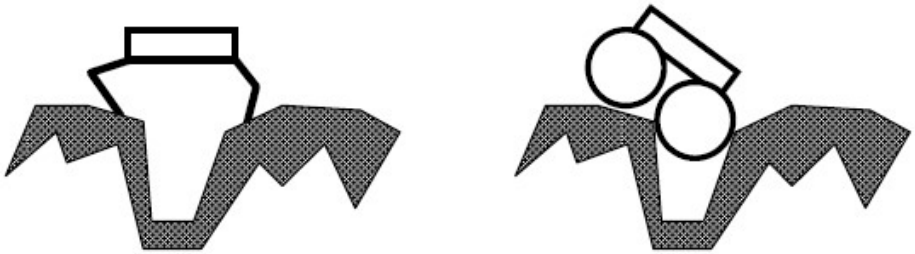


Figura 3: Ventaja de desplazamiento en terrenos irregulares

2.1.3 Estabilidad

Estas ventajas son compartidas por todos los robots caminantes, independientemente del número de patas que tengan. Sin embargo, los robots hexápodos o aquellos robots con más de seis patas (Figura 5) se pueden catalogar como **estáticamente estables**, mientras que los bípedos (Figura 4) y, hasta cierto punto, los cuadrúpedos, no lo son. La estabilidad estática es la capacidad de permanecer de pie sin estar continuamente cambiando la entrada de control y se logra manteniendo un mínimo de 3 patas apoyadas. Sin estabilidad estática, los robots deben ser **dinámicamente estables** o se desestabilizarán.

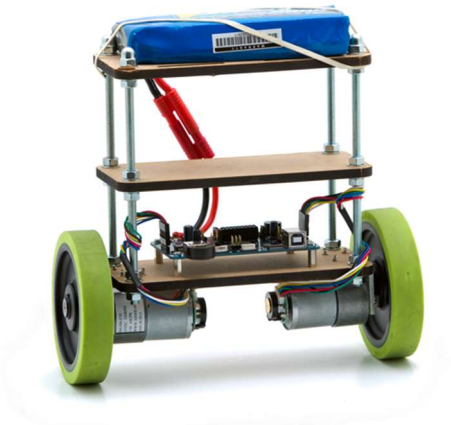


Figura 4: Robot dinámicamente estable

Figura 5: Robot estáticamente estable

Para que un robot hexápodo sea estable, ya sea parado o en movimiento, no puede haber dos patas consecutivas en movimiento o posadas al mismo tiempo. De esta manera, se consigue que el baricentro del triángulo formado por los tres vértices de las patas que tocan el suelo, coincida con el centro de masas del robot (Figura 6), haciendo que el robot se mantenga estable; mientras que en la Figura 7 estos no coinciden, produciéndose un desequilibrio.

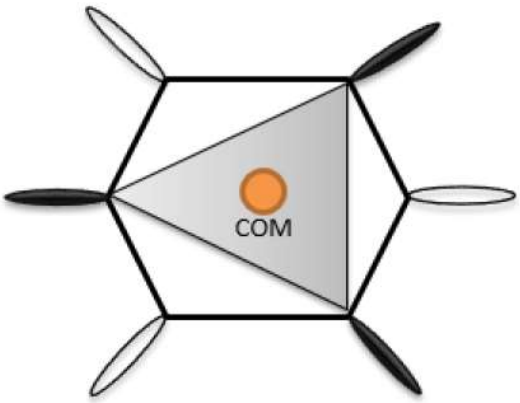


Figura 6: Baricentro centrado con COM

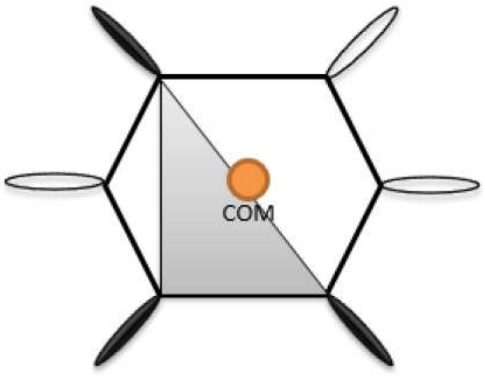


Figura 7: Baricentro descentrado com COM

2.1.4 Utilización de las patas

Las patas adicionales, por ejemplo ocho o diez, pueden mejorar la estabilidad a velocidades altas. No obstante, la estabilidad a partir de la sexta pata ya es muy notoria. Además de esto, tener más patas de apoyo permite a los robots cargar con más peso (hasta cierto número de patas).

También para la navegación autónoma en entornos exigentes, la virtud de tener numerosas patas otorga la capacidad de operar con patas dañadas o incapacitadas, algo que los insectos pueden hacer y que los robots cuadrúpedos o inferiores no son capaces. Por lo que seis patas es el balance perfecto entre estabilidad y número de patas.

2.2 Inspiración biológica

Los animales son la principal inspiración biológica en el desarrollo de la robótica. Son el resultado, dependiendo del punto de vista, de un billón de años de desarrollo evolutivo adaptándose cada especie de la mejor manera al hábitat en el que se encuentra.

Tampoco hay que caer en el extremo de calcar absolutamente todo el diseño biológico. Hay muchos enfoques intermedios útiles en los que uno se puede inspirar, como son:

- Supongamos que la estructura física de una especie en particular es óptima para su nicho ecológico, y por lo tanto imitar todo el diseño para un robot con requerimientos aparentemente similares.

- Modelar sólo la parte de interés (ej. Patas o cuerpo) de una especie en particular.

- Adaptar las capacidades sensoriales de los animales con tecnología actual, para ser capaz de interactuar con su entorno.

- Utilizar tecnologías adaptativas basadas en sistemas biológicos fundamentales (redes neuronales, algoritmos genéticos) y desarrollar un comportamiento.

El último enfoque es el más prometedor para el desarrollo de soluciones de ingeniería, pero por contrapartida requiere de un mayor trabajo actualmente. Al observar y entender los principios de por qué los organismos se comportan o crecen como lo hacen, los ingenieros podrían construir nuevas máquinas con propiedades "orgánicas" hasta ahora no vistas en el mundo natural.

Para este robot se ha adaptado la fisionomía de las patas con piezas y servomotores. Además se utilizará la misma nomenclatura para las patas que la que se utiliza en los estudios sobre insectos, como podemos ver en la siguiente figura.

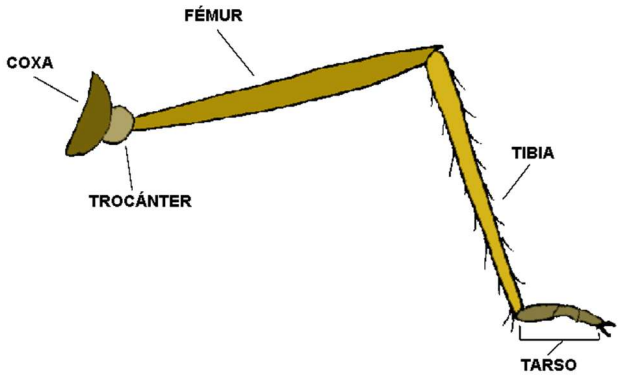


Figura 8: Nomenclatura de una pata

2.3 Actualidad de los robots hexápodos

En la actualidad, podemos encontrarnos robots hexápodos de todo tipo, los cuáles se pueden clasificar de muchas maneras, dependiendo del tipo de desempeño que vayan a realizar, del material del que estén fabricados, de su estructura, circular hexagonal, rectangular, etc… También dependiendo del tipo de actuadores que posean, o si son de software libre o cerrado, etc. se pueden encontrar numerosos robots con distintas características. Sin embargo, el factor determinante es el desempeño que vaya a realizar el robot, ya que no se utilizaran los mismos materiales o actuadores para un robot de tipo hobby, que un robot que se utilice para el rescate en un desastre natural.

2.3.1 Según el tipo de desempeño

2.3.1.1 Hobbistas

Con el avance actual de la robótica es normal encontrar soluciones sencillas que pueden ser fácilmente reproducibles por un determinado público. De esta idea surgen proyectos denominados DIY (do it yourself = hazlo por tí mismo). Destacar que en este ámbito puede haber personas que no sean profesionales del tema y solo quieran aprender (amateurs) o, por otra parte, personas con estudios más específicos o profesionales que quieran enfrentarse a un reto, mejorar sus habilidades o poner en práctica sus conocimientos. Por lo que encontraremos un gran abanico de materiales como plástico, fibra de vidrio, fibra de carbono o metal y actuadores tanto eléctricos, como hidráulicos o mecánicos, así como multitud de diseños y formas.

Para el marco del *hobbysta,* el software suele ser libre para que puedas aprender de él, aunque no todas las plataformas ofrecen el mismo soporte en el que te explican cómo funciona el proyecto. Cabe destacar, como experiencia, que cuanto más complejo es el proyecto menos soporte hay detrás, aunque el software sea libre, ya que no añaden ninguna explicación del funcionamiento, más allá de alguna guía de montaje y uso.

Aquí tenemos algunos modelos que parten de los 40 euros, como el KIKO.893, hasta más de los 1000 euros como el PhantomX AX Metal Hexapod Mark III.

a) Lynxmotion AH2

b) Lynxmotion CH3-R

c) Robot Hexápodo KIKO.893

d) PhantomX AX Metal Hexapod Mark III

Figura 9: Distintos robots hexápodos hobbystas

2.3.1.2 Enseñanza y divulgación robótica

Con el auge de la robótica existen empresas o plataformas que apoyan la enseñanza de la robótica y/o la electrónica con ejemplos de robots hexápodos. Un ejemplo de estas empresas es *Arbotics*, la cual promueve la robótica con el robot hexápodo Hexy de una manera sencilla, proporcionando guías no sólo de montaje de piezas, sino también de montaje electrónico e incluso de programación. *Arbotics* también realiza charlas explicativas sobre el funcionamiento de Hexy, promoviendo la robótica y la electrónica.

Figura 10: Hexy robot

Como vemos, Hexy es un robot básico formado por un sensor de ultrasonidos, 18 servomotores y piezas de plástico. Esto lo convierte en una solución muy accesible y barata para dar a conocer la robótica.

Vorpal robotics es otra empresa que con su robot hexápodo Vorpal proporciona similares características que el robot Hexy y además proporciona los archivos para imprimir en 3D las piezas que componen el robot, lo cual lo hace muy atractivo.

Figura 11: Vorpal The Hexapod

Además cabe destacar que este robot se mostró en El Hormiguero, un programa de la televisión española, donde mostraron su movimiento y algunos principios de su funcionamiento.

2.3.1.3 Entornos peligrosos

Entornos de rescate en catástrofes naturales, guerras e investigación de planetas o exploración de zonas desconocidas, son entornos en los que un robot hexápodo se puede desenvolver con soltura gracias a las grandes capacidades de movimiento que poseen y que los hacen posicionarse como una mejor opción frente a sus competidores directos, los robots con ruedas, relevándolos en un futuro próximo.

El robot Asterisk está siendo desarrollado por la universidad de Osaka (Japón) desde 2005 como un robot de búsqueda y rescate que puede ser usado en terremotos o tifones. Una de las características más útiles de este prototipo es que tiene dos caras, es decir, si se cae y queda patas para arriba entonces sus extremidades se plegarán hacia el otro lado y volverá a caminar. También tiene la capacidad de trepar por vallas u otras zonas con orificios, además de que sus patas son capaces de ajustarse para compensar un peso adicional en alguna parte del robot.

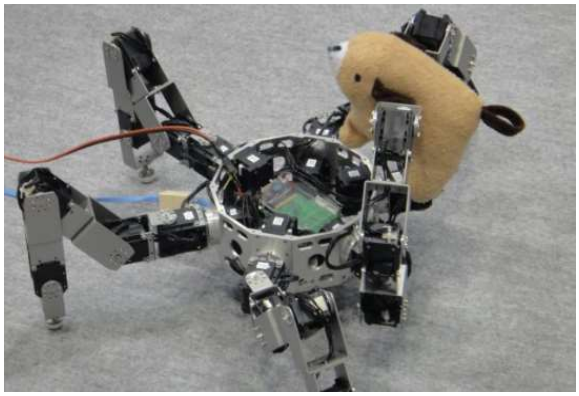


Figura 12: Asterisk agarrando un objeto

Figura 13: Asterisk trepando

LAURON es otro robot hexápodo que se desarrolló en el FZI (Centro de investigación para la tecnología de la información) y está biológicamente inspirado en el insecto palo Carausius Morosus. Lleva siendo desarrollado desde 1994 y ha ido evolucionando hasta la versión V que se presentó en 2013.

Este robot está provisto de numerosos sistemas de sensores, que pueden mapear en 3D el terreno que le rodea para navegar autónomamente e ir de un punto a otro que él mismo fija. Cada pata posee un sistema de sensores de presión y medición de corriente para detectar colisiones y saber cuándo está en contacto con el suelo.

El robot caminante LAURON es una de las mejores opciones para tareas de exploración y servicio en terrenos accidentados y no estructurados y áreas inaccesibles o peligrosas para los humanos. Dichas tareas podrían incluir el barrido de minas terrestres, la exploración de volcanes o las tareas de búsqueda y rescate después de desastres naturales.

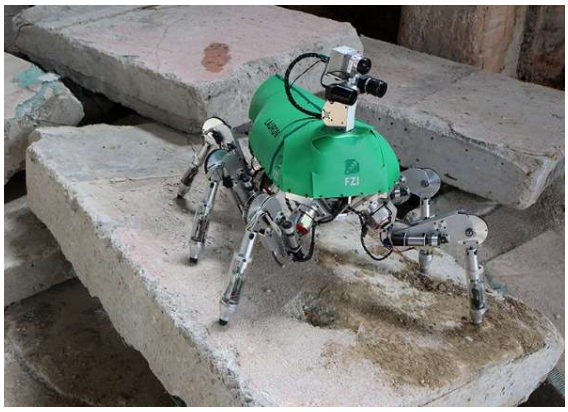


Figura 14: Robot LAURON

Existen aplicaciones para el ejército, ya sea en maniobras defensivas u ofensivas, que tratan sobre la creación de robots hexápodos para la desactivación de minas antipersona, con el fin de evitar las bajas de soldados y civiles.

El robot SILO-6 desarrollado en el CSIC, en la Politécnica de Madrid, es un robot de seis patas que puede buscar y desactivar minas. Posee un brazo adicional que le permite detectar y desactivar las minas y está compuesto de un detector de metales y sensores infrarrojos. Además, posee comunicación por WI-FI y GPS.

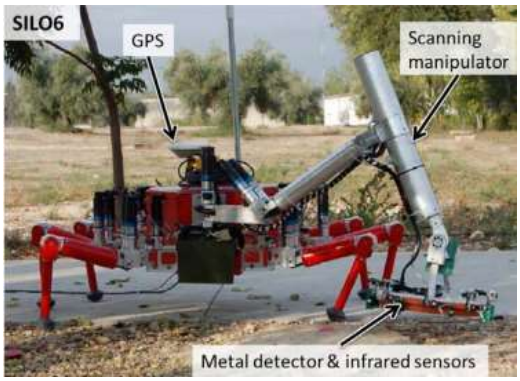


Figura 15: SILO-6

2.3.1.4 Industria

En la industria forestal vemos como surgen prototipos de hexápodos. Estos presentan una ventaja evidente en este tipo de terrenos. No obstante, estos todavía son lentos y toscos para su uso comercial.

Figura 16: Timberjack Walking Machine

2.3.2 Investigaciones actuales

Con el surgimiento de la inteligencia artificial, hay grupos de desarrollo que están intentando aplicar el *machine learning* para mejorar la forma de andar del robot y que ésta se adapte según el terreno en el que se encuentre o a la pérdida de alguna extremidad.

No obstante, estas investigaciones todavía están lejos de dar un resultado satisfactorio, por lo que todavía hay mucho que desarrollar en este ámbito.

3. HARDWARE

En este capítulo se explicarán las partes físicas de las que se compone el robot. Comenzando con las piezas que conforman su estructura física, los actuadores que dotarán de movimiento al robot, terminando con la electrónica que lo controlará y dotará de movimiento.

3.1. Hardware mecánico

Las dos partes que forman el hardware mecánico de este proyecto son las piezas que componen la estructura o armazón del robot y los servomotores que en este proyecto toman el rol de las articulaciones de las patas.

Con anterioridad otro compañero realizó en su TFG un diseño 3D de las piezas que componen un robot hexápodo que, posteriormente, fabricó con una impresora 3D, como ya se ha dicho antes. Para diferenciar este proyecto del realizado con anterioridad, el esfuerzo se focalizó en mejorar de manera notoria las posibilidades para el control del mismo y se ha optado por la compra de un kit de piezas sobre el cual se trabajará.

De esta forma podemos conseguir unas piezas que se adapten mejor a las exigencias que buscamos para el robot, haciendo una selección de los materiales y formas estructurales más adecuados para nuestro caso. Esto es un punto a favor frente a la impresión 3D ya que, en ésta, la calidad de los objetos depende de la calidad de la impresora y del material que se utilice, teniendo un coste mayor cuanto mejores sean las características.

Para el movimiento de las patas de nuestro robot hexápodo necesitaremos unos actuadores que desempeñen el rol de articulaciones. Esta función la desempeñarán los servomotores, los cuáles nos permitirán crear toda clase de movimientos de una forma controlada.

3.1.1 Estudio de mercado

En esta sección se va a estudiar las características más relevantes del hardware que queremos adoptar, las piezas estructurales y los servomotores. Obviamente, el costo de cada elemento es el pilar fundamental de este estudio, por lo que es lo primero que se tendrá en cuenta**.**

- **Requerimientos de las piezas**: Nuestra búsqueda se centrará principalmente encontrar unas piezas cuyo material de fabricación confiera unas propiedades de ligereza y resistencia adecuadas para el desempeño a realizar y cuyo diseño favorezca la robustez y estabilidad del robot. Los materiales más comunes para este tipo de robots son fibra de vidrio o aluminio, aunque hay modelos de muy bajo presupuesto que se componen de acrílico.

- **Requerimientos de servomotores**: Buscaremos un compromiso entre la velocidad del actuador y el torque proporcionado. También valoraremos la exactitud en la posición, así como el rango de movimiento que posee.

En el mercado se pueden encontrar por separado piezas y servomotores, la cuestión está en realizar un estudio donde discernir si la obtención de los materiales debe hacerse de forma separada o por el contrario es mejor comprar un kit que contenga ambas partes. No obstante, este estudio también busca mostrar la variedad y las características de los elementos que componen este proyecto y que se pueden encontrar por internet.

El modo de estudio será el siguiente: primero se analizará los kits de piezas disponibles en el mercado y en segundo lugar los servomotores, para hacer un estudio de los componentes existentes y ver si existe alguna combinación que resulte más barata que comprar un kit conjunto de piezas más servos.

3.1.1.1 Búsqueda de piezas

Para las piezas, las principales webs de búsqueda van a ser Trossenrobotics, Roboshop y Aliexpress, debido a que robots hexápodos no se encuentran fácilmente. Las dos primeras son webs específicas de robótica y electrónica mientras que la tercera es una web china que vende todo tipo de productos y su principal ventaja es la agresividad de los precios. En los siguientes puntos analizaremos los productos de cada tienda online.

- Trossenrobotics

Esta web vende un kit completo de piezas, servomotores, controlador y mando, pero se pueden adquirir los elementos por separado, siendo los elementos mecánicos los mostrados en la siguiente ilustración.

Estas piezas están fabricadas en aluminio, lo que las hace bastante resistentes a la par que ligeras.

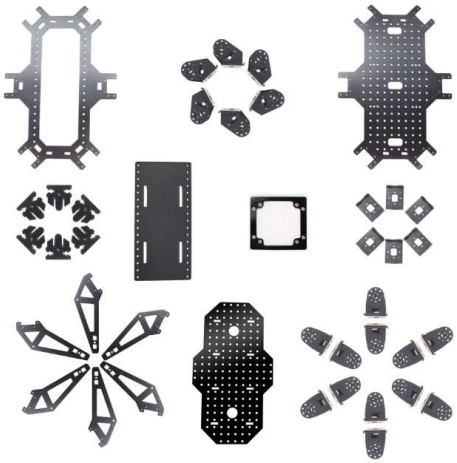


Figura 17: PhantomX AX Metal Hexapod Mark III Kit

Podemos ver en la ilustración inferior cómo la pata alcanza una estructura muy conseguida. Esto es debido, en parte, al tipo de servomotores que incorpora; éstos tienen un eje que no sólo sobresale solo por un lado, sino que lo hace por los dos lados. Esto hace que las patas tengan dos puntos de anclaje con el eje en lugar de uno como ocurría en el caso anterior, aumentando la rigidez de la pata. A este tipo de estructura la calificaremos como la más estable y rígida vistos los resultados y a la de un solo anclaje como de menos capaces.

Las patas se dispondrán en los vértices del hexágono que forma el cuerpo central; hay otras formas de cuerpo a parte de ésta, como la circular o la rectangular, pero las más usadas son las dos primeras puesto que consiguen una mayor estabilidad estática.

Figura 18: PhantomX AX Metal Hexapod Mark III

En el apartado siguiente se estudiarán más en profundidad los servomotores, pero se puede observar dos tipos de estructuras: las que tienen dos puntos de anclaje y las que solo tienen uno, dependiendo del tipo de servomotores para el que estén pensadas acoplarse.

Por último, y más importante, este kit de piezas lo podemos encontrar por 229.95 dólares, lo cual es muy caro sabiendo que solamente se considera la estructura. Además, las piezas no son universales, es decir, que no las podemos usar con otros servos estándar que no sean los que están pensados para este modelo, que son de la marca Dynamixel, los cuáles también son muy caros. Por todo esto, esta opción la descartaremos pero, aunque sea de manera informativa y divulgativa, en el siguiente apartado analizaremos estos servomotores Dynamixel.

- Robotshop

En esta página encontramos muchos modelos de robots hexápodos, ya sean de cuerpo circular, rectangular o hexagonal y además con varias configuraciones de anclaje de servomotores. Pero, a la hora de buscar kits de piezas solas, sólo encontramos el modelo Hexapod 3DOF Phoenix Lynxmotion

Este kit se compone de piezas de aluminio, que comparten las mismas bondades que el modelo de Trossenrobotics, un cuerpo con forma hexagonal y unas piezas de anclaje de servomotores con sólo una sola fijación al eje, por lo que con respecto a las patas estas serían menos rígidas y por lo tanto menos estables que el modelo de Trossenrobotics, aparte de limitarnos a usar servos de esta tipología.

Figura 19: Hexapod 3DOF Phoenix Lynxmotion

Respecto al precio 264.19 euros sin incluir controladora de servomotores, ni los propios servomotores, lo hace extremadamente caro.

- Aliexpress

Lo principal que destacamos de esta web es que encontramos precios mucho más agresivos, aparte tenemos para elegir estructuras de los dos tipos, dependiendo del tipo de anclaje con los servomotores. Vamos a estudiar dos modelos, uno de cada tipo.

1) Modelo con un solo anclaje

Aquí tenemos un ejemplo de un modelo con estructura de aluminio, el diseño y material de fabricación de las piezas es prácticamente el mismo que el modelo de Roboshop.

Figura 20: Estructura aluminio Robo-Soul

Figura 21: Robo-Soul 18 DOF Silver Aluminium Hexapod

El modelo resulta prácticamente igual que la versión de la web anterior, pero la diferencia se encuentra en el precio, ya que hay modelos de esta tipología que rondan los 50-60 euros

2) Modelo de doble anclaje

Por último existen otros hexápodos que tienen las piezas de fibra de vidrio, más barato que las piezas de aluminio. Este material no es tan rígido como el aluminio, pero para este tipo de aplicaciones da un muy buen resultado.

Respecto al modelo en concreto, este está fabricado con piezas de aluminio y fibra de vidrio; los elementos que conectan unos servomotores con otros están fabricados en aluminio, mientras que el cuerpo y el extremo final de la pata son de fibra de vidrio. El cuerpo posee un diseño hexagonal, muy común como hemos visto en todos los modelos anteriores, además este modelo posee un diseño similar al primer kit analizado de Trossenrobotics.

Figura 22: Robot robosoul CR-6 Hexapod

La estructura de anclaje con los servomotores tiene dos sujeciones, lo cual es un punto a favor. No obstante, el resultado no es tan bueno como en el primer modelo analizado, y aquí la agresividad con el coste de este kit juega un papel fundamental, ya que se encuentra por unos 60-70 euros.

3.1.1.2 Búsqueda de servomotores

Las piezas analizadas en el apartado anterior poseen anclajes con los servomotores (ya sean de uno o dos anclajes por eje) de dimensiones estándar; un servo de dimensiones estándar tiene las siguientes dimensiones: 40 x 20 x 37 mm. Pueden variar un poco dependiendo del modelo, pero generalmente las medidas de los orificios de fijación y la posición del eje de salida coinciden en casi todos los modelos.

Para los servomotores hay multitud de webs para su compra, además de una gran cantidad de modelos y tamaños. Debido a esto analizaremos sólo los más relevantes y que sean totalmente compatibles en dimensiones con las estructuras mencionadas anteriormente.

- **Modelos con un anclaje por eje**

Modelo	**HS-422**	
Fabricante	 *Figura 23: Hitec servo*	
Descripción	Servomotor de propósito general, con torque medio-bajo y 180º de movimiento	
Características generales	Peso	45.5 g
	Dimensiones	41*20*37mm
	Cable conector	30 cm
	Velocidad	0.16sec/60°(7.4V)
	Precisión	No especificado
	Torque	3.3 kg·cm / 4.8V; 4.1 kg·cm / 6V
	Rango de rotación	180º
	Diámetro eje	6mm
Características eléctricas	Voltaje de trabajo	4.8-6 V
	Máx Corriente de trabajo	No especificado
	Corriente sin carga	150mA
Control	Método de control	Ancho de pulso (PWM)
	Ancho de pulso	1000us~2000us
	Periodo pulsos	20ms (50Hz)
Precio	12.29 €	

Tabla 1: Características servomotor HS-422

Modelo	**GS-4060BB**	
Fabricante	 *Figura 24: GoTeck*	
Descripción	Servomotor de propósito general, con torque medio y 180º de movimiento	
Características generales	Peso	41 g
	Dimensiones	41*20*38mm
	Cable conector	30 cm
	Velocidad	0.16sec/60°(7.4V)
	Precisión	No especificado
	Torque	6.3 kg·cm / 4.8V; 7 kg·cm / 6V
	Rango de rotación	180º
	Diámetro eje	6mm
Características eléctricas	Voltaje de trabajo	4.8-7 V
	Máx Corriente de trabajo	1.2A/ 4.8V; 1.3A/6V
	Corriente sin carga	200-250 mA/ 4.8-6V
Control	Método de control	Ancho de pulso (PWM)
	Ancho de pulso	1000us~2000us
	Periodo pulsos	20ms (50Hz)
Precio	13.85 €	
Notas	Datasheet: http://www.goteckrc.com/Download/GS-4060BB.pdf	

Tabla 2: Características servomotor GS-4060BB

- **Modelos con dos anclajes por eje**

Modelo	**Longruner Digital 17kg LDX 218**	
Fabricante	 *Figura 25: Longruner*	
Descripción	Servomotor de propósito general, con gran torque y 180º de movimiento	
Características generales	Peso	181 g (todo el producto)
	Dimensiones	40*20*40.5mm
	Cable conector	30 cm
	Velocidad	0.16sec/60°(7.4V)
	Precisión	No especificado
	Torque	15 kg·cm / 6.6V; 17 kg·cm / 7.4V
	Rango de rotación	180º
	Diámetro eje	6mm
Características eléctricas	Voltaje de trabajo	6.6-7.4 V
	Máx Corriente de trabajo	No especificado
	Corriente sin carga	150mA
Control	Método de control	Ancho de pulso (PWM)
	Ancho de pulso	500us~2500us
	Periodo pulsos	20ms (50Hz)
Precio	18.99 €	
Nota	Incluye anclajes de aluminio	

Tabla 3: Características servomotor Longruner

Modelo	**LDX-218**	
Fabricante	 *Figura 26: Lobot servo*	
Descripción	Servomotor de propósito general, con gran torque y 180º de movimiento	
Características generales	Peso	60 g
	Dimensiones	40*20*40.5mm
	Cable conector	30 cm
	Velocidad	0.16sec/60°(7.4V)
	Precisión	0.3°
	Torque	15 kg·cm / 6.6V; 17 kg·cm / 7.4V
	Rango de rotación	180º, además se puede rotar 360º cuando no hay alimentación
	Diámetro eje	6mm
Características eléctricas	Voltaje de trabajo	6-8.4V
	Máx Corriente de trabajo	1 A
	Corriente sin carga	100mA
Control	Método de control	Ancho de pulso (PWM)
	Ancho de pulso	500us~2500us
	Periodo pulsos	20ms (50Hz)
Precio	11.22 €	
Nota	Este servo utiliza una falsa doble sujeción con el eje, ya que la segunda sujeción no tiene conexión ninguna con el eje del motor; ésta se utiliza para que las estructuras se fijen a ella y puedan girar sin restricciones, lo cual es una buena solución para mejorar la rigidez de los montajes sin que esto suponga un gran sobrecoste.	

Tabla 4: Características servomotor Lobot

Modelo	Dynamixel AX-12W	
Fabricante	 *Figura 27: Servo Dynamixel*	
Descripción	Servomotor de propósito general, con gran torque y 180º de movimiento	
Características generales	Peso	55 g
	Dimensiones	32 x 50 x 40 mm
	Cable conector	20 cm
	Velocidad	0.169sec/60°
	Precisión	0.29°
	Torque	15.3 kg·cm / 12V
	Rango de rotación	300° o 360º cuando no hay alimentación
	Diámetro eje	6mm
Características eléctricas	Voltaje de trabajo	9-12V (Voltaje recomendado 11.1V)
	Máx Corriente de trabajo	900 mA
	Corriente sin carga	50 mA
Control	Método de control	Ancho de pulso (PWM)
	Ancho de pulso	No especificado
	Periodo pulsos	No especificado
Precio	44.9 dólares	
Nota	Estos servos se controlan mediante un protocolo de comunicación TTL: TTL Half Duplex Async Serial	

Tabla 5: Características servomotor Dynamixel

Los modelos de servomotores Dynamixel, que son los empleados en los hexápodos de Trossenrobotics, no se contemplarán puesto que se pasan desorbitadamente de presupuesto, costando el modelo más básico 20 dólares, lo cual hace inviable comprar 18 de ellos. Por lo que previsiblemente se descartará comprar cualquier componente de esta página.

3.1.1.3 Búsqueda de kits completos

De la página web Roboshop sólo analizaremos un producto, que dentro de la variedad que nos brindaba esta página, he considerado como el más interesante a destacar.

<table>
<tr><td>Modelo</td><td colspan="3">Robot Hexapod Phoenix Lynxmotion</td></tr>
<tr><td>Fotos</td><td colspan="3">  </td></tr>
<tr><td>Estructura</td><td colspan="3">Aluminio</td></tr>
<tr><td rowspan="7">Servomotores</td><td>Modelo</td><td colspan="2">Hitec HS-645MG</td></tr>
<tr><td rowspan="6">Características generales</td><td>Velocidad</td><td>0.20 sec/60°</td></tr>
<tr><td>Torque</td><td>9.6 kg.cm</td></tr>
<tr><td>Tamaño</td><td>41 x 20 x 38 mm</td></tr>
<tr><td>Peso</td><td>55.2 g</td></tr>
<tr><td>Rotación</td><td>180º</td></tr>
<tr><td>Precio por separado</td><td>30.93€</td></tr>
<tr><td>Precio</td><td colspan="3">806,67 €</td></tr>
<tr><td>Opinión</td><td colspan="3">El precio es desorbitado y no entra dentro de nuestro presupuesto, por lo cual no se contemplará</td></tr>
</table>

Figura 28: Hexapod Phoenix Lynxmotion

Tabla 6: Características modelo Phoenix Lynxmotion

De la web de Aliexpress analizaremos dos modelos, uno de cada tipo de anclaje con los servomotores

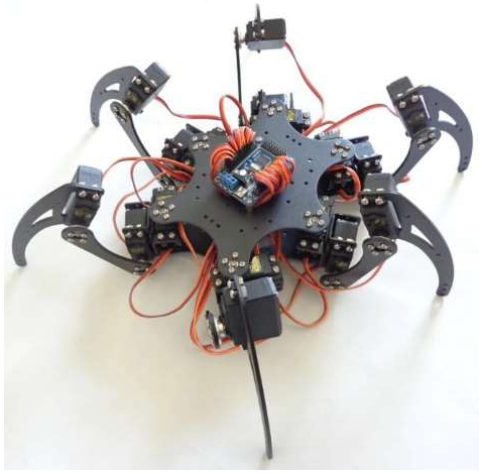
<table>
<tr><td>Modelo</td><td colspan="3">Smaring: 18 DOF Aluminio Metal Hexapod Robot</td></tr>
<tr><td>Fotos</td><td colspan="3"> 
Figura 29: Smaring Hexapod Robot</td></tr>
<tr><td>Estructura</td><td colspan="3">Aluminio</td></tr>
<tr><td rowspan="7">Servomotores</td><td>Modelo</td><td colspan="2">DS3218 Digital Servo</td></tr>
<tr><td rowspan="6">Características generales</td><td>velocidad</td><td>0.14sec/60º(6v) ó 0.16sec/60º (7.2v)</td></tr>
<tr><td>Torque</td><td>18kg.cm.at(6v) ó 20kg.cm.at(7.2v)</td></tr>
<tr><td>peso</td><td>60 g</td></tr>
<tr><td>Tamaño</td><td>40*20*40.5 mm</td></tr>
<tr><td>Rotación</td><td>180º</td></tr>
<tr><td>Precio por separado</td><td>11.05 €</td></tr>
<tr><td>Precio</td><td colspan="3">206.3 €</td></tr>
<tr><td>Opinión</td><td colspan="3">Este es uno de los mejores modelos encontrados, los servomotores que incorpora tienen un buen compromiso entre velocidad y torque, y la estructura de aluminio proporcionará una buena estabilidad. El tipo de estructura-servo es de la clase de un solo anclaje con el eje del servomotor.</td></tr>
</table>

Tabla 7: Características Smaring Metal Hexapod Robot

<table>
<tr><td>Modelo</td><td colspan="3">Robo-soul 18 LDX-218</td></tr>
<tr><td>Fotos</td><td colspan="3"> 
Figura 30: Robo soul LDX-218</td></tr>
<tr><td>Estructura</td><td colspan="3">Aluminio y Fibra de vidrio</td></tr>
<tr><td rowspan="7">Servomotores</td><td>Modelo</td><td colspan="2">Lobot ldx-218</td></tr>
<tr><td rowspan="6">Características generales</td><td>velocidad</td><td>0.16sec/60°(7.4V)</td></tr>
<tr><td>Torque</td><td>15 kg·cm /6.6V;17 kg·cm/7.4V</td></tr>
<tr><td>peso</td><td>55 g</td></tr>
<tr><td>Tamaño</td><td>40*20*40.5mm</td></tr>
<tr><td>Peso</td><td>60 g</td></tr>
<tr><td>Precio por separado</td><td>11.22€</td></tr>
<tr><td>Precio</td><td colspan="3">217€</td></tr>
<tr><td>Opinión</td><td colspan="3">La ventaja de este kit es que las piezas de las patas poseen dos anclajes para la sujeción con el servomotor, lo que le confiere mayor rigidez y estabilidad. Posee piezas de aluminio y fibra de vidrio que son bastante resistentes. Respecto a los servomotores, incorpora unos lobot ldx-218 que tienen muy buenas características de velocidad y torque.</td></tr>
</table>

Tabla 8: Características Robo-soul 18 LDX-218

- Hay que dejar claro que sólo se contemplarán las opciones de Aliexpress debido a que las demás se salen del presupuesto.

3.1.2 Selección de materiales

En esta sección vamos a hacer una recopilación de las características más importantes de los productos que consideramos más de interés, comparando pros y contras para finalmente hacer una elección final.

En primer lugar los kits de piezas de Trossenrobotics y Roboshop no se tendrán en cuenta por su excesivo precio, superando ambos los 200 euros solo contando las piezas; tampoco el kit completo de Roboshop, el cual se eleva hasta más de 800 euros. No obstante, nos ha servido para recopilar qué opciones hay en el mercado.

Los kits de piezas más económicos los encontramos en Aliexpress y los servos que incluyen los kits completos son más baratos que cualquier otro servo mostrado en los anteriores apartados. Por lo tanto, al final se concluirá con el análisis de los dos kits completos proporcionados por esta web, mostrando las diferencias de cada modelo y eligiendo uno definitivo, ya que previamente se han analizado sus características por separado y mostraban buenos diseños, así como sus materiales.

3.1.2.1 Elección del pack

Los dos packs que analizaremos serán 18 DOF aluminio Metal Hexapod Robot y Robo-soul 18 LDX-218, los cuáles llamaremos Modelo1 y Modelo2 respectivamente, para su análisis.

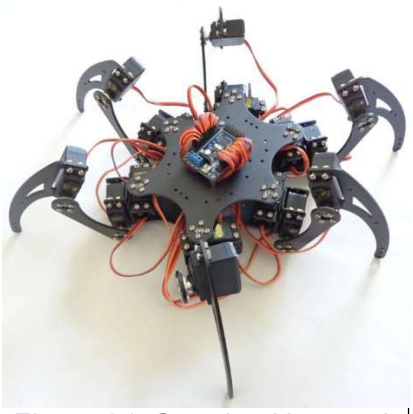
<table>
<tr><th colspan="4">Comparativa</th></tr>
<tr><td colspan="2" rowspan="2">Características</td><td>Modelo 1</td><td>Modelo 2</td></tr>
<tr><td> 
Figura 31: Smaring Hexapod Robot</td><td> 
Figura 32: Robo soul LDX-218</td></tr>
<tr><td rowspan="2">Estructura</td><td>Cuerpo</td><td colspan="2">• Los dos modelos poseen un cuerpo cuyos vértices forman un hexágono, garantizando una buena estabilidad, por lo que en este aspecto son iguales</td></tr>
<tr><td>Patas</td><td>• Estas son menos estables al tener solo un punto de anclaje con el eje del servomotor</td><td>• Gozan de dos puntos de anclaje con el servomotor, por lo que son mejores en estabilidad</td></tr>
<tr><td colspan="2">Materiales</td><td>• Toda la estructura está formada por aluminio</td><td>• Tiene partes de aluminio pero otras de fibra de vidrio</td></tr>
<tr><td rowspan="4">Servos</td><td>Velocidad</td><td colspan="2">• La velocidad es la misma para los dos: 0.16sec/60° (7.4 V)</td></tr>
<tr><td>Torque</td><td>• 20 kg.cm</td><td>• 17 kg.cm</td></tr>
<tr><td>Control</td><td colspan="2">• Ambos modelos se controlan a una frecuencia de 50 Hz
• Con los mismos anchos de pulso: 500 us – 2500 us</td></tr>
<tr><td>Rango de movimiento</td><td colspan="2">• 180º para ambos modelos</td></tr>
<tr><td colspan="2">Precio</td><td>206.3 €</td><td>217 €</td></tr>
</table>

Tabla 9: Elección de Modelo

- **Conclusión**

Los factores diferenciadores más relevantes son el tipo de estructura de las patas y el material de construcción, siendo factores menos relevantes la leve diferencia en el torque de los servomotores incluidos, así como la diferencia de precio que es mínima.

El primer factor involucra al material de construcción y la estructura de las patas, el modelo 2 tiene algunas piezas construidas en fibra de vidrio, esto hace que el precio del producto sea más económico sin perjudicar la rigidez de las mismas. No obstante, el diseño de éstas suple la ligera desventaja debida al material de construcción. Por lo que en tema de materiales y estructura nos quedaríamos con el segundo modelo por el diseño de la estructura.

Respecto a los servomotores, la diferencia de prestaciones es mínima y no vamos a notar diferencias apreciables en nuestro proyecto, por lo que este aspecto no será determinante en la elección.

Por último el precio de ambos es similar, diferenciándose solo en 10 euros. Es de destacar que el precio suele fluctuar mucho debido a que en estas páginas suelen hacer rebajas o debido a que los fabricantes van modificando sus precios para ser más competitivos.

Por estas razones y en especial por la estructura de las patas, se elegirá el modelo 2 para el robot hexápodo.

3.1.2.2 Modelo 2

Figura 33: Robo-soul 18 LDX-218

Características:

- Kit de montura aleación de aluminio y material de fibra de vidrio, peso ligero, alta resistencia.
- Posición de agujero de reserva para interruptor y controlador.
- 18 Grados de libertad (18 servos) que permiten una gran variedad de movimientos.
- Adopta servo digital LDX-218 como actuador.
- Peso: 1920 gramos

3.2. Hardware electrónico

El hardware electrónico del robot está compuesto por una placa Arduino Uno R3 que realiza el papel de controlador global, además de esto, se tiene un controlador por modulación de ancho de pulso para el control de los servomotores y por último, se ha diseñado una shield para Arduino, que incluye una determinada circuitería adicional necesaria y los conectores para la comunicación con la controladora PWM, batería y leds indicadores.

3.2.1 Controlador principal

El controlador es el elemento que gobierna el movimiento del robot, su objetivo es dar distintas señales de control para cada pata y servomotor. Estas órdenes son recibidas por los controladores de servomotores y en consecuencia éstos se orientan de forma que todas las patas se posicionen de forma correcta para el movimiento del robot.

En este proyecto nuestro controlador es Arduino Uno R3. Sin embargo, Arduino es el nombre de la plataforma, la cual se va a explicar a continuación. No obstante, se ha elegido esta plataforma debido a que anteriormente se había trabajado por lo que resultó interesante ampliar el conocimiento sobre ella, antes de abordar otras alternativas más profesionales de fabricantes como STMicroelectronics o Microchip. Además, al ser una plataforma de desarrollo libre, tenemos una gran cantidad de librerías que nos ayudarán en nuestro proyecto.

3.2.1.1 Arduino

Arduino es sólo una plataforma de desarrollo de hardware libre basada en una placa de desarrollo de un microcontrolador y un software o entorno de desarrollo. Todo esto es diseñado para facilitar el uso de la electrónica en proyectos multidisciplinares.

Su “lenguaje propio” no es más que un conjunto de librerías que nos permiten programar con el lenguaje de programación C++ a un nivel más alto de abstracción, haciéndolo más sencillo y accesible para su aprendizaje pero, por contrapartida, nos quita eficiencia y configurabilidad para proyectos más complejos.

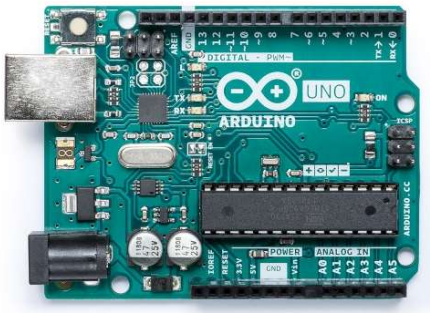


Figura 34: Arduino Uno R3

Gracias a estas librerías podemos controlar multitud de microcontroladores que forman parte de la plataforma Arduino y que poseen diferentes características, como manejo de datos de 32 bits, más memoria flash o número de gpio etc… dándonos multitud de opciones para nuestros proyectos.

El IDE o entorno de desarrollo está basado en processing, un software que se basa en el lenguaje Java y cuya filosofía de desarrollo es la misma que la de Arduino: simplificar y facilitar el desarrollo.

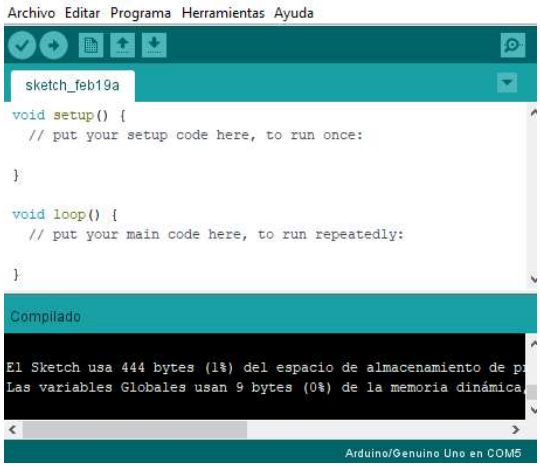


Figura 35:IDE oficial de Arduino

Este entorno se caracteriza por tener dos grandes bloques, setup y loop.

- Setup
  Este bloque se utiliza para las configuraciones e inicializaciones de variables.

- Loop
  En esta parte el código escrito se estará ejecutando en bucle infinitamente.

Para las exigencias de este proyecto nos es suficiente con las características de un Arduino Uno, a excepción del número de pines con la funcionalidad de PWM. Este problema lo solventaremos con la adición al proyecto de dos controladores PWM, para controlar el total de 18 servomotores que poseemos. Se decantó por esta solución debido a que el siguiente modelo que nos encaja

(Arduino Mega) cuesta 35 euros, lo cual es 15 euros más que Arduino Uno y estos controladores cuestan alrededor de uno o dos euros vendidos desde China. A parte de esto, añadimos al proyecto la implementación de comunicación I2C, enriqueciendo el total de características implementadas en el mismo.

3.2.1.1.1 Características Atmel 328p

Arduino Uno es una placa de microcontrolador que incluye el ATmega328 cuyas características se encuentran en la tabla inferior. A parte de esto, posee una conexión USB para programarlo, un conector de alimentación, un encabezado ICSP y un botón de reinicio.

Microcontrolador	**ATmega328p**
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines digitales I/O	14 (de los cuales 6 tienen la funcionalidad PWM)
Pines Digitales PWM I/O	6
Pines de entrada analógicos	6
Corriente continua por pin I/O	20 mA
Memoria Flash	32 KB (ATmega328P) de los cuales 0.5 KB son reservados para el bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad del oscilador	16 MHz
largo	68.6 mm
ancho	53.4 mm
peso	25 g

Tabla 10: Características ATmega328

3.2.1.2 IDE Eclipse

El IDE o entorno de desarrollo para Arduino es muy sencillo y fácil de usar sin demasiadas opciones, menús, etc… Las cuáles podrían asustar a un usuario inexperto. Por este motivo carece de muchas de las funcionalidades que nos brindan entornos de desarrollo más completos y que nos pueden ayudar significativamente en proyectos más complejos como el actual. Por este motivo, elegiré el IDE que mejor se adapte a las necesidades del proyecto.

Existen multitud de entornos con los que tendremos funcionalidades extra como mayor rapidez de compilación o mejor control del código; algunos de los mejores son: Visual studio, Atmel studio o Eclipse.

Visual studio es el entorno de desarrollo por excelencia de Microsoft y posiblemente el mejor IDE para Arduino, posee una versión gratuita denominada Community y junto con el plugin visual micro se puede utilizar con Arduino. Atmel studio es una IDE basada en el anteriormente mencionado Visual studio y que se utiliza para programar microcontroladores de Atmel. Como mejor alternativa de código abierto tenemos Eclipse. Este IDE nos permite trabajar con diferentes lenguajes como Java o C++ y en su versión de C++ cuenta con varios plugin para trabajar con Arduino.

La IDE que se usará en este proyecto será Eclipse con el plugin Sloeber, debido a que es gratis y de código abierto, por lo que tiene mucho soporte; aparte de este motivo, la razón de mayor peso es que ya se tenía experiencia con este entorno, por lo que se decantó por esta solución por las facilidades y funcionalidades que incorpora.

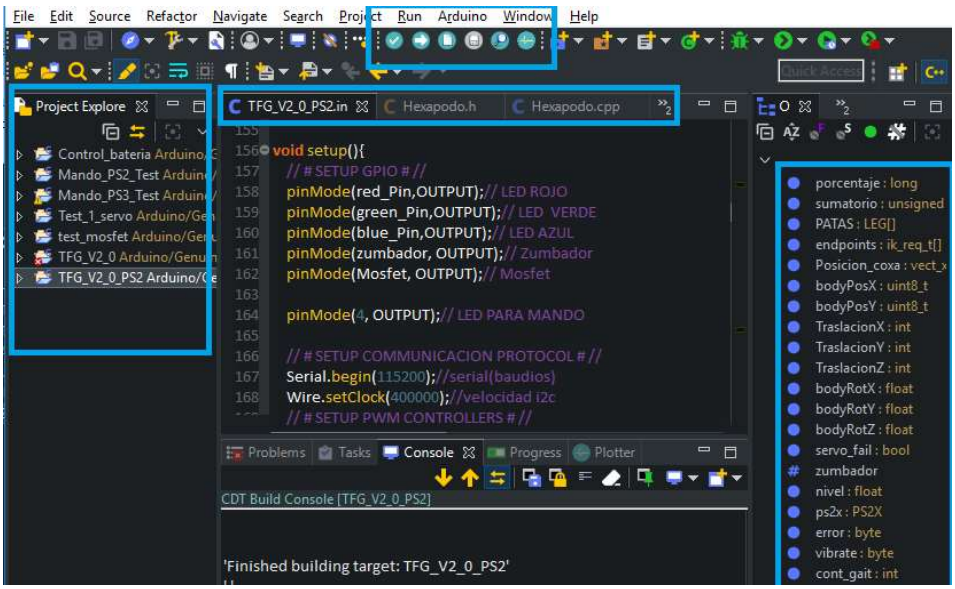


Figura 36: Eclipse para Arduino con plugin Sloeber

Ventajas que aporta el uso de Eclipse:

- Formatos y colores: Colores para los distintos tipos de objetos, librerías, y variables. Pudiendo distinguir rápidamente las principales partes del código, e indicación del número de línea.
- Sistema inteligente de escritura: Nos permite escribir automáticamente partes del código para agilizar la tarea, así como completar el nombre de variables o acceder a los métodos de un objeto.
- Podemos tener varios proyectos en el explorador, así como la posibilidad de manejar y editar archivos .h y .cpp
- Se pueden usar todas las funcionalidades de C++
- Este IDE nos proporcionará un inmejorable control de errores, indicando la línea en la que ocurre y un comentario o sugerencia del posible motivo.
- Un gestor de librerías mejorado

Todas estas características nos permiten mejorar en rapidez y eficiencia a la hora de programar y también en solventar errores, consiguiendo un resultado que no podríamos haber conseguido con el IDE estándar.

3.2.2 Controlador de servomotores

Como se ha dicho anteriormente, hay que añadir al proyecto dos controladoras pwm debido a que una sola tiene un total de 16 salidas, cuando nuestro robot posee 18 servos. Para que cada controladora trabaje con la misma carga asignaremos 9 servomotores a cada controladora.

Esta controladora se basa en el integrado PCA9685 el cual es un generador de PWM cuya comunicación se realiza por I2C por lo que podemos conectar un Microcontrolador como Arduino para modificar los pulsos a conveniencia y de este modo controlar la posición de los servomotores.

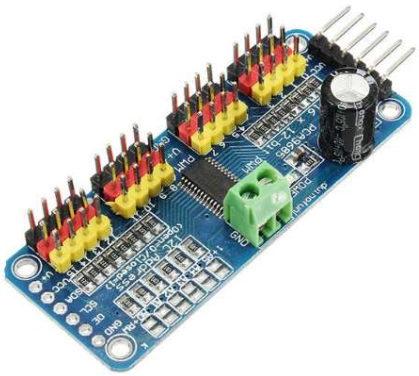


Figura 37: Controlador PCA 9685

El PCA9685 emplea únicamente 2 líneas para su control, los pines SDA y SCL son utilizados para el protocolo I2C. La frecuencia del PWM es ajustable hasta a 1600 Hz y la precisión de 12 bits, lo que supone un valor entre 0 y 4095.

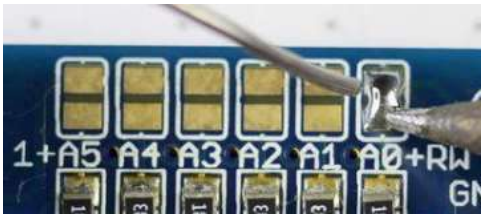


Figura 38: Pads soldables

Como vemos en la imagen superior, dispone de 6 terminales soldables, en los que una soldadura equivale a poner un uno, que nos permiten elegir la dirección de comunicación I2C, lo que supone que puede direccionarse hasta 62 módulos, ya que dos direcciones están restringidas. Como ejemplo, en la imagen superior la Dirección es igual a 0x41, en binario=1+ 000001.

El esquema de conexionado es el siguiente:

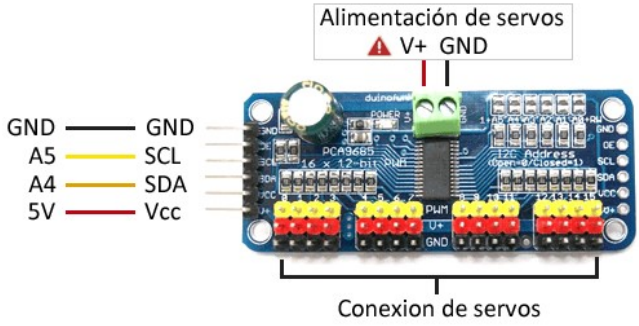


Figura 39: Conexionado comunicación i2C

El integrado funciona a 5 V, por lo que puede ser alimentado por el Arduino. Sin embargo, los servos tienen que ser alimentados por una fuente externa, ya que en nuestro caso funcionan con un voltaje de 7.4V y además Arduino no puede suministrar suficiente corriente para ellos.

Esta controladora incorpora un condensador de 1000uF. Fabricantes como Adafruit recomiendan un condensador con una capacidad de n * 100uF, donde n es el número de servos, aunque depende del torque del motor y de la corriente demandada. Nosotros sólo vamos a utilizar 9 servos por controlador, por lo que es un buen valor.

3.2.2.1 Comunicación i2c

El estándar I2C (Inter-Integrated Circuit) fue desarrollado por Philips en 1982 para la comunicación interna de sus dispositivos electrónicos. Posteriormente, se fue adoptando por otros fabricantes hasta convertirse en un estándar.

El bus I2C sólo necesita de dos cables para su funcionamiento, uno para la señal de reloj (CLK) y otro para el envío de datos (SDA), lo que es una ventaja frente al bus SPI. Por contra, su funcionamiento es un poco más complejo, así como la electrónica necesaria para implementarla.

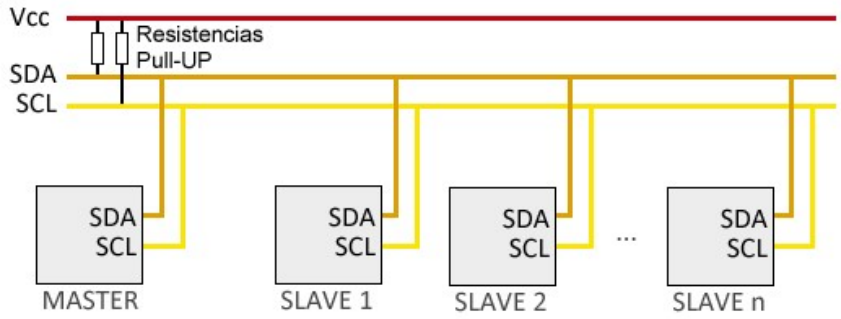


Figura 40: Protocolo I2C

En este protocolo cada dispositivo dispone de una dirección única, que se emplea para acceder a los dispositivos de forma individual. Esta dirección puede ser fijada por hardware, como en nuestro caso, o por software.

Este bus I2C tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación (el maestro tiene que preguntarles), ni hablar entre sí directamente.

El bus I2C es síncrono. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, evita la necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar una velocidad de transmisión y mecanismos para mantener la transmisión sincronizada (como en UART).

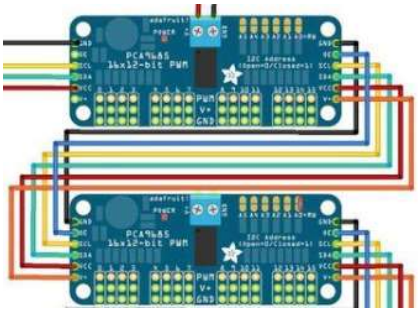


Figura 41: Conexión en serie de controladoras

El protocolo I2C prevé resistencias de Pull-up de las líneas a Vcc. En Arduino la librería Wire activa las resistencias internas de Pull-up, por lo que no es necesario añadir unas. No obstante, cada controlador incluye las suyas propias de 10k.

3.2.3 Mando Controlador

En este proyecto vamos a poder controlar en todo momento el movimiento del robot en cualquier dirección, también podemos mover el propio cuerpo del robot en los tres ejes del espacio e incluso hacer que rote un determinado ángulo sobre ellos (Roll, Pitch and Yaw). Para esto necesitamos un controlador externo, un mando.

3.2.3.1 Mando PS2 por cable

Para este proyecto utilizaremos un mando PS2 con cable, por la sencillez que presenta su implementación y su bajo coste. Anteriormente se han descartado opciones como mandos bluetooth u mandos PS2 inalámbricos ya que dependiendo del modelo podrían funcionar o no. Por lo que se ha optado por la solución más fiable, debido a que si en un futuro el mando llegase a fallar este pudiera ser fácilmente reemplazable.

Figura 42: Mando PS2 por cable

Para controlar el mando se conectarán los pines del receptor del mando a Arduino. Hay varias maneras de hacerlo, puesto que la librería que controla los comandos recibidos (*PS2X_lib*) nos permite dejar a nuestra elección los puertos a los que se conectará. No obstante, hay que tener cuidado en qué pines se eligen, para no interferir con otras funcionalidades del controlador.

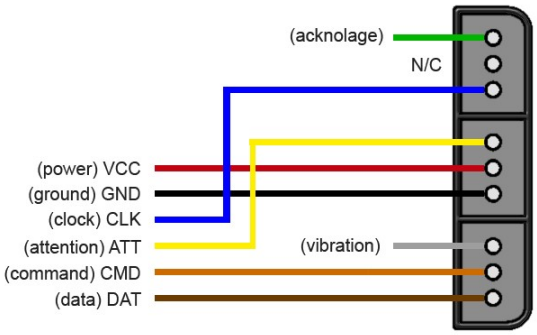


Figura 43: Patillaje conector PS2

Mando	Arduino
Data	Pin 13
Command	Pin 12
Attention	Pin A0
Clock	Pin 8
VCC	3.3 V
GND	GND

Tabla 11: Pines de comunicación mando-Arduino

3.2.3.2 Control sonoro

Para la funcionalidad de aviso sonoro en respuesta de un cambio de funcionamiento del robot se utilizará un buzzer.Este es un piezoeléctrico que funciona como un mini altavoz que puede ser controlado a través de un puerto digital, activándose y desactivándose para emitir sonidos a distintas frecuencias.

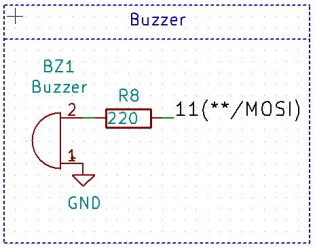


Figura 44: Esquema buzzer

Figura 45: Buzzer

3.2.4 Control del encendido

Para controlar el encendido del robot se usarán dos interruptores, uno global que al activarse provee de energía al microcontrolador, entrando éste en funcionamiento; y otro que es controlado por Arduino y cuya función es la de activar la alimentación de los servomotores cuando detecte que hay suficiente carga en la batería para el funcionamiento.

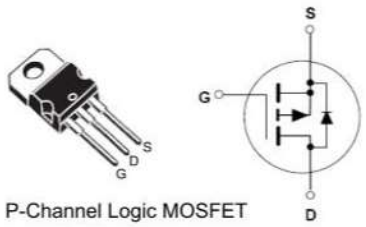


Figura 46: Mosfet de canal P

El primer interruptor es de tipo palanca y lo activará el usuario físicamente, mientras que el papel del segundo lo desempeña un mosfet controlado electrónicamente por Arduino. Este mosfet es de tipo p debido a que el interruptor se tiene que colocar en la línea de alimentación, no en la de tierra, por motivos de diseño.

La característica fundamental de los mosfets tipo p es que el voltaje V_{gs} debe ser negativo para que este se active, lo cual no implica que haya que aplicar un voltaje negativo a la puerta, sino que este voltaje tiene que ser más negativo que el del surtidor para activarse (dentro de un rango). Para nuestro mosfet el rango válido de V_{gs} para su activación se muestra en la siguiente imagen.

$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	-0.034	—	V/°C
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	—	—	0.06	Ω
$V_{GS(th)}$	Gate Threshold Voltage	-2.0	—	-4.0	V
g_{fs}	Forward Transconductance	8.0	—	—	S

Figura 47: Características Mosfet IRF 5305P

Pero surge un problema, para su activación sólo se cuenta con un Arduino, el cual sólo puede poner una salida a 0V o a 5V. Si conectamos directamente la salida de Arduino con la puerta, con 0V V_{gs} sería -Vbat (-6.4V ~-8.4V), el mosfet se activaría sin problemas dejando pasar la corriente. Sin embargo, con 5V, V_{gs} estaría variando entre -1.4V y -3.4V estando en un estado dudoso entre activo y desactivado. Por este motivo necesitamos un circuito externo que nos active o desactive perfectamente el mosfet.

Con la puerta del transistor conectada a Vbat (1 lógico) $V_{gs} = 0$, con lo cual el transistor está desactivado. Cuando la conectamos a tierra (0 lógico) $V_{gs} = -Vbat$, y como consecuencia se activa.

Esta selección se hará con un segundo transistor, en este caso un bjt, el cual se controlará controlar desde Arduino. Este funcionamiento corresponde con una puerta lógica inversora, cuya tabla de verdad se muestra a continuación.

Salida Arduino	Puerta Mosfet	Activación Mosfet
1	0	1
0	1	0

Tabla 12: Tabla de verdad para activación mosfet

El funcionamiento se muestra en las siguientes imágenes:

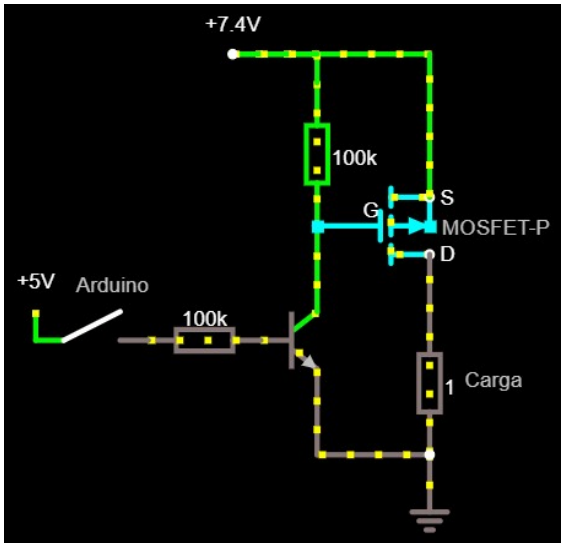


Figura 48: Mosfet desactivado

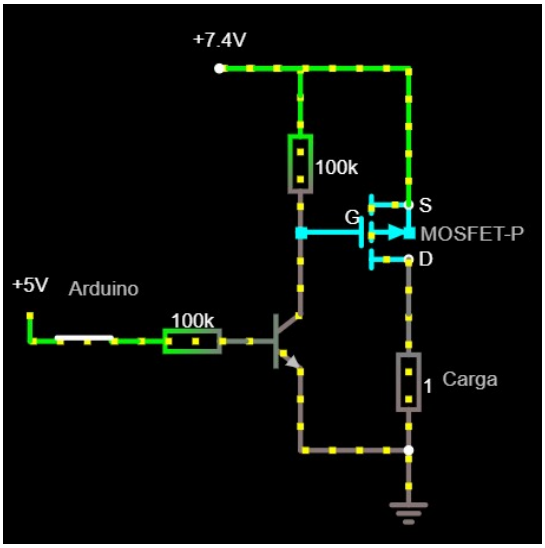


Figura 49: Mosfect activado

<u>Desactivado</u>

$$V_g = Vbat;$$

$$V_{gs} = Vbat - Vbat = 0$$

<u>Activado</u>

$$V_g = 0;$$

$$V_{gs} = 0 - Vbat = -Vbat$$

El último paso es verificar que el consumo del mosfet está dentro de unos márgenes normales de funcionamiento y que no hará falta añadir disipador. Las características mostradas en la siguiente tabla son las necesarias para calcular este consumo.

R_{ds}	Resistencia interna del mosfet, entre el drenador y el surtidor	$0.06\ \Omega$
$R_{\theta JA}$	Resistencia térmica entre la junta y ambiente	$62\ \frac{ºC}{W}$
T_{MJ}	Temperatura máxima de la junta	$175ºC$

Tabla 13: Características de consumo de potencia IRF 5305P

Potencia máxima disipada (sin disipador), considerando una temperatura ambiente de 25ºC:

$$P_D = \frac{175º - 25º}{62\ \frac{ºC}{W}} \cong 2.42\ W$$

Potencia que disiparía nuestro mosfet suponiendo un consumo máximo de 6 amperios:

$$P = R_{ds} * I^2 = 0.06 * 6^2 = 2.16\ W$$

Por lo tanto como $P < P_D$ no necesitamos disipador.

3.2.5 Control de batería

La batería usada en este proyecto es de tipo LIPO (polímero de litio), la cual es utilizada para otros para proyectos similares, como drones, coches radio control etc… El voltaje que proporciona esta batería es de 7.4V, el cual es el necesario para alimentar nuestros servomotores.

Figura 50: Batería lipo 7.4V HRB

Especificaciones	
Voltaje batería	7.4V
Voltajes máximos	6.4-8.4V
Nº celdas	2
Capacidad	6000 mA
Tasa de descarga continua	60C
Conector	JST-XH
Certificado	CE, ROHS, MSDS, UN38.3
Peso	358g

Tabla 14: Características de batería LIPO 7.4V HRB

Para cargar la batería se necesitará un cargador especial para baterías tipo LIPO. Este se encargará de balancear todas las celdas para aumentar su duración.

Figura 51: Cargador-balanceador de baterías Lipo

Por tanto, la circuitería de control a diseñar debe vigilar el voltaje de la batería para que esta no se descargue más de lo requerido y reduzca su vida útil. Además, un led rgb visualizará el nivel en el que se encuentra la carga de esta.

3.2.5.1 Divisor de Tensión

Para controlar el nivel de carga de la batería hay que saber a qué voltaje está trabajando, pero Arduino no admite en sus pines más de 5V, por lo que debemos diseñar un divisor de tensión que reduzca este voltaje hasta uno que podamos medir. El voltaje máximo que puede alcanzar la batería es de 8.4V por lo que es bueno hacer una sobredimensión hasta 9V por precaución.

Para reducir el ruido de alta frecuencia se incluirá un condensador cerámico de 100nF y para proteger el puerto analógico frente a una posible sobretensión se colocará un diodo zener de 5.1V.

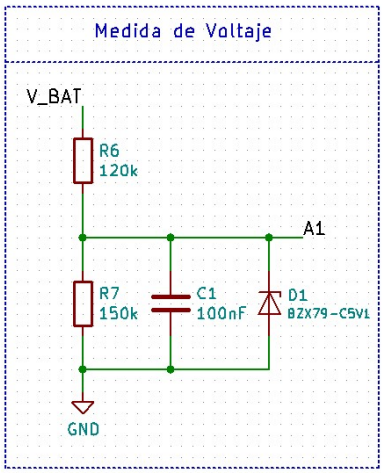


Figura 52: Esquema medida voltaje

- Divisor de tensión:

$$V_{A1} = \frac{R7}{R6 + R7} * V_{bat}$$

Considerando Vbat (maximo) = 9V, $V_{A1} = 5V$ y si fijamos R7 a 150k obtenemos un valor de R6 de 120k.

Siendo V_{bat}: $V_{bat} = \frac{R6+R7}{R7} * V_{A1}$

Como se muestra en la expresión 2 el voltaje de la batería sigue una relación lineal, por lo que en Arduino hay que implementar una función que nos devuelva el voltaje real de la batería, teniendo en cuenta la tolerancia de las resistencias y la precisión del puerto analógico, que en este caso es de 10 bits (valores entre 0 y 1023).

La mejor forma de realizar esto es calcular una recta experimental; ésta relacionará el voltaje de la batería con el nivel de voltaje medido en Arduino. Por lo que un punto estará formado por dos componentes, uno será el voltaje de la batería y el cual se medirá con el multímetro, y el otro componente será el valor digital medido por el puerto analógico del Arduino. Se tomarán dos puntos

a distintos voltajes para calcular la recta experimental que se asemejará al comportamiento real.

	V_{bat} (V)	V_{A1}(nivel digital)
Punto 2	8.4	943
Punto 1	6.4	718

Tabla 15: Tabla puntos experimentales

Utilizando la ecuación de la recta en forma punto pendiente:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = 0.00\hat{8}$$

$$y - y_1 = m(x - x_1) = mx - mx_1 + y_1$$

$$y = mx + n$$

Obtenemos finalmente nuestra recta a implementar en el código:

$$y = 0.00\hat{8}x + 0.01\hat{7}$$

- Y: Voltaje de la batería
- X: Nivel digital del puerto analógico

3.2.5.2 LED RGB

Para visualizar el estado de la batería al usuario se utilizará un Led RGB el cual dependiendo del nivel de carga de la batería variará su color desde verde, pasando por amarillo, hasta rojo.

En este caso se dispondrá de un led RGB de cátodo común, por lo que esta patilla se tendrá que conectar a tierra. Las demás patillas van conectadas a salidas digitales PWM de Arduino, con las cuáles podremos controlar su luminosidad y la activación del color que se quiera.

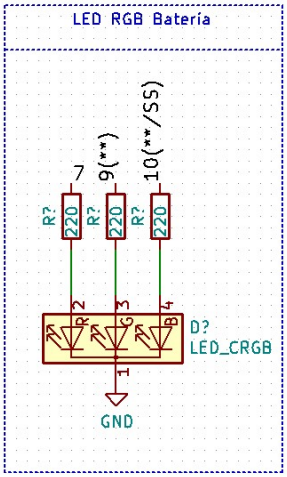


Figura 53: Led RGB Batería

Las resistencias de 220 ohmios colocadas permiten limitar la corriente que pasa por el led RGB.

3.2.6 Shield Arduino

Una vez se han diseñado todos los elementos necesesarios para el correcto funcionamiento del robot, estos se deben ubicar en el mismo. Como prácticamente todo hay que conectarlo a Arduino y además, sus pines de son hembra, la mejor opción es diseñar un shield que se conecte a todos los pines, de esta manera se ahorrará espacio y cableado innecesario.

Para el desarrollo del PCB se empleará el software libre Kicad. Éste fue desarrollado por Jean-Pierre Charras en 1992 y desde entonces ha crecido enormemente gracias a colaboradores, voluntarios y donaciones. A partir del 2013 el software tuvo un gran empuje gracias a que una sección del CERN llamada BE-CO-HT (Hardware and Timing), comenzó a aportar recursos a KiCad para ayudar a fomentar el desarrollo de hardware abierto.

El Software kicad está organizado en 5 partes:

- kiCad – Es el administrador de proyectos.
- eeschema - Editor de esquemáticos.
- pcbnew - Entorno de diseño de los circuitos impresos (PCB).
- gerbview - Visualizador de archivos Gerber (vista em 3D).
- Bitmap2Component - Herramienta que convierte imágenes a footprints (huellas) para realizar "PCB artwork".
- Calculadora rápida - Permite calcular el ancho de las pistas dependiendo de distintas variables; además de otros cálculos.

3.2.6.1 Proceso de creación

El proceso de desarrollo de un PCB sigue el siguiente esquema:

Figura 54: Proceso de desarrollo de PCB

1. Diseño de circuito

   Este paso ya se ha completado en los apartados anteriores, diseñando los circuitos necesarios que cumplan con las funcionalidades requeridas.

2. Creación de símbolos para el esquemático

   En kicad hay multitud de librerías que aportan símbolos para los componentes. No obstante, si faltase algún símbolo, se podría crear.

   Para este proyecto se han descargado las librerías de todos los componentes menos para el conector del mando PS2 que se ha tenido que crear un símbolo propio.

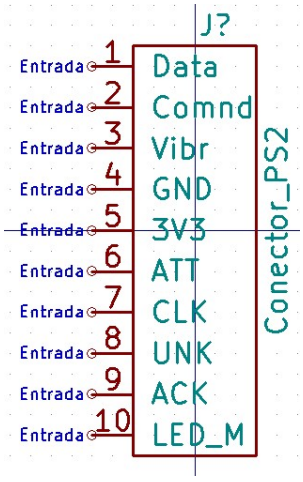


Figura 55: Conector PS2

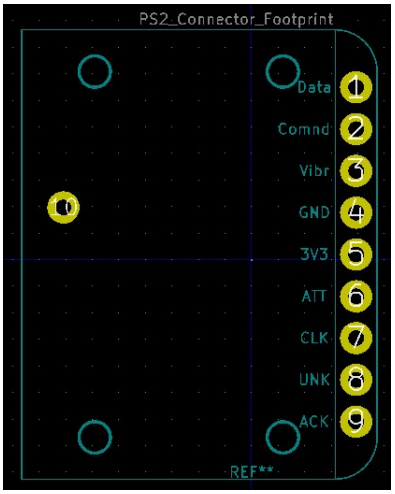
3. Realización del esquema de conexiones

En este esquema se ubican todos los componentes y conectan todos entre sí de manera correcta para el posterior enrutado en PCB.

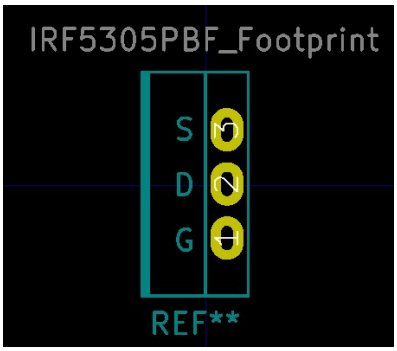
- Ver: ANEXO 3: ESQUEMÁ ELECTRÓNICO SHIELD PCB

4. Creación de huellas

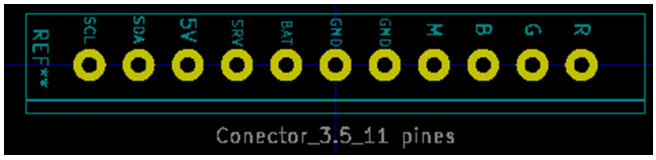
Cada símbolo necesita tener su propia huella para la pcb, la cual indica tanto sus dimensiones, como forma y diámetro de pines. En este proyecto se han tenido que crear las siguientes huellas:



a) *Huella conector PS2*



b) *Huella Mosfet*



c) *: Huella Conector 11 pines*

Figura 56: Huellas creadas para PCB

5. Generación del Netlist

El Netlist es un archivo que indica cómo y de qué manera están conectados todos los pines. Se obtiene a partir del esquemático y es necesario para el subprograma pcbnew, para hacer el conexionado de todos los componentes en la placa pcb.

6. Layout de la placa

El contorno de la placa (de forma shield Arduino) lo podemos importar como una plantilla predefinida de las cuales existen más opciones. Una vez hecho esto cargamos el netlist, el cual nos cargará la huella de cada componente y nos unirá con líneas los pines que tengan que estar conectados. Una vez ubicados todas las huellas en su lugar correspondiente, procedemos a enrutarlas.

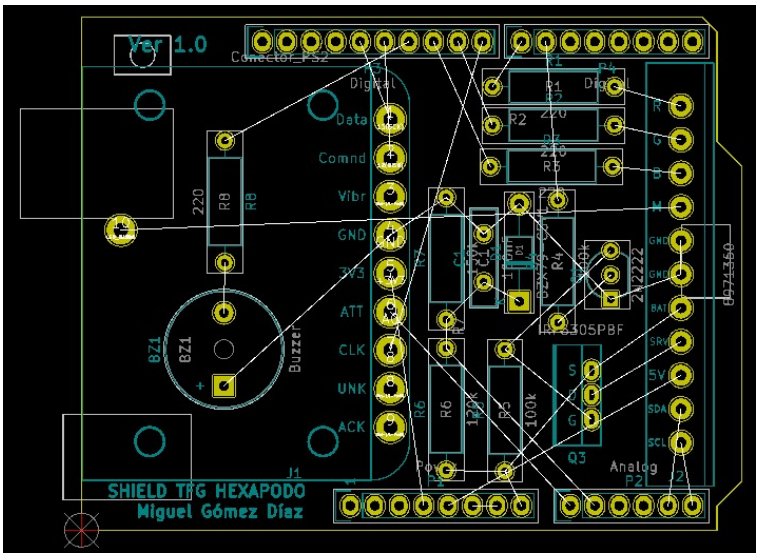


Figura 57: Lyout antes de enrutar

Totas las pistas tienen un ancho estándar, ya que no se requieren grandes corrientes. No obstante, para la línea de alimentación esto no puede ser así. Para el ancho de la pista en las conexiones de Vbat se ha utilizado la opción de calculadora de Kicad,que permite calcular cómo de ancha debe ser la pista en función de la intensidad máxima que se requiera, el grosor de cobre o la diferencia máxima de temperatura que puede alcanzar.

Para el grosor de esta pista, considerando un aumento de temperatura máximo respecto a la ambiental de 20ºC y una intensidad de 4.5 A, resulta un ancho de aproximadamente 1.6 mm de grosor.

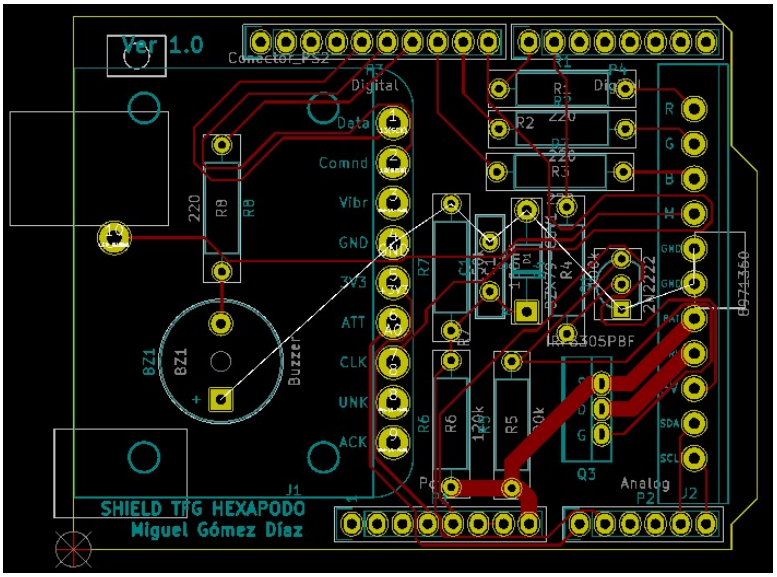


Figura 58: Lyout después de enrutar

La única unión que todavía no se ha realizado es la conexión de las tierras. Para estas conexiones es recomendable crear un plano de tierra, para que la disipación de calor sea más efectiva, además esto ayuda a la reducción de ruido.

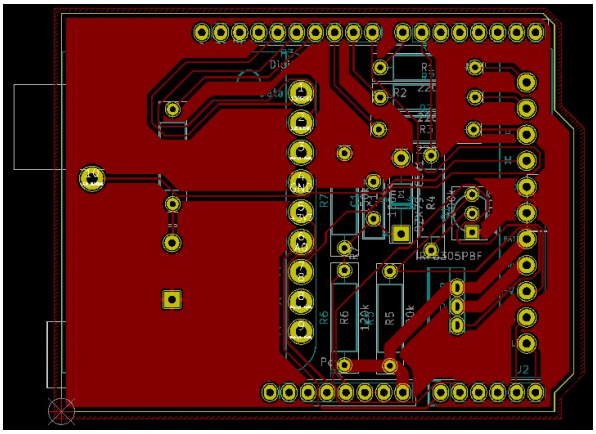


Figura 59: Lyout finalizado

7. Gerber y BoM

Por último se generan los archivos gerber, que son unos archivos que indican el enrutado de la placa, así como las coordenadas y dimensión de los orificios a realizar, para que los fabricantes tengan todos los datos relacionados con la placa y puedan fabricarla. Además, una vez finalizado todo el proceso de creación, podemos hacer una lista de los materiales necesarios (BoM).

3.2.6.2 Resultados

Kicad nos ofrece una vista el 3D que muestra una aproximación a lo que podría ser el resultado de la placa.

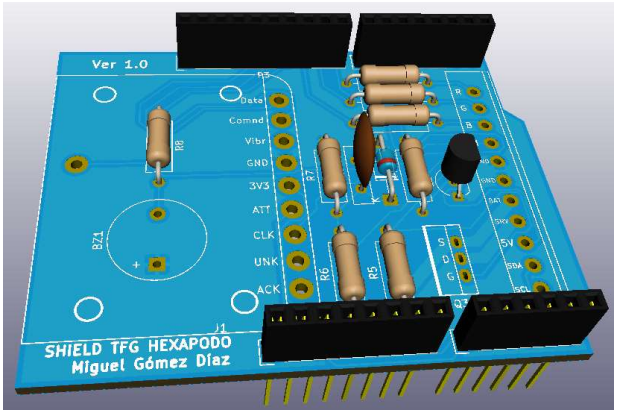


Figura 60: Renderizado en 3D

La empresa a la que se solicitó el producto fue JLCPCB, empresa mundialmente reconocida por la rapidez y calidad de sus productos. Además, los precios de fabricación son muy bajos gracias a su gran volumen de producción por lo que obtendremos un producto de calidad y a un precio asequible.

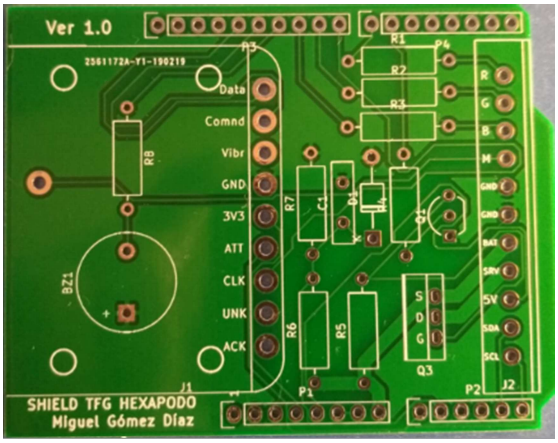


Figura 61: Shield PCB fabricada

Figura 62: Resultado final PCB

4. SOFTWARE

El objetivo de este apartado es mostrar los algoritmos que dotarán de movimiento al robot, como la implementación de la cinemática o el control de los servomotores, así como consideraciones previas como la ubicación espacial del robot o el ajuste inicial de los servomotores. Finalmente, se muestran los flujogramas de control resultantes.

4.1. Situación espacial del robot

Para establecer un método con el que controlar al robot primero tenemos que saber sus dimensiones y establecer un sistema de coordenadas en el que ubicarlo, para de esta manera saber exactamente en qué posición se encuentra cada extremo de cada una de las patas.

4.1.1 Patas y cuerpo

La pata está compuesta por tres articulaciones y longitudes características que unen cada articulación. El punto de origen de la pata, el punto coxa, está compuesto por dos componentes x e y, respecto del origen de coordenadas que se encuentra en el centro del robot, siendo su altura cero.

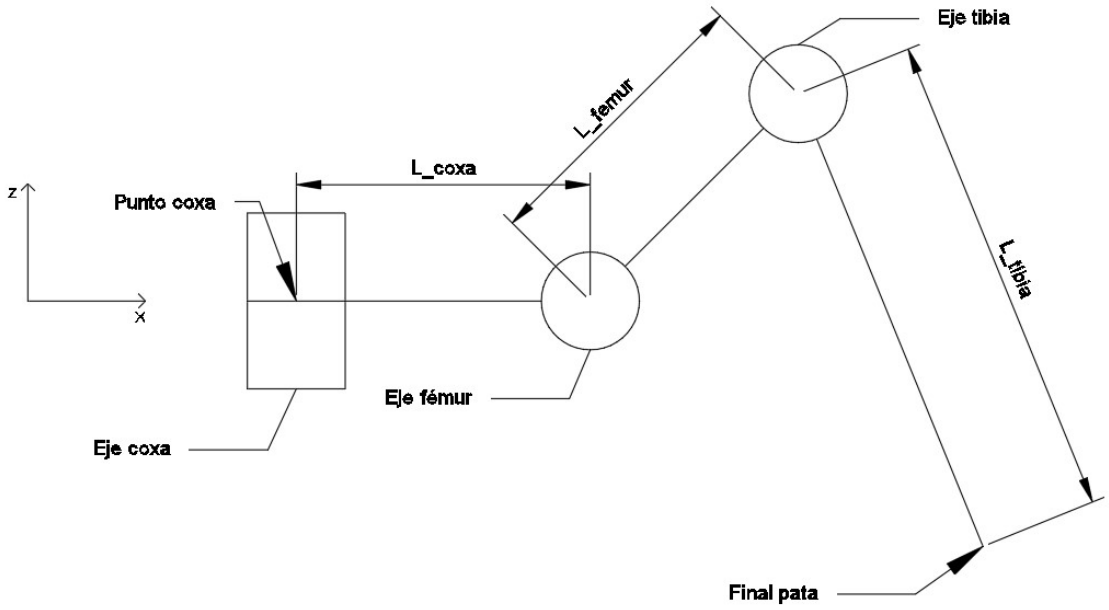


Figura 63: Esquema simplificado de una pata

El cuerpo del robot se compone de dos placas de fibra de vidrio, las cuáles sujetan por la parte superior e inferior a los servomotores coxa y cuyo origen

de coordenadas se encuentra en el centro del mismo tal y como se muestra en las siguientes dos ilustraciones.

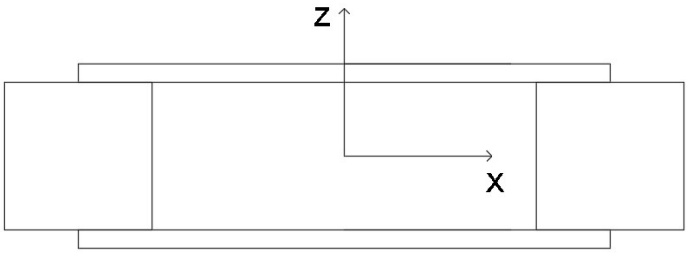


Figura 64: Cuerpo robot parte trasera

En la siguiente figura vemos como el cuerpo tiene forma hexagonal. En los vértices del hexágono se encuentran los puntos coxa de las patas. La disposición es la mostrada a continuación.

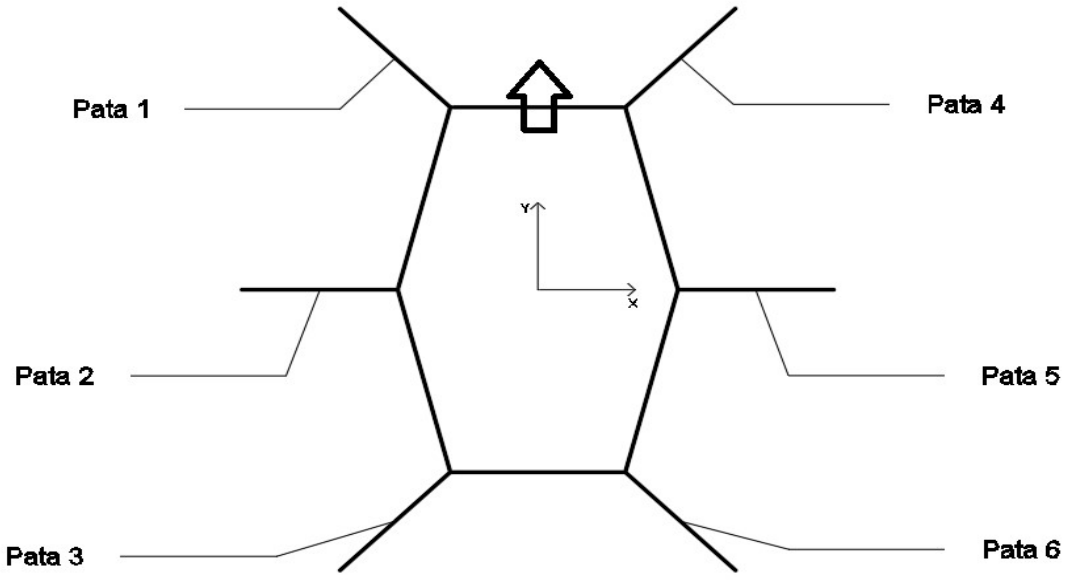


Figura 65: Disposición de patas

Asumiendo que los puntos coxas de las dos patas superiores con las dos patas inferiores forman un rectángulo, el valor absoluto de las distancias desde el punto coxa hasta el origen de coordenadas está definido por las constantes

X_COXA e Y_COXA, mientras que las patas centrales están alineadas con el origen, por lo que sólo están separadas de él una distancia M_COXA en valor absoluto, sobre el eje x.

A continuación se muestran los valores de las distancias características del cuerpo y las patas.

X_COXA	60 mm
Y_COXA	60 mm
M_COXA	100 mm

Tabla 16: Distancias coxa

L_COXA	48 mm
L_FEMUR	75 mm
L_TIBIA	135 mm

Tabla 17: Longitudes patas

Los puntos coxa son los siguientes y los utilizaremos para rotarlos y trasladarlos para conseguir un movimiento del cuerpo del robot.

PATA	Coordenada X	Coordenada Y
Pata 1	- X_COXA	Y_COXA
Pata 2	-X_COXA	0
Pata 3	- X_COXA	-Y_COXA
Pata 4	X_COXA	Y_COXA
Pata 5	X_COXA	0
Pata 6	X_COXA	-Y_COXA

Tabla 18: Coordenadas puntos coxa

En la siguiente tabla tenemos las coordenadas del punto final de la pata. Este punto se encuentra en el extremo de la pata y es medido respecto al punto coxa de la misma. La cinemática inversa tomará las coordenadas de este punto para calcular los ángulos de las articulaciones.

PATA	Coordenada X (mm)	Coordenada Y (mm)	Coordenada Y (mm)
Pata 1	-115	100	-100
Pata 2	-155	0	-100
Pata 3	-115	-100	-100
Pata 4	115	100	-100
Pata 5	155	0	-100
Pata 6	115	-100	-100

Tabla 19: Coordenadas de los extremos de las patas

4.1.2 Resolución de ángulos en código C++

Antes de comenzar con la cinemática del robot se ha de seleccionar un método para la resolución de los ángulos en el lenguaje de programación C++. Hay dos métodos entre los cuáles debemos elegir el que más ventajas nos aporte. Esta cuestión es muy importante porque según el cuadrante en el que se encuentre el punto en cuestión y dependiendo del método usado, obtendremos un ángulo distinto.

- atan (x)

Esta función devuelve el ángulo en radianes del parámetro x, en el intervalo [-pi/2,+pi/2].

Por lo tanto sólo se consideran el primer y el cuarto cuadrante, no teniendo información de los demás cuadrantes, o teniendo que realizar cálculos previos en los parámetros de entrada para averiguarlo.

- atan2 (y, x)

Esto se soluciona con atan2. Esta función devuelve el ángulo en radianes de los parámetros de entrada x e y, en el intervalo [-pi,+pi], teniendo en cuenta el signo de éstos para la elección del cuadrante. Se permite que el parámetro x sea 0, sin que de error.

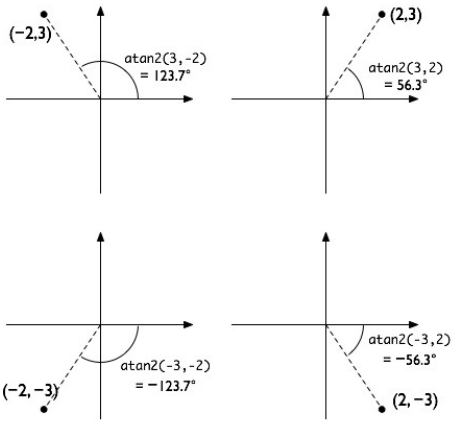


Figura 66: Ángulos según función atan2

El problema de estas resoluciones es que si el rango de movimiento de nuestro servomotor pasa por varios cuadrantes, tendremos un problema si el ángulo cambia de forma abrupta al pasar a otro cuadrante. Con atan2, como el intervalo es de [-pi,+pi], esto sólo se produciría una vez al pasar de un ángulo 180 (2º cuadrante), a -180 (3º cuadrante), aunque es solo cuestión de adaptarse según el método usado. En este proyecto se adaptará la resolución de los ángulos con este método, ya que nos simplifica más el problema.

4.2 Control de servomotores

Como ya se ha mencionado anteriormente los servomotores son los actuadores que harán la función de articulaciones en las patas de nuestro robot. Estos tienen que orientarse de una determinada manera dependiendo de la señal de control que se les envíe, para que cada pata alcance la posición requerida. En este subcapítulo se verá el método usamos para poder controlarlos.

4.2.1 Control por PWM

Para este tipo de servomotores el control se realiza por una señal PWM. Esto es una señal cuadrada en la cual el ancho del pulso varía de duración dependiendo de la posición en la que se quiera ubicar el servo.

Para el caso en cuestión los pulsos se realizan a una frecuencia constante de 50 hercios o cada periodo de 20 ms, de manera que si el ancho de pulso a nivel alto se mantiene en 1,5 ms el servo se localizará en una posición centrada. Si el pulso fuese más corto, por ejemplo 1 ms, el servo girará hacia la derecha, si el pulso es mayor, por ejemplo 2 ms, el servo girará hacia la izquierda, tal y como se ve en la siguiente figura.

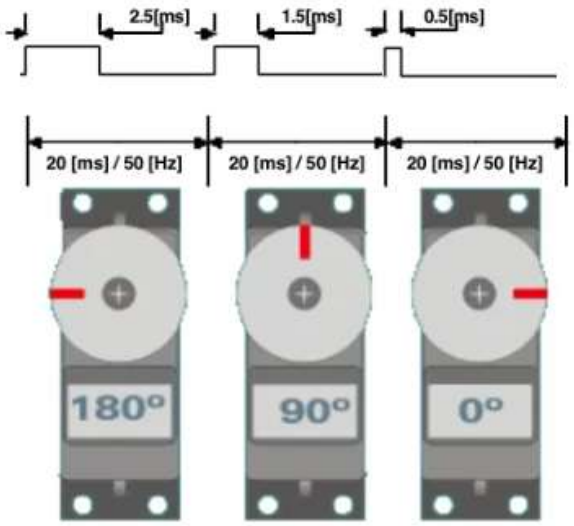


Figura 67: Giro vs ancho de pulso

4.2.2 Calibración de posición

Para el ajuste del servomotor se necesitará encontrar el valor correspondiente de PWM que oriente a éste al ángulo que deseemos. No obstante, se tiene la problemática de no poder orientarlo correctamente por el error de posición que ocasionan los engranajes, montaje, errores de fabricación etc… Por lo que, por ejemplo, no concordaría el valor de PWM teórico correspondiente a 0º, con la posición física de éste.

Figura 68: Detalle engranaje servo

La solución sería el ajuste vía software. Se procederá entonces a variar gradualmente la señal PWM hasta alcanzar la posición física deseada, para obtener una equivalencia entre nivel de PWM y grados. De este modo podemos obtener el nivel digital de PWM necesario para alcanzar el 0º (al cual le corresponde un valor a pesar de ser 0º), 90º o el ángulo que busquemos.

Además sabiendo el nivel de PWM para 0º y 90º se puede obtener una constante de proporcionalidad que nos indica cuánto se tiene que incrementar

el pulso de la señal (en niveles digitales) para obtener un incremento determinado de ángulo, debido a que la relación varía linealmente.

Sabiendo que los controladores tienen una precisión de 12 bits, esto da un total de 4906 valores posibles, que van de 0 a 4905.

Habiendo realizado un análisis experimental para todos los servomotores, se sabe que un incremento de 90º supone un aumento de 195 niveles digitales en la señal de PWM, por lo que se obtiene la relación:

$$\frac{\Delta\ nivel\ PWM}{\Delta\ 90º} = \frac{195\ niveles}{\Delta\ 90º} = 2.1\hat{6}\ niveles/º$$

O también:

$$\frac{195\ niveles}{\Delta\ 90º} * \frac{90º}{\frac{\pi}{2}\ rad} = \frac{195\ niveles}{\frac{\pi}{2}\ rad} = 124.1409\ niveles/rad$$

Por lo tanto la función a implementar en el código será la siguiente:

$$\frac{valor\_PWM}{x\ rad} = \frac{195\ niveles}{\frac{\pi}{2}\ rad}$$

$$valor\_PWM = \frac{195\ niveles}{\frac{\pi}{2}\ rad} * x\ rad$$

También existen otros valores críticos, que serán el nivel mínimo y máximo digital de PWM que cada servo puede alcanzar, y que puede variar de un servo a otro por cuestiones de fabricación. Esto hace que varíe el nivel necesario de PWM para alcanzar los ángulos anteriormente mencionados. No obstante, la constante de proporcionalidad permanece invariante para todos ellos, por lo que el incremento de PWM para un incremento 90º es igual en todos los servomotores.

A continuación se muestra una tabla con la recopilación de todos los valores de PWM tenidos en cuenta en el proyecto.

		PATA 3	PATA 1	PATA 6	PATA 4	PATA 2	PATA 5
		Valores PWM					
COXA	MIN	60	60	60	60	55	60
	MAX	500	500	500	500	495	500
	0º	105	110	90	80	110	75
	90º	300	305	285	275	300	270
	Δ 90º	195	195	195	195	195	195
FEMUR	MIN	60	70	55	60	60	55
	MAX	500	500	495	495	500	495
	0º	85	105	75	80	100	110
	90º	280	300	270	275	295	305
	Δ 90º	195	195	195	195	195	195
TIBIA	MIN	55	60	55	55	55	60
	MAX	495	505	495	495	490	500
	0º	155	170	180	150	180	155
	90º	350	365	375	345	375	350
	Δ 90º	195	195	195	195	195	195

Tabla 20: Tabla de constantes PWM

4.3 Desplazamiento del Robot

Para que el robot se desplace por el terreno se tiene que hacer que las patas tracen distintas trayectorias de manera coordinada, de forma que vayan alcanzando distintos puntos predefinidos en el espacio. Para ello se hará uso de las herramientas que nos proporciona la **cinemática**.

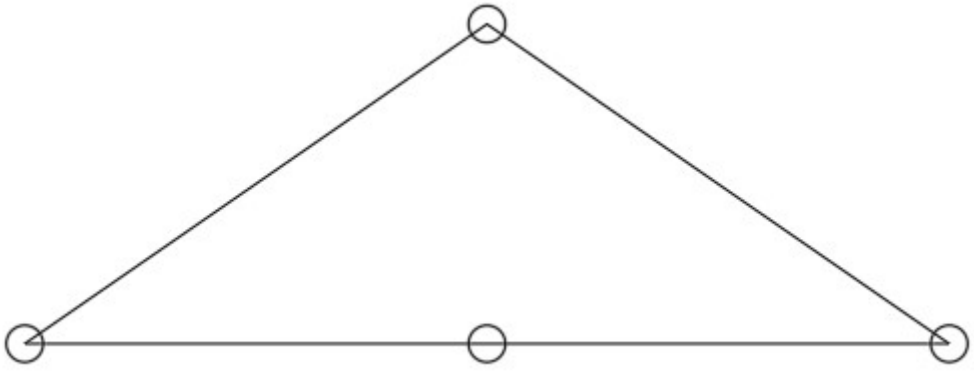


Figura 69: Ejemplo de trayectoria

La cinemática en robótica estudia el movimiento del mismo respecto a un sistema de referencia, por ello, previamente hemos definido un sistema de coordenadas en el cual ubicar el robot. De este modo se relaciona la posición y orientación del extremo final del robot con los valores que toman sus coordenadas articulares, obteniendo un método matemático para el control del mismo.

A la hora de resolver la cinemática existen dos problemas fundamentales: la cinemática directa que consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas, conociendo las coordenadas articulares; y la cinemática inversa que consiste en hallar las coordenadas articulares necesarias para adoptar una posición y orientación conocidas de antemano. Por lo que dependiendo del objetivo, convenientemente, usaremos una u otra.

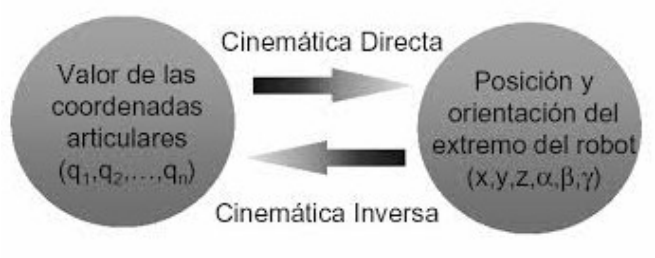


Figura 70: Relación entre cinemática directa e inversa

4.3.1 Aplicación cinemática inversa

A partir de un punto dado, que en este caso es el extremo de una pata, la cinemática inversa obtiene los valores que deben tomar las coordenadas articulares de nuestro robot para que el extremo de las patas se posicione a la forma requerida. En este caso, ya que sólo hay tres articulaciones, su resolución se obtendrá por el **método geométrico**.

Un punto clave es el referido al llamado "**Ángulo real**" del servomotor, ya que el ángulo devuelto por la cinemática inversa no es el ángulo que tenemos que enviarle al servomotor, esto es debido a que la cinemática inversa nos devuelve un ángulo respecto a su sistema de referencia y el servomotor tiene el origen de 0º en otra posición, por lo que para cada caso en particular habrá que hacer una conversión.

Cabe puntualizar que cada ángulo del servomotor corresponde con un valor de PWM, por lo que cuando se indica 0º, en realidad se refiere al valor de PWM necesario para alcanzar esos 0º, que no son 0 PWM; o cuando se habla de un incremento de 45º se traduce en el incremento de ancho de pulso (en niveles digitales) necesario para alcanzarlo.

4.3.1.1 Resolución ángulo teta 1

Debido a que cada pata está orientada de distinta forma en el cuerpo, para obtener una disposición hexagonal, habrá seis resoluciones distintas, una para cada pata. Además, por lo comentado en el apartado 5.1.2, *Resolución de ángulos en código C++,* algunas patas tendrán dos resoluciones dependiendo del cuadrante donde esté ubicado el extremo de la pata.

A continuación se verá la disposición de cada una de las patas:

- Pata 1

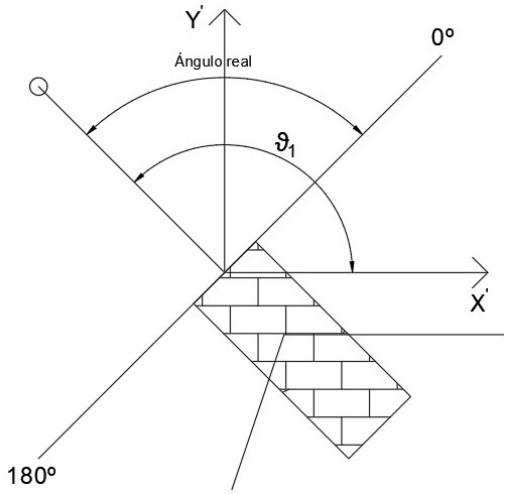


Figura 71: Resolución θ1 Pata 1 A

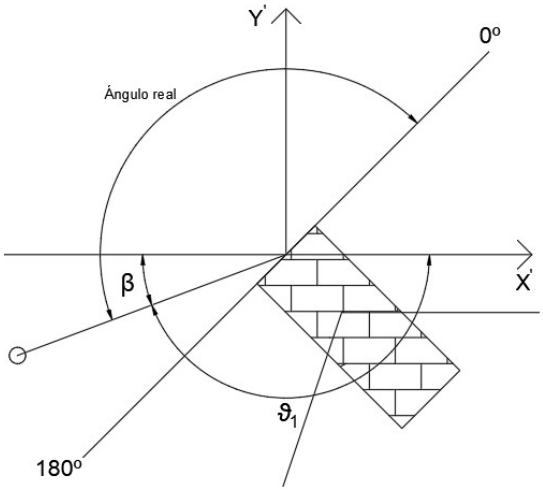


Figura 72: Resolución θ2 Pata 1 B

$$\theta_1 = Tg^{-1}\left(\frac{y}{-x}\right)$$

$$Ángulo\ real = \theta_1 - \Delta 45º + 0º$$

$$\theta_1 = Tg^{-1}\left(\frac{-y}{-x}\right)$$

$$\beta = 180º - |\theta_1|$$

$$Ángulo\ real = 3 * \Delta 45º + \beta + 0º$$

- Pata 2

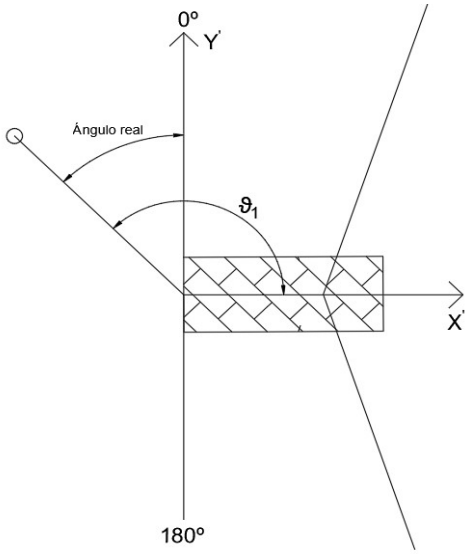


Figura 73: Resolución θ1 Pata 2 A

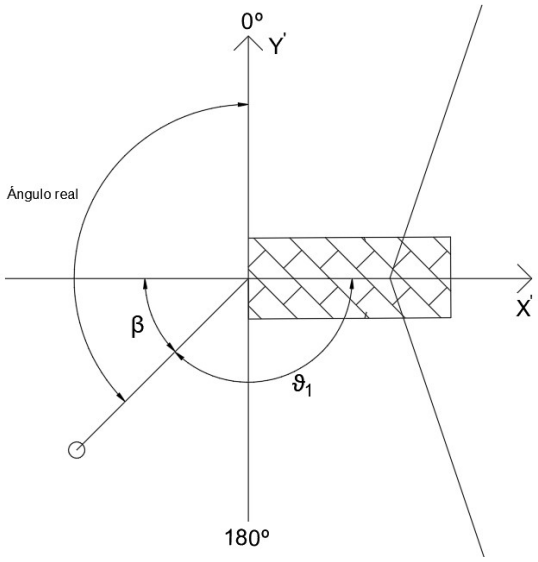


Figura 74: Resolución θ1 Pata 2 B

$$\theta_1 = Tg^{-1}\left(\frac{y}{-x}\right)$$

$$Ángulo\ real = \theta_1 - \Delta 90º + 0º$$

$$\theta_1 = Tg^{-1}\left(\frac{-y}{-x}\right)$$

$$\beta = 180º - |\theta_1|$$

$$Ángulo\ real = \Delta 90º + \beta + 0º$$

- Pata 3

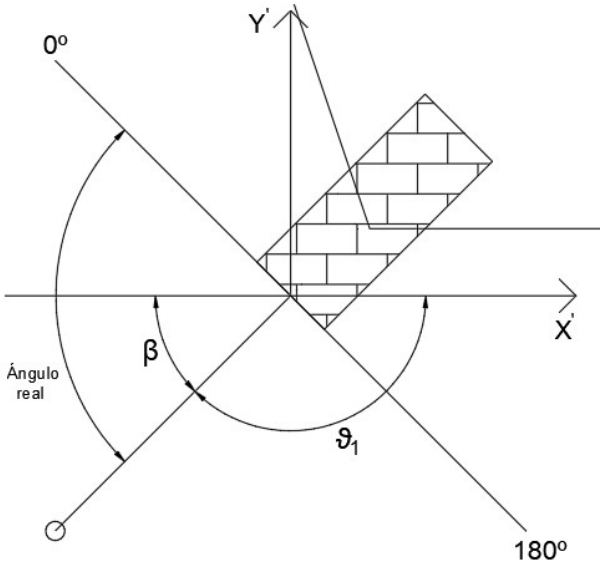


Figura 75: Resolución θ1 Pata 3 A

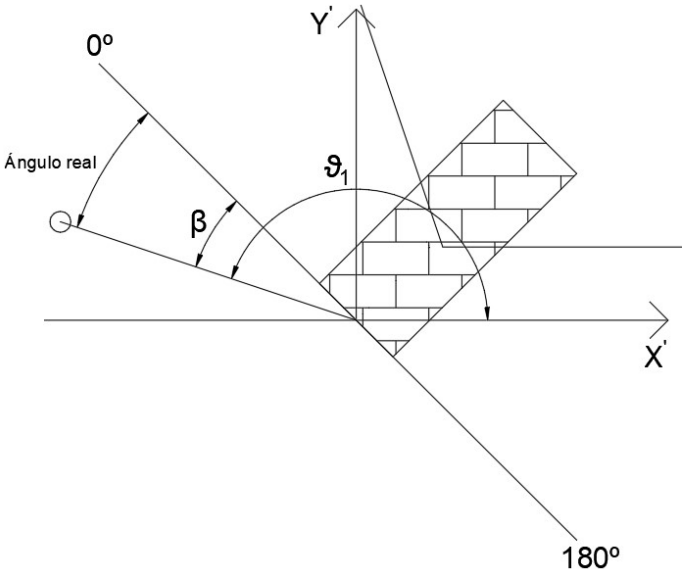


Figura 76: Resolución θ1 Pata 3 B

$$\theta_1 = Tg^{-1}\left(\frac{-y}{-x}\right)$$

$$\beta = 180º - |\theta_1|$$

$$Ángulo\ real = \Delta 45º + \beta + 0º$$

$$\theta_1 = Tg^{-1}\left(\frac{y}{-x}\right)$$

$$\beta = \theta_1 - 3 * \Delta 45º$$

$$Ángulo\ real = \beta + 0º$$

- Pata 4

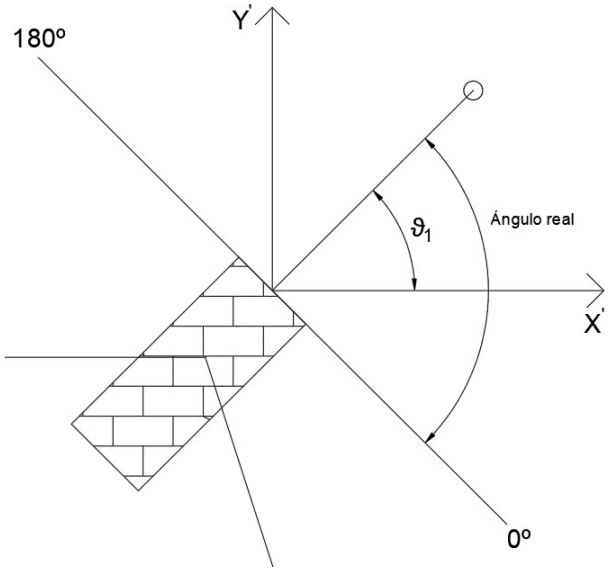


Figura 77: Resolución θ1 Pata 4

$$\theta_1 = Tg^{-1}\left(\frac{y}{x}\right)$$

$$Ángulo\ real = 0º + \Delta 45º + \theta_1$$

- Pata 5

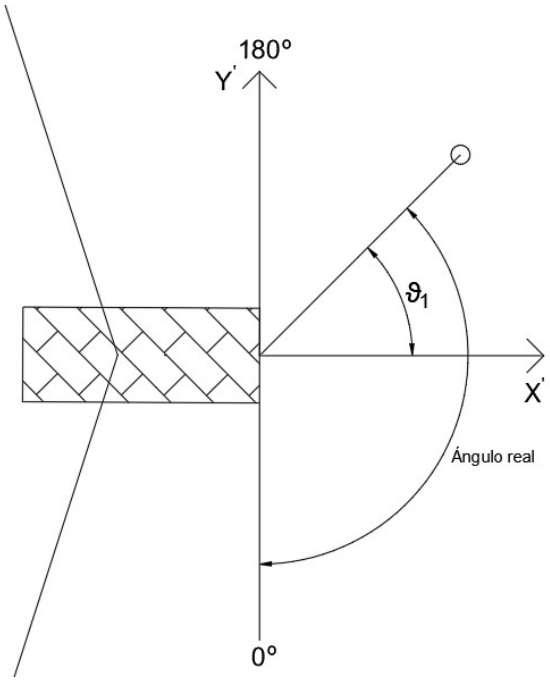
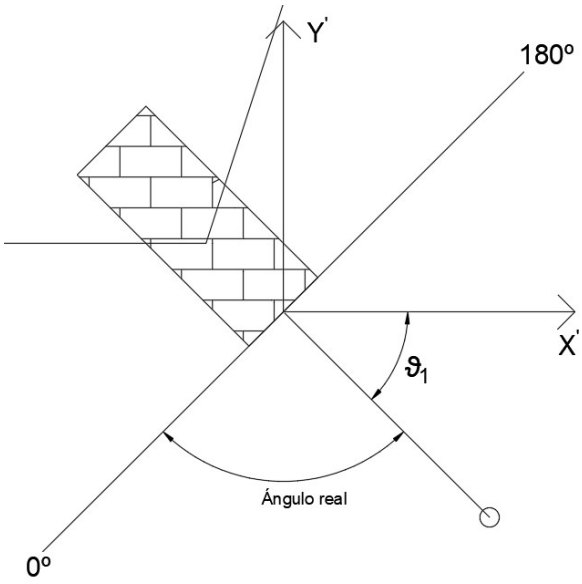


Figura 78: Resolución θ1 Pata 5

$$\theta_1 = Tg^{-1}\left(\frac{y}{x}\right)$$

$$Ángulo\ real = 0º + \Delta 90º + \theta_1$$

- Pata 6



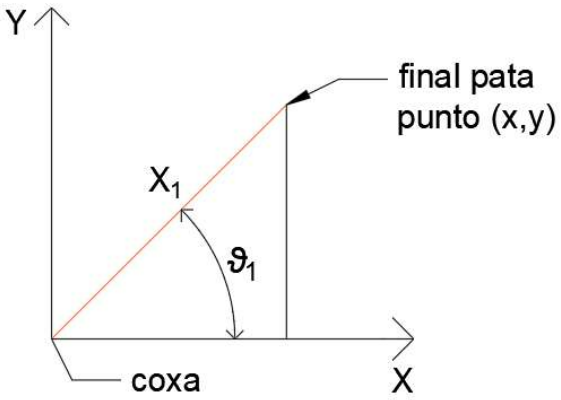
$$\theta_1 = Tg^{-1}\left(\frac{-y}{x}\right)$$

$$Ángulo\ real = 0º + 3 * \Delta 45º + \theta_1$$

Figura 79: Resolución θ1 Pata 6

4.3.1.2 Resolución ángulo teta 2

Una vez resuelto el ángulo coxa nuestro problema se reduce a una resolución en dos dimensiones, por lo que analizaremos la pata en un hipotético plano z', x'.



La distancia X_1 se mide desde el eje de rotación coxa hasta el punto final de la pata y representa la longitud total de la pata proyectada al plano xy.

$$\boldsymbol{X_1 = \sqrt{x^2 + y^2}}$$

Figura 80: Longitud característica X1

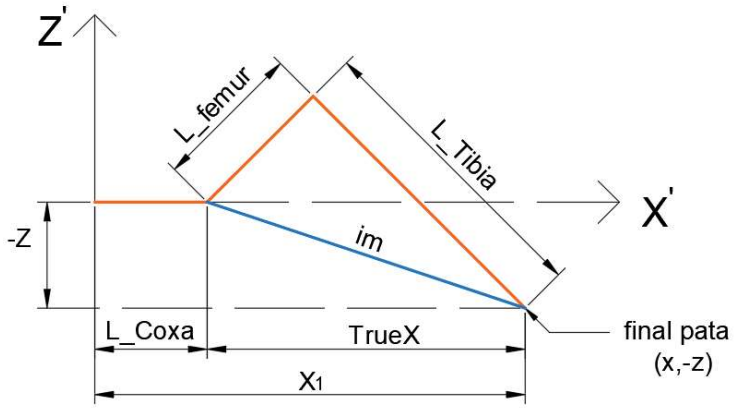


Figura 81: Diagrama lateral pata

Ahora trabajaremos con el triángulo formado por los ejes de rotación fémur y tibia, y con el punto final de la pata, formándose 2 medidas características.

$$\boldsymbol{TrueX = X_1 - L\_Coxa}$$

$$\boldsymbol{im = \sqrt{(TrueX)^2 + (-z)^2}}$$

$\boldsymbol{TrueX}$ Es la distancia lineal no alineada desde el punto final de la pata hasta el eje de rotación fémur.

$\boldsymbol{im}$ Es una distancia ficticia desde el eje de rotación fémur y el final de la pata.

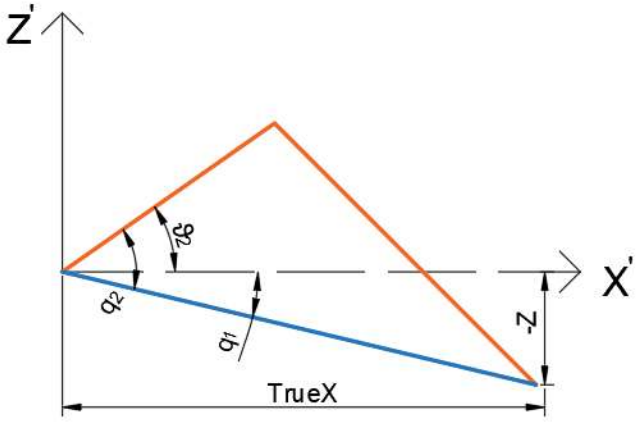


Figura 82: Resolución θ2

El ángulo q1 viene dado por:

$$q_1 = tg^{-1}\left(\frac{-z}{TrueX}\right)$$

Mientras que Z sea negativo, el ángulo saldrá negativo, cuando la pata sobrepase el eje horizontal, el ángulo será positivo.

Usando el teorema de los cosenos:

$$q_2 = cos^{-1}\left(\frac{(L\_Tibia)^2 - (L\_Femur)^2 - (im)^2}{2 * L\_Femur * im}\right)$$

Finalmente: $\theta_2 = q_2 + q_1$

Todos los servos fémur trabajan de la misma manera por lo que no hay ninguna variación entre ellos. Esto es debido a que el ángulo fémur se calcula en un plano ficticio, donde todas las medidas son iguales para todas las patas, por lo que no importa la orientación de los ejes.

Finalmente a partir del ángulo θ_2 obtenemos el "Ángulo real" que hay que mandarle al servomotor para que se oriente de forma correcta.

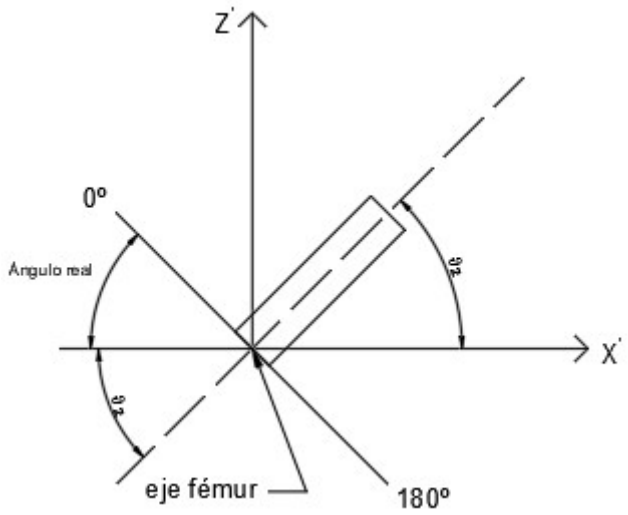


Figura 83: Ángulo real fémur

$$Ángulo\ real = 0º + \Delta 90º - \theta_2$$

El ángulo θ_2 pasa de positivo a negativo cuando pasa por el eje X', ajustándose.

4.3.1.3 Resolución ángulo teta 3

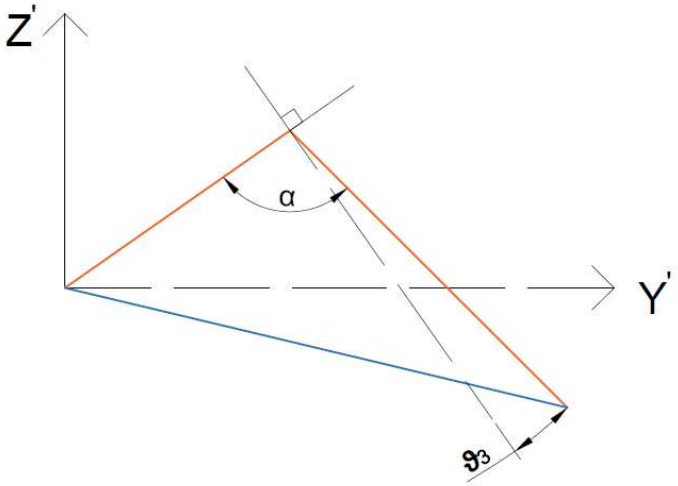


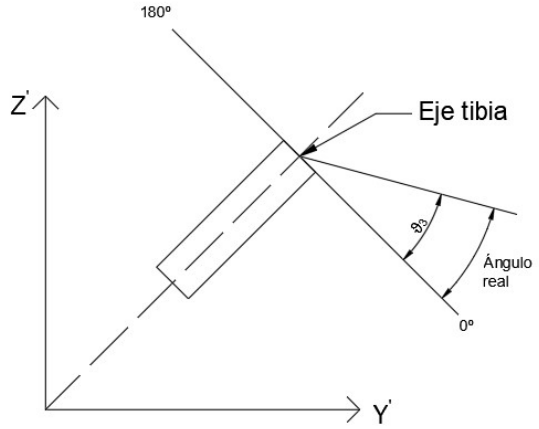
Figura 84: Resolución θ3

$$\theta_3 = \alpha - 90º$$

Usando el teorema de los cosenos:

$$\alpha = cos^{-1}\left(\frac{(im)^2 - (L\_Femur)^2 - (L\_Tibia)^2}{-2 * L\_Femur * L\_Tibia}\right)$$

Por el mismo motivo que en los servos fémur, los servos Tibia se orientan todos de la misma manera. El "Ángulo real" correspondiente es el siguiente.



$$Ángulo\ real = 0º + \theta_3$$

Figura 85: Ángulo real tibia

En este punto nos encontramos con otro problema, y es que debido a que estos servos digitales solo pueden tener un rango de movimiento de 180º, vemos restringida la capacidad de movimiento de las patas.

Para solucionar esto, cuando el servo esté posicionado en 0º, físicamente se coloca la pieza de manera que ganamos un ángulo β, obteniendo más rango de movimiento y pudiendo contraer más las patas hacia el cuerpo; por contrapartida perdemos rango de movimiento por la parte superior, lo cual para nosotros es irrelevante ya que no queremos la pata en el aire, sino en el suelo.

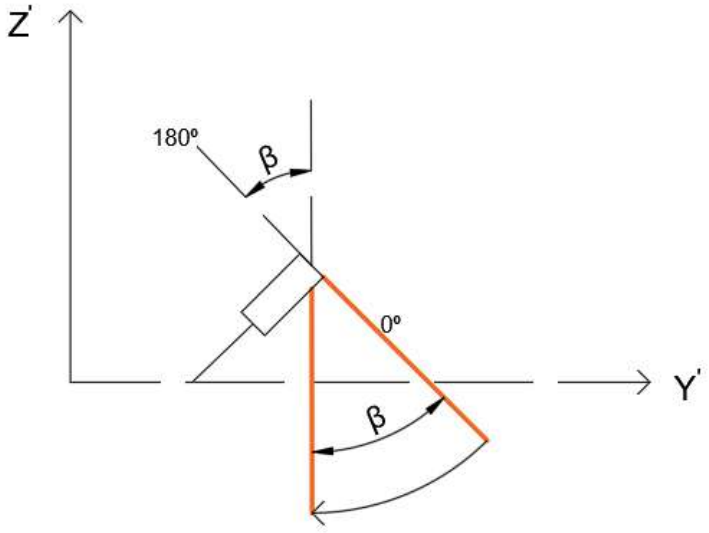


Figura 86: Ángulo tibia adicional

Lógicamente si se realiza esto, la referencia de PWM que teníamos para 0º o 90º cambia, por lo que tendremos que volver a realizar el ajuste entre PWM y posición física, asignando los nuevos valores correspondientes.

4.3.2 Matrices de transformación

Para que cada extremo de cada pata del robot se mueva describiendo una trayectoria, es necesario aplicar al punto una serie de operaciones de rotación y traslación para ubicarlo en la posición deseada.

Para ello se utilizarán las llamadas matrices de transformación homogénea. Estas matrices están compuestas por submatrices que realizan distintas operaciones.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ Perspectiva & Escalado \end{bmatrix}$$

Una matriz de transformación homogénea se compone de una submatriz R_{3x3} que corresponde a una matriz de rotación; una submatriz p_{3x1} que corresponde al vector de traslación; una submatriz f_{1x3} que representa una transformación de perspectiva y que en nuestro caso será cero, y una submatriz w_{1x1} que representa un escalado global y que es la unidad puesto que no queremos escalado.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ 0 & 1 \end{bmatrix}$$

Las matrices de rotación pueden componerse para expresar la aplicación continua de varias rotaciones. Hay que tener en cuenta el orden en que se realizan las rotaciones, pues el producto de matrices no es conmutativo.

$$T = R(z,\theta)R(y,\phi)R(x,\alpha) = \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C\phi & 0 & S\phi \\ 0 & 1 & 0 \\ -S\phi & 0 & C\phi \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} =$$

$$= \begin{bmatrix} C\theta C\phi & -S\theta C\alpha + C\theta S\phi S\alpha & S\theta S\alpha + C\theta S\phi C\alpha \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha \\ -S\phi & C\phi S\alpha & C\phi C\alpha \end{bmatrix}$$

Donde $C\theta$ se expresa $cos\theta$ y $S\theta$ expresa $sen\theta$.

De esta forma, si le aplicamos al vector que representa el extremo de la pata la siguiente matriz de transformación homogénea, la cual aplica al vector una rotación sobre los 3 ejes coordenados y una traslación en el espacio, resultando:

$$\begin{bmatrix} \boldsymbol{r_x} \\ \boldsymbol{r_y} \\ \boldsymbol{r_z} \\ \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} C\theta C\phi & -S\theta C\alpha + C\theta S\phi S\alpha & S\theta S\alpha + C\theta S\phi C\alpha & p_x \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha & p_y \\ -S\phi & C\phi S\alpha & C\phi C\alpha & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} p_x + \mathrm{C}\theta\mathrm{C}\phi r_u + (-\mathrm{C}\alpha\mathrm{S}\theta + \mathrm{C}\theta\mathrm{S}\alpha\mathrm{S}\phi) r_v + (\mathrm{S}\alpha\mathrm{S}\theta + \mathrm{C}\alpha\mathrm{C}\theta\mathrm{S}\phi) r_w \\ p_y + \mathrm{C}\phi\mathrm{S}\theta r_u + (\mathrm{C}\alpha\mathrm{C}\theta + \mathrm{S}\alpha\mathrm{S}\theta\mathrm{S}\phi) r_v + (-\mathrm{C}\theta\mathrm{S}\alpha + \mathrm{C}\alpha\mathrm{S}\theta\mathrm{S}\phi) r_w \\ p_z - \mathrm{S}\phi r_u + \mathrm{C}\theta\mathrm{S}\alpha r_v + \mathrm{C}\alpha\mathrm{C}\phi r_w \\ 1 \end{bmatrix}$$

Esta matriz resultante es la que tendremos que implementar en nuestro código para trasladar y rotar los puntos que describirán las trayectorias de las patas.

En el siguiente ejemplo se puede ver la transformación que sufre un punto de una pata cualquiera, rotándolo y trasladándolo respecto del centro del robot. Una vez hecha la transformación del punto, el parámetro que se le tiene que pasar a la cinemática inversa es el vector resultante de restar el punto del final de la pata con el punto coxa de la misma, para calcular las coordenadas articulares.

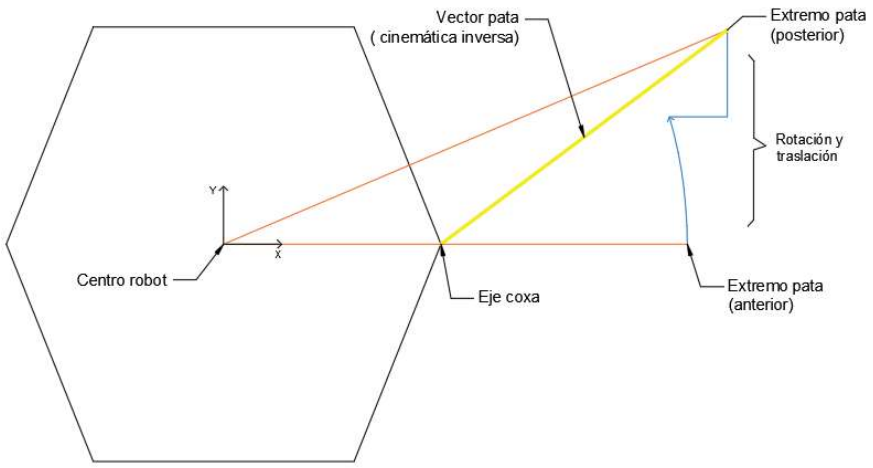


Figura 87: Aplicación de Matriz de transformación homogénea

4.3.3 Trayectorias

En los apartados anteriores se ha analizado cómo pasar del vector que representa la envergadura de la pata, a sus respectivas coordenadas articulares para que las patas se orienten de la forma adecuada. También cómo se puede transformar ese punto, rotándolo y trasladándolo para dotar de movimiento al extremo de la pata.

Estos eran pasos previos necesarios para poder diseñar las trayectorias necesarias para el movimiento, ya que éstas no son otra cosa que una sucesión de puntos en un orden concreto.

La trayectoria de la pata se caracteriza por dos fases. Una en la cual la pata se levanta del suelo y avanza hacia delante, llamada **fase de avance** o recuperación, y otra en la cual la pata se encuentra apoyada en el suelo y empuja o se retrae hacia detrás, llamada **fase de empuje.**

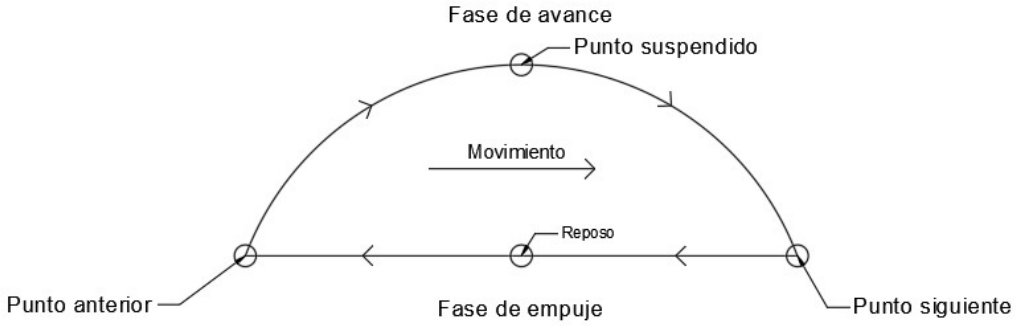


Figura 88: Fases de la trayectoria

Todos los modos de caminar poseen un punto suspendido, por el que pasan en la fase de avance, para llegar hasta el punto siguiente. Un punto anterior al cual acaba la pata en su fase de empuje; y por último un punto de reposo que está en mitad de la fase de empuje.

4.3.3.1 Fases de desplazamiento

Para el avance del robot, éste se sirve de sus patas para impulsar el cuerpo hacia delante. Éstas tienen que trazar una trayectoria que se distingue por varias fases las cuáles vamos a ver a continuación.

4.3.3.1.1 Reposo

Es la posición intermedia entre el punto siguiente y el punto anterior y en al cual se encuentra el robot cuando no está en movimiento.

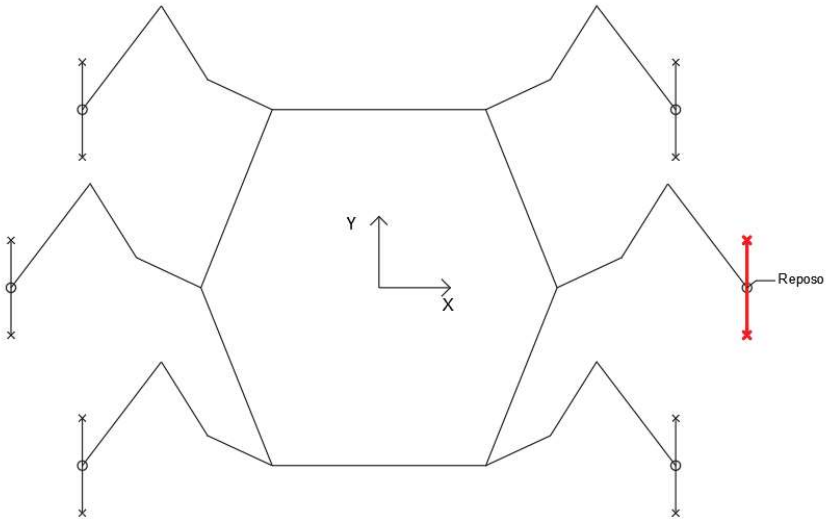


Figura 89: Estado de reposo

4.3.3.1.2 Avance

En esta primera fase de avance, como partimos de la posición de reposo solo avanzamos la mitad de la distancia de avance de la trayectoria.

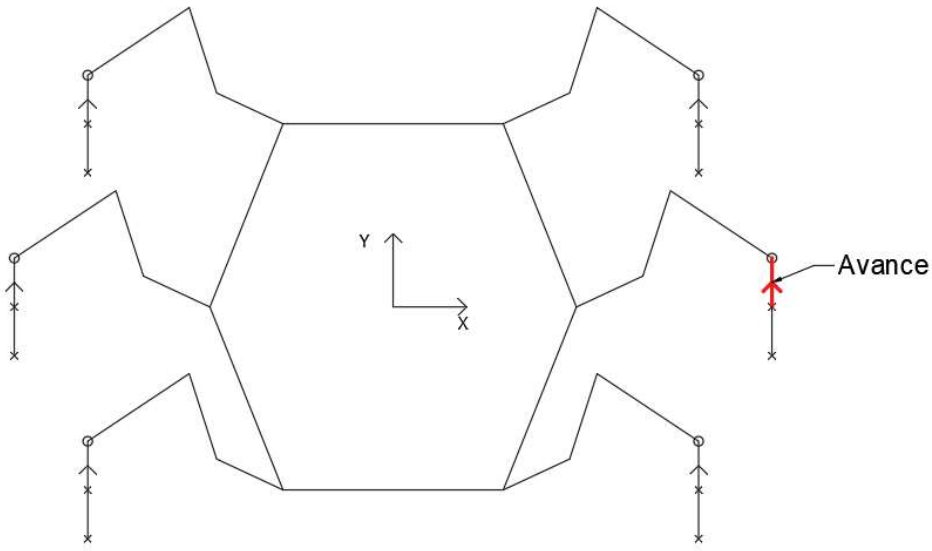


Figura 90: Mitad de avance

4.3.3.1.3 Empuje total

Esta es la fase en la cual se produce el verdadero movimiento del robot. En ésta fase, las patas se mueven empujando al cuerpo para que este avance hacia delante.

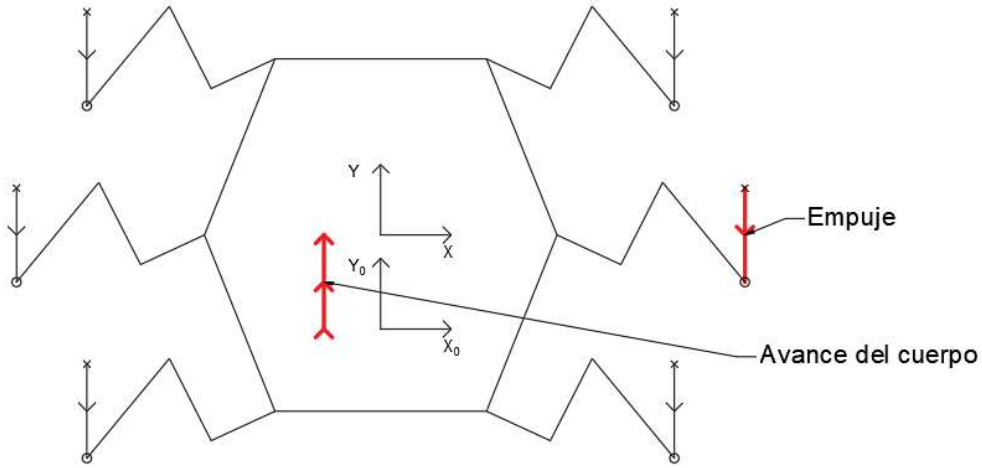


Figura 91: Empuje del robot

4.3.3.1.4 Avance total

Cuando el robot ya se encuentra en pleno movimiento y se ha alcanzado el punto más retrasado de la trayectoria, se procede al levantamiento de la pata para realizar de nuevo la fase de avance pero ahora con toda su amplitud.

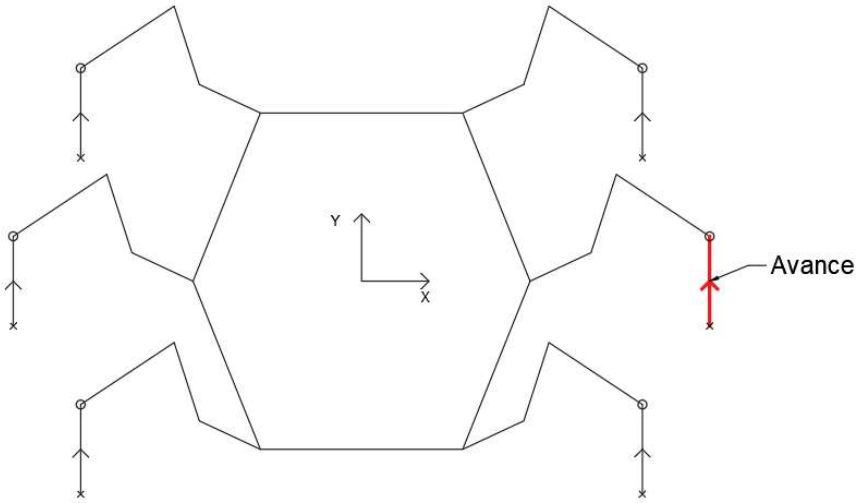


Figura 92: Avance completo

4.3.3.2 Fases de giro

Para el giro del robot el principio es el mismo, el movimiento se basa en impulsarse con las patas, el único cambio es que ahora la trayectoria incluye un giro de la dirección del robot. En el ejemplo expuesto el cuerpo del robot giraría un total de 20º.

4.3.3.2.1 Reposo

Se parte de la misma posición de reposo.

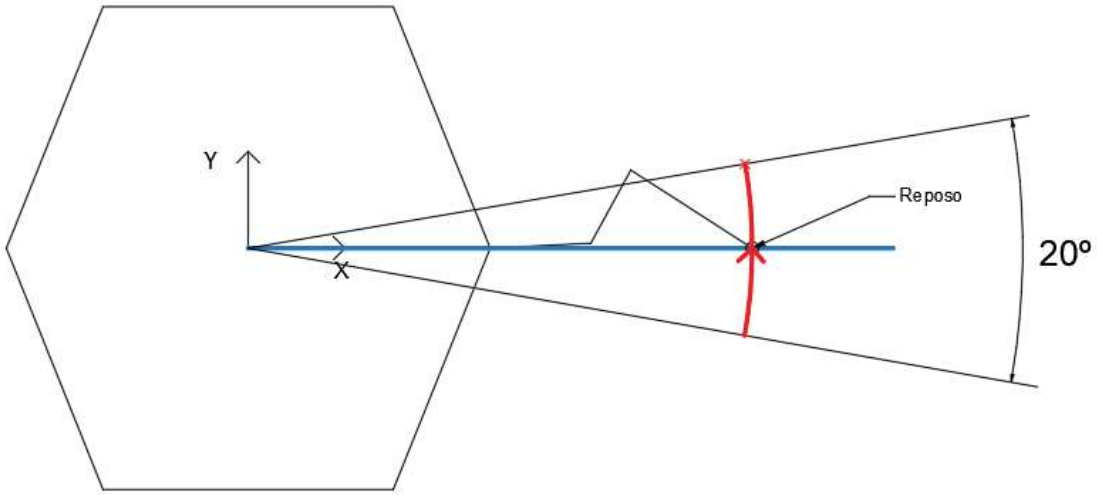


Figura 93: Reposo

4.3.3.2.2 Avance

Primero se hace que avance la pata, para luego empujar con ella al cuerpo.

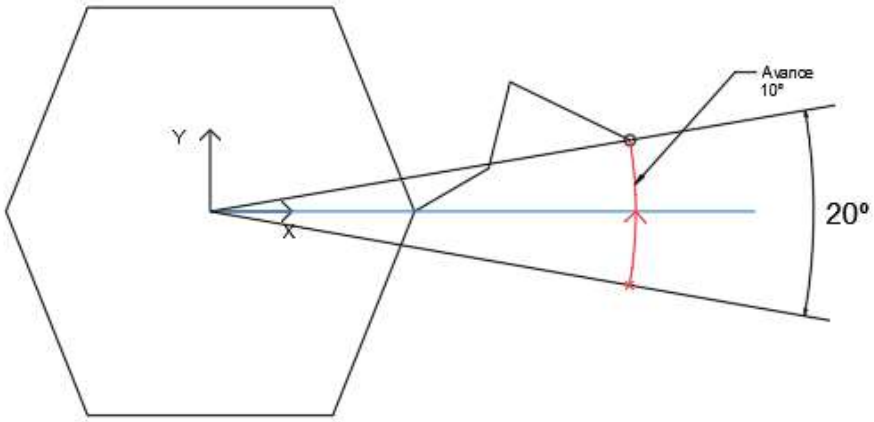


Figura 94: Mitad de avance

4.3.3.2.3 Empuje total

La pata empuja al cuerpo para que este avance 20º.

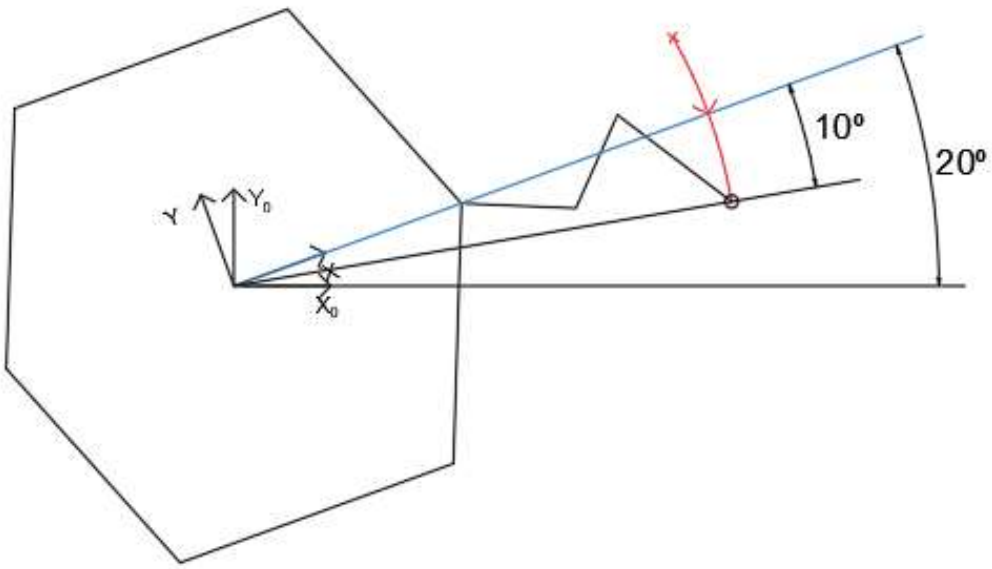


Figura 95: Empuje del cuerpo

4.3.3.2.4 Avance total

Una vez el cuerpo avanza, la pata vuelve a desplazarse hacia delante para avanzar otros 20º.

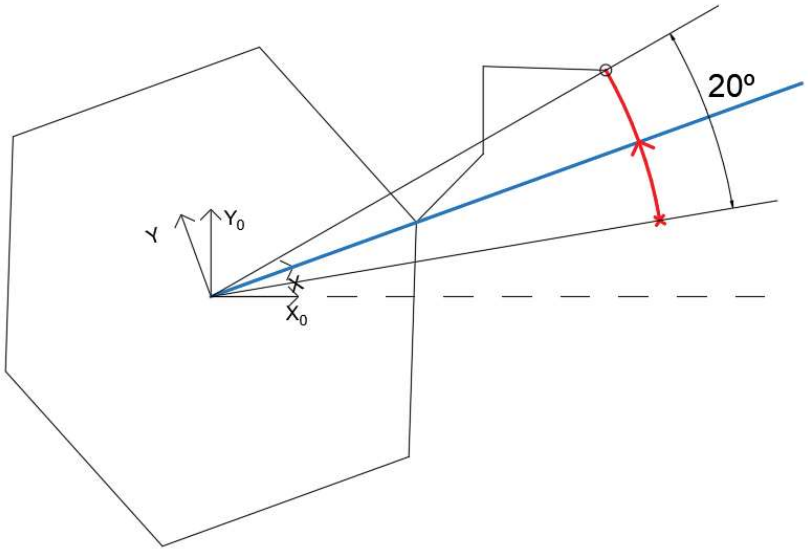


Figura 96: Avance completo

4.3.3.3 Parametrización

Para tener un control sobre el desplazamiento del robot, las trayectorias realizadas tienen que depender de unas determinadas variables. De esta forma se podrá controlar la velocidad con la que se reproduce la trayectoria, la distancia recorrida o incluso la forma de ésta.

$N_{pasos\ ciclo}$	Cantidad de pasos que posee una trayectoria para completarse y que variará dependiendo de la trayectoria realizada
$t_{paso}(s)$	Tiempo necesario para la duración de un paso(función de la velocidad de actuación de los servos)
$t_{ciclo}\ \left(\frac{s}{ciclo}\right) = t_{paso}(s) * N_{pasos\ ciclo}$	Tiempo que tarda la trayectoria en realizarse
$Altura_{suspendida}$ (mm)	Altura levantada en los puntos suspendidos

Tabla 21: Parámetros de la trayectoria 1

En el suelo, la distancia entre pasos se mantiene constante para que se pueda definir una velocidad de camino. Sin embargo, no ocurre lo mismo con los pasos suspendidos, ya que estos tienen una determinada altura. No obstante, entre todos los pasos, ya estén elevados o no, siempre se mantiene constante el tiempo entre ellos.

Cabe destacar que **los pasos son las transiciones** entre los puntos de la trayectoria. Como se puede ver en la siguiente figura.

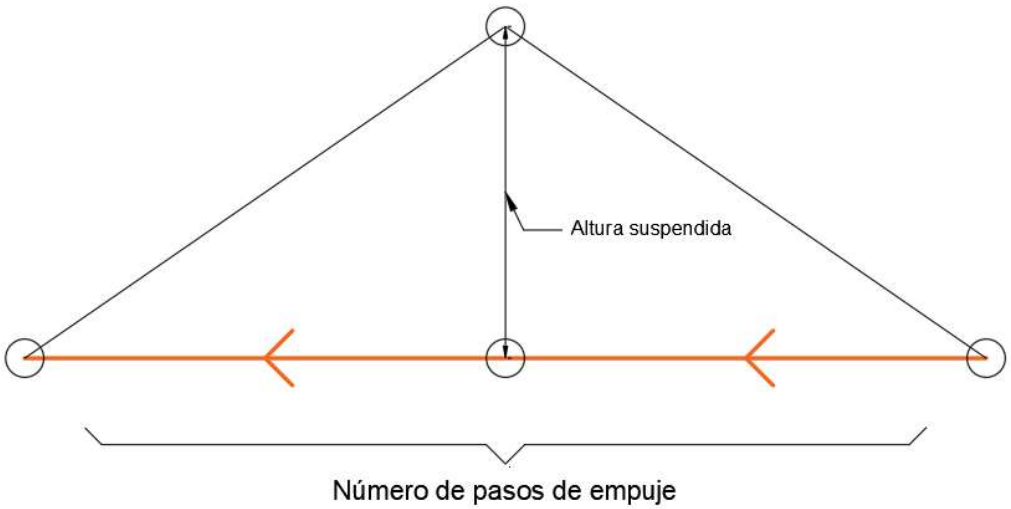


Figura 97: Pasos en una trayectoria simple

$N_{pasos\ empuje}$: $Cantidad\ de\ pasos\ de\ la\ trayectoria\ en\ contacto\ con\ el\ suelo$
$N_{pasos\ suspendidos}$: $Cantidad\ de\ pasos\ de\ la\ trayectoria\ en\ el\ aire$

Figura 98: Parámetros para fase de empuje y avance2

Por lo tanto se puede definir un factor de trabajo definido por el cociente entre el número de pasos en el suelo, partido el número de pasos totales definidos en la trayectoria, para saber que porción del ciclo los pasamos en el suelo.

$$Factor\ trabajo = \frac{N_{pasos\ empuje}}{N_{pasos\ ciclo}}$$

La velocidad es un parámetro que viene dado a partir del valor que tome el joystick, y el cual tiene que estar modelado para dar un valor lógico de movimiento. Dependiendo de la posición de éste, la velocidad se descompondrá en una componente en el eje x y otra en el eje y.

$$Velocidad\ \left(\frac{mm}{s}\right) : Velocidad\ de\ avance\ en\ el\ eje\ en\ cuestión$$

De esta forma la distancia recorrida en la fase de empuje viene dada por:

$$Distancia_{\ empuje}\ (mm) = Velocidad * \ t_{ciclo} * Factor\ trabajo$$

La cual tendrá dos componentes al igual que la velocidad.

Siendo la distancia por paso la mostrada a continuación, y que estará definida en cada eje, dependiendo de las componentes de la variable velocidad.

$$Distancia_{\ paso}\ (mm) = Velocidad\ * \ t_{paso}$$

Así, por cada paso suspendido, se deberá recorrer una determinada distancia en cada eje, dependiendo del número de pasos suspendidos que haya y de la distancia de empuje, tal que:

$$Distancia_{\ por\ pas\ \ suspendido}\ (mm) = \frac{Distancia_{\ empuje}}{N_{\ pasos\ suspendidos}}$$

En cuestión al giro del robot la metodología es la misma, el único factor distinto es la velocidad, la cual ahora se considerará una velocidad angular, obteniendo como resultado el ángulo girado en el empuje, cuya unidad será el radián.

$$Velocidad_{\ Giro\ Z}\ \left(\frac{rad}{s}\right) : Velocidad\ de\ giro\ sobre\ el\ eje\ Z$$

$$Ángulo_{\ Rotación\ Z}\ (rad) = Velocidad_{\ Giro\ Z} *\ t_{ciclo} * Factor\ trabajo$$

$$Ángulo_{\ por\ paso\ suspendido\ Rot\ Z}\ (rad) = \frac{Ángulo_{\ Rotación\ Z}}{N_{\ pasos\ suspendidos}}$$

Finalmente si se requiere una $Distancia_{\ empuje}\ = 100\ mm$ por ejemplo, como se sabe que el tiempo de ciclo y el factor de trabajo son constantes impuestas previamente; la única variable la cual podremos modelar a nuestro antojo para conseguir esta distancia, es la **velocidad**, la cual obtenemos despejando de la anterior ecuación.

$$Velocidad\ (máxima) = \frac{Distancia_{\ empuje}}{t_{ciclo} * Factor\ trabajo}$$

Ahora se tiene que establecer una proporción entre el valor obtenido del joystick y la velocidad correspondiente, limitada a los máximos inferior y superior, dependiendo si la velocidad es negativa o positiva (dependiendo del sentido).

4.3.3.3.1 Interpolación de posiciones de servomotores

La interpolación de posiciones consiste en introducir posiciones intermedias entre un paso del servomotor. Si el recorrido es lo suficientemente amplio para que tenga lógica introducir un paso intermedio entre las dos posiciones, esto consigue hacer el movimiento más suave y continuo. Por otra parte haciendo sucesivos movimientos más pequeños, en vez de unos muy grandes, se consigue mejorar el consumo de corriente del motor.

Como hemos definido anteriormente el t_{paso} es el tiempo dedicado para pasar de un punto a otro de la trayectoria, por lo que para la interpolación necesitamos dedicar un $\boldsymbol{Tiempo_{base}}$ el cual tiene que ser el tiempo del paso dividido el número n de pasos intermedios que necesitemos. Este tiempo característico no puede ser menor que el tiempo de respuesta del servomotor, ni tampoco tiene lógica avanzar en pasos tan cortos que sean menores que la precisión del servomotor.

Como se ha visto anteriormente, independientemente del eje considerado, tenemos que:

$$Distancia_{\,empuje} = Velocidad * t_{ciclo} * Factor\ trabajo$$

- Sustituyendo los valores de t_{ciclo} y $Factor\ trabajo$ por sus respectivas equivalencias:

$$Distancia_{\,empuje} = Velocidad * t_{paso} * N_{pasos\ ciclo} * \frac{N_{pasos\ empuje}}{N_{pasos\ ciclo}}$$

Nos queda:

$$Distancia_{\,empuje}\ (mm) = Velocidad * t_{paso} * N_{pasos\ empuje}$$

De esta última expresión observamos que la distancia recorrida es función de los anteriores parámetros. No obstante, $N_{pasos\ empuje}$ es constante dependiendo del modo de caminar en el que nos encontremos. De esto se deduce que en secuencias donde haya un mayor número de pasos, se recorrerá una mayor distancia. Entonces si se está en una misma secuencia sólo podemos aumentar la distancia recorrida si incrementamos la velocidad o el tiempo entre pasos.

Como es lógico si aumentamos t_{paso} la distancia que recorreremos será mayor pero también aumentaremos el tiempo que las patas están en el suelo, reduciendo la velocidad de marcha del robot pero, a su vez, aumentando la estabilidad del mismo, por lo que si queremos un movimiento más suave, sólo necesitamos aumentar esta variable. También, aumentando la velocidad se incrementa la distancia recorrida para un mismo tiempo de ciclo. Por lo que dependiendo de lo que se busque, se deberá llegar a un **compromiso entre velocidad y tiempo de paso**, para no llegar a la distancia de empuje crítica, en la que se choquen unas patas con otras, o no alcancen las patas hasta esa posición. Si se aumenta mucho el tiempo de paso, tendremos que reducir la velocidad y viceversa, para no sobrepasar esta distancia que es alrededor de unos 12 cm.

4.3.3.3.2 Velocidad límite

La velocidad teórica máxima para nuestros servomotores es de 0.16 segundos por cada 60º recorridos, por lo que pasando esta velocidad a radianes por segundo:

$$Velocidad\ angular\ máxima = 6.545\ \left(\frac{rad}{s}\right)$$

Las patas tiene una envergadura de unos 12 cm las más cortas, por lo que si el trayecto realizado son 10 cm en total, de aquí se deduce que el ángulo recorrido por la articulación coxa es de 45º, resultando este ángulo como el mayor de todas las articulaciones(los otros son más pequeños). Por lo que ésta tomará el mayor tiempo necesario por articulación para alcanzar el punto esperado.

Este tiempo es el máximo que hay que esperar para que todas las articulaciones se encuentren posicionadas en sus respectivas orientaciones. Por lo que será una trayectoria punto a punto, cuyo movimiento de articulaciones es simultáneo, no realizándose el siguiente movimiento hasta que todas las articulaciones no estén orientadas correctamente.

No obstante, esta velocidad es la máxima teórica sin carga, por lo que con carga le aplicaremos una reducción del 20% resultando en $5.236\ \left(\frac{rad}{s}\right)$

El tiempo necesario para completar el giro será:

$$t_{crítico} = \frac{45º}{5.236\left(\frac{rad}{s}\right)} * \frac{\pi\ rad}{180º} * \frac{1000\ ms}{1\ s} \cong 150\ ms$$

Este es el análisis del peor caso, en el cual tenemos que recorrer la mayor distancia posible y para la cual tardaremos el mayor tiempo posible, obteniendo el tiempo necesario para completar la fase de empuje. Empleando más tiempo hacemos al modo de caminar más lento pero más estable, y si lo reducimos corremos el riesgo de que la pata no alcance su posición a tiempo, teniendo ya que ir ya al siguiente punto, aumentando la inestabilidad.

Por tanto, el tiempo mínimo entre pasos será:

$$t_{paso\ mínimo} = \frac{t_{crítico}}{N_{pasos\ empuje}}$$

Siendo t_{paso} múltiplo de $Tiempo_{base}$ de forma que:

$$t_{paso} = N_{pasos\ intermedios} * Tiempo_{base}$$

Para nuestro robot el $Tiempo_{base}$ lo consideraremos con un valor de 20ms.

Finalmente, viendo una aplicación, si una articulación pasa de un nivel PWM de 200 a otro de 400, en un $t_{paso} = 80ms$, dividiendo por el tiempo base obtenemos que este incremento se divide en 4 partes de 20 ms cada una. Esto pasa con cada articulación y pata para cada respectivo incremento de PWM.

Por lo tanto en cada parte se tiene que incrementar:

$$\frac{\Delta 200\ niveles\ pwm}{4\ partes} = 50\ \frac{niveles\ pwm}{parte}$$

Como se ha visto, para completar la trayectoria entre un punto y el siguiente hay que esperar un tiempo $t_{paso\ mínimo}$ para el cual todas las articulaciones ya se encontrarán en sus respectivas orientaciones. Esto evita que los actuadores se fuercen en cuanto a las velocidades y aceleraciones innecesariamente. A este tipo de trayectoria se le denomina **trayectoria coordinada o isócrona** (2.2 Trayectorias coordinadas o isocronas).

4.3.3.4 Secuencias

Existen distintos tipos de patrones en el movimiento de las patas que consiguen un movimiento eficaz del robot, aunque hay algunas secuencias que son más estables que otras, lo que es debido a que cuanto más tiempo estén las patas levantadas del suelo, menos estable será el robot en su caminata. Por tanto, modos de caminar que mantengan el mayor número de patas en contacto con el suelo y durante el mayor tiempo posible serán las más estables.

En general, para cada tipo de patrón, todas las patas describen la misma trayectoria paso a paso. Sin embargo, dependiendo de la secuencia que estemos reproduciendo, cada pata estará desfasada un número $N\ de\ pasos$ una respecto de otra; esto es necesario para crear las secuencias que se muestran a continuación.

4.3.3.4.1 Ripple

Este tipo de secuencia es la segunda más estable ya que mantiene 4 patas tocando el suelo siempre.

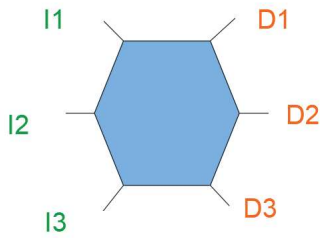


Figura 99: Grupos de patas ripple

Este patrón de caminata consiste en dos sucesiones de patas superpuestas, una desfasada con otra la mitad de la secuencia. En un grupo tenemos a las patas derechas (D3-D2-D1) y en el otro las patas izquierdas (I3-I2-I1). Cuando en el grupo derecho la pata D2 se encuentra en su paso suspendido e inicia el siguiente paso en el cual se posa en el suelo, la pata I3 inicia su paso de elevación, empezando su secuencia; por lo que quedan dos secuencias desfasadas medio periodo. De este modo, en este patrón de caminata dos patas coinciden moviéndose de manera simultánea.

Este patrón de movimiento se puede ver en la siguiente figura. La fase de avance está representada por una franja negra, mientras que la fase de empuje está representada por una línea discontinua.

Figura 100: Patrón Ripple

En la siguiente figura se puede apreciar mejor el movimiento. En ella, las patas derechas e izquierdas están diferenciadas por distinto color y podemos apreciar como los dos grupos forman una especie de onda que se propaga y que se repite continuamente. Este tipo de secuencia se caracteriza por tener siempre dos patas en el aire en cada paso.

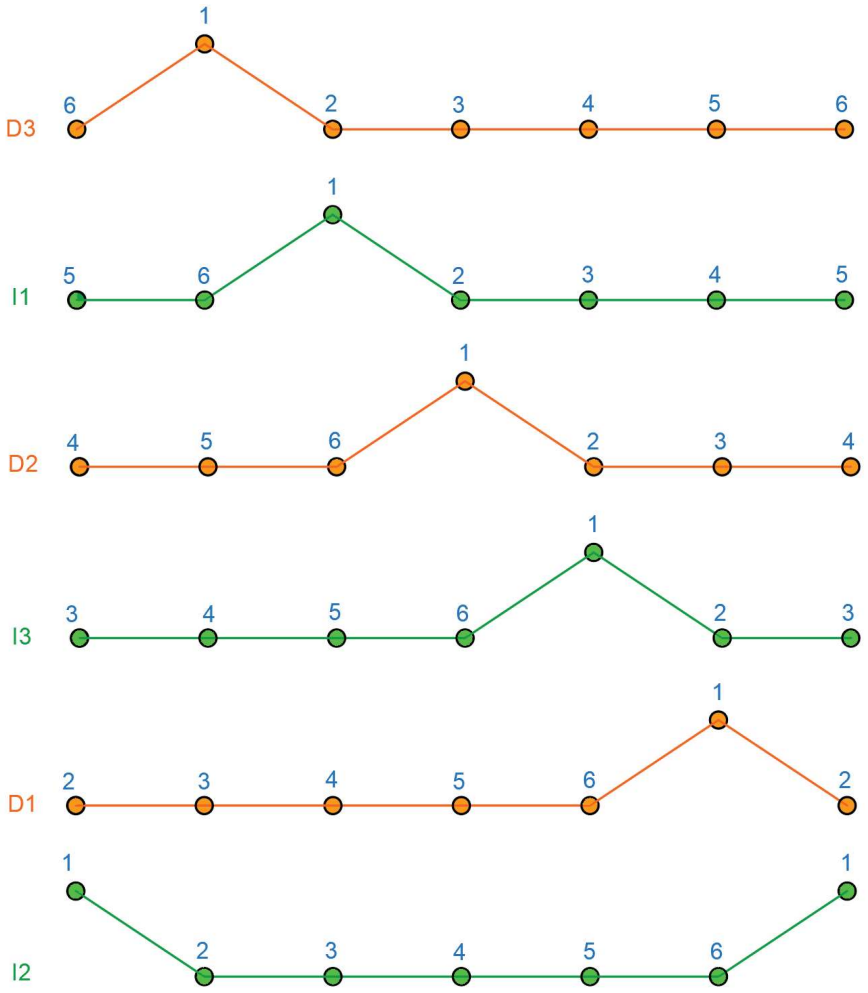


Figura 101: Secuencia Ripple 6 pasos

En la siguiente tabla se tienen los valores característicos para la secuencia Ripple según sus variantes:

Secuencias Ripple			
$N_{pasos\ ciclo}$	6	12	18
$N_{pasos\ empuje}$	4	8	12
$Desfase$	1	2	3

Tabla 22: Secuencias Ripple

4.3.3.4.2 Tripod

La marcha Tripod es un tipo de patrón en la cual se mueven tres patas simultáneamente. Hay 2 grupos de 3 patas, los cuáles están formados por patas no consecutivas (una sí y una no) en el mismo grupo. Mientras que un grupo de patas está en la fase de avance, el otro se encuentra en la fase de empuje.

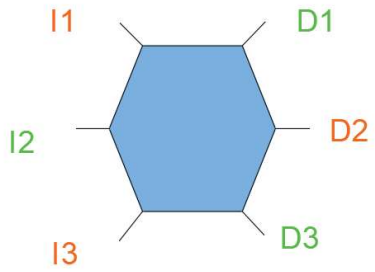


Figura 102: Grupos de patas tripod

Este tipo de secuencia es la menos estable ya que solo tiene 3 patas en contacto con el suelo siempre. De nuevo, el patrón lo podemos ver en la siguiente figura:

Figura 103: Patrón tripod

En esta figura podemos ver perfectamente los dos grupos de patas, en los cuales en cada grupo se mueven tres patas al unísono.

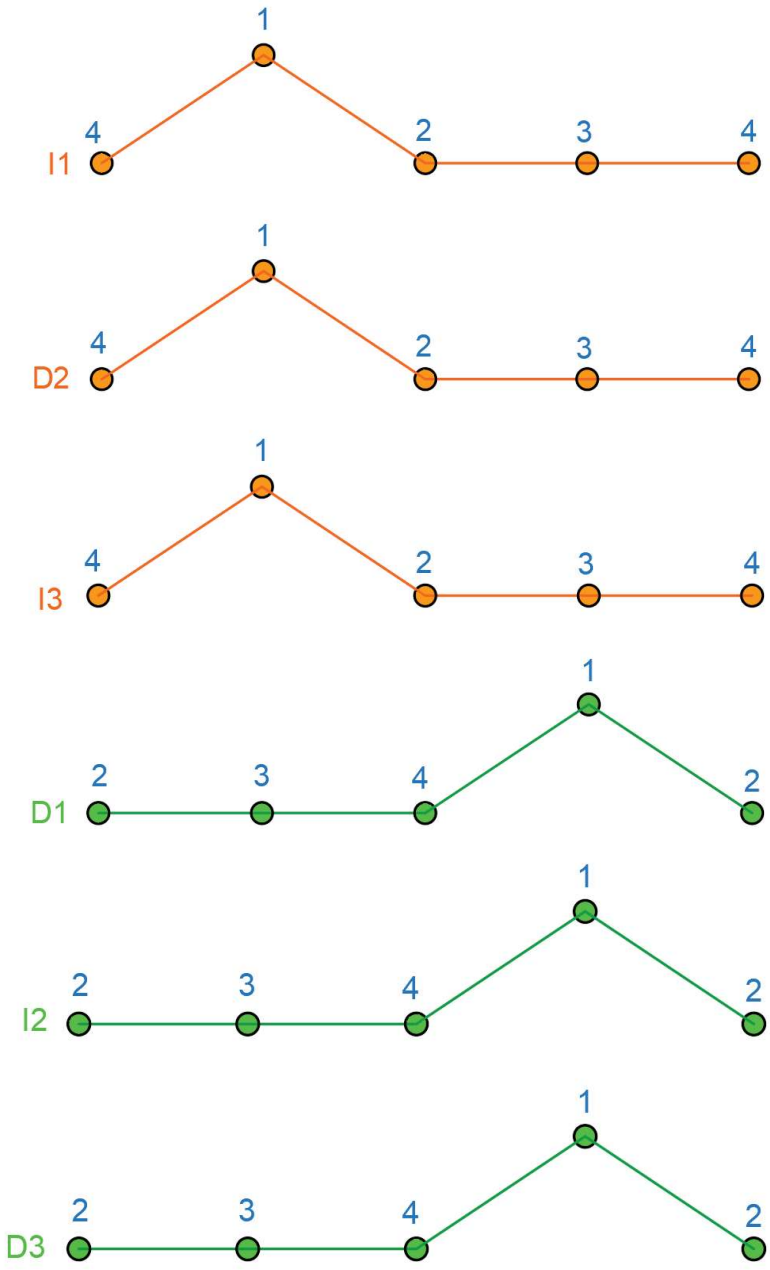


Figura 104:Secuencia Tripod 6 pasos

Secuencia Tripod			
$N_{pasos\ ciclo}$	4	8	12
$N_{pasos\ empuje}$	2	4	6
$Desfase$	2	4	6

Tabla 23: Parámetros secuencias Tripod

4.3.3.4.3 Wave

Este tipo de modo de caminar es el llamado modo de ola, ya que una pata se mueve después de otra consecutivamente. Cuando una pata está en fase de avance, todas las demás están en fase de empuje. Esta secuencia se considera como la más estable debido a que todas las patas menos una están en el suelo, aumentando la estabilidad, menos una que está en fase de avance. Esto hace que este tipo de caminata sea la más estable pero también la más lenta.

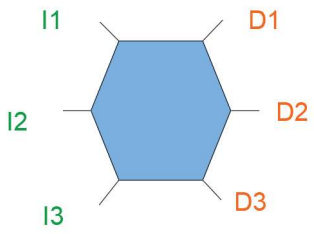


Figura 105: Grupos de patas wave

El patrón se muestra a continuación:

Figura 106: Patrón Wave

En esta ocasión se observa que para que esta secuencia pueda ser reproducida tenemos que aumentar el número de puntos de la trayectoria a doce, ya que en este tipo de secuencia no hay movimientos simultáneos de patas.

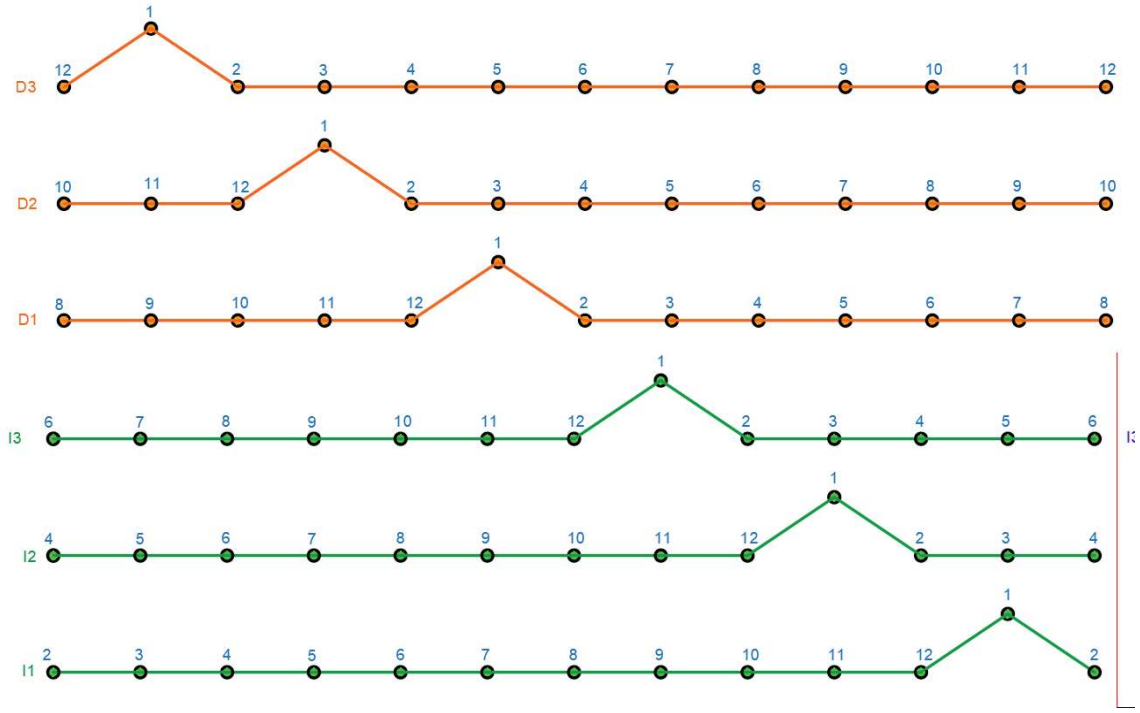


Figura 107: Secuencia Wave 12 pasos

Secuencia Wave		
$N_{pasos\ ciclo}$	12	24
$N_{pasos\ empuje}$	10	20
$Desfase$	2	4

Tabla 24: Parámetros secuencias Wave

4.4 Movimiento del cuerpo

En el caso del movimiento del cuerpo del robot se tiene dos opciones; la traslación o la rotación del mismo respecto de uno de los ejes coordenados. Estas dos opciones implican una misma resolución, la cual consistirá en la aplicación de la matriz de transformación homogénea a los puntos coxa de cada pata. De este modo podemos rotar o trasladar estos puntos para conseguir que el cuerpo gire o se desplace; todo esto manteniendo las patas en la misma posición.

Una vez aplicada la matriz de transformación a los puntos coxa, el único parámetro que tenemos que proporcionarle a la cinemática inversa para que la pata se oriente de manera adecuada, es el vector formado por el punto del extremo de la pata y el punto coxa.

- Traslación

Como vemos en la siguiente imagen, para trasladar el robot sólo tenemos que incrementar la componente x o y de los puntos coxa, para que el cuerpo se desplace sobre el plano xy, aunque también se puede variar la coordenada z, resultando en un incremento o disminución de la altura del robot

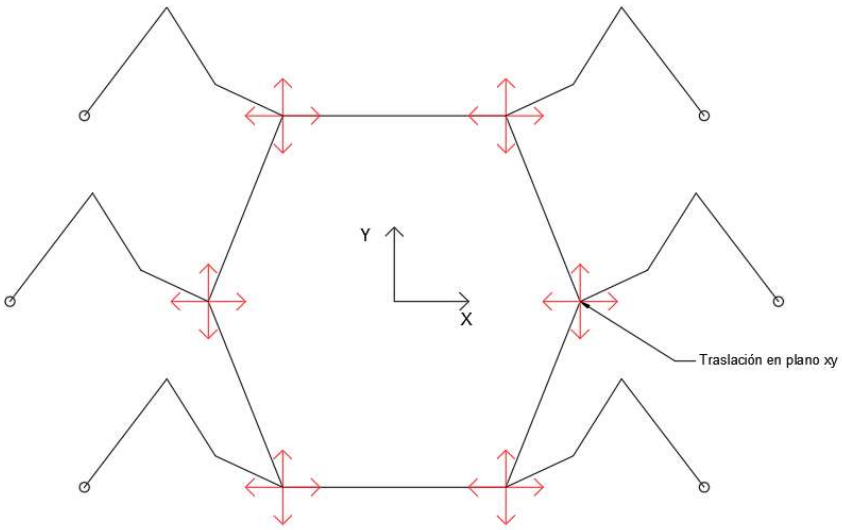


Figura 108: Traslación del cuerpo del robot

- Rotación

En las siguientes ilustraciones se muestran las posibles rotaciones del cuerpo del robot sobre los ejes coordenados, pudiendo rotar en sentidos horario y antihorario.

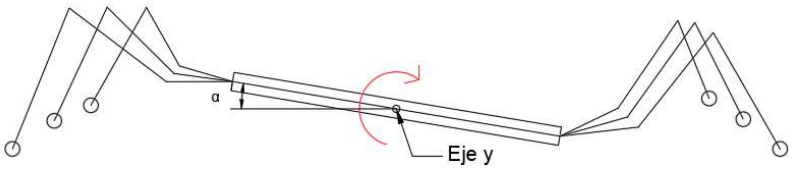


Figura 109: Rotación sobre el eje y

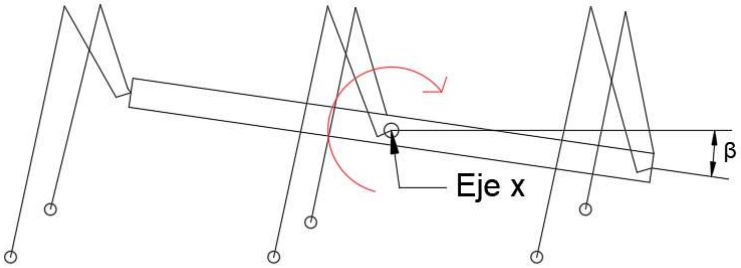


Figura 110: Rotación sobre el eje x

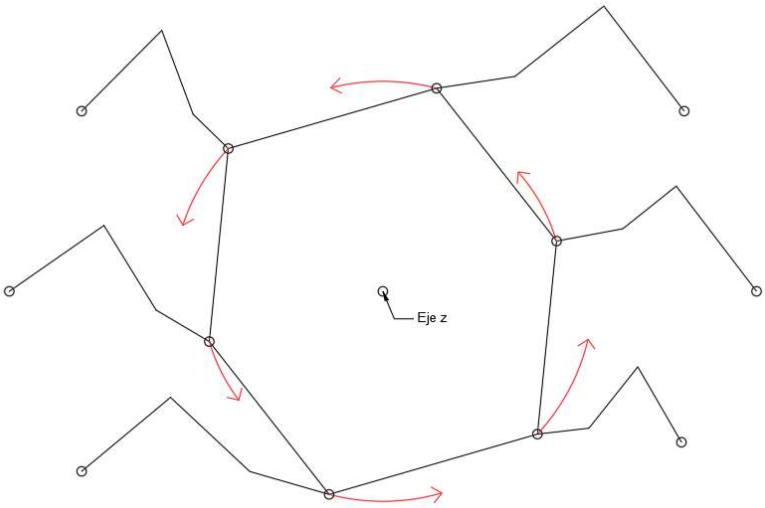


Figura 111: Rotación sobre el eje z

4.5 Estados predefinidos

Los estados predefinidos son posiciones específicas que puede adoptar el robot con sus patas. Estas han sido previamente almacenadas en la memoria del Arduino y consisten en una serie de grupos de valores que representan las orientaciones que necesitan adoptar los servomotores para conseguir este posicionamiento del robot.

Así se pueden guardar distintas posiciones útiles para distintas situaciones. Por ejemplo para la inicialización del robot, que el robot vaya reproduciendo en secuencia varios grupos de posiciones para conseguir ubicarse en la posición de inicio de manera correcta.

4.6 Comandos del mando

En este apartado se mostrarán las funciones asignadas para cada botón del mando, con el objetivo de controlar el movimiento omnidireccional del robot.

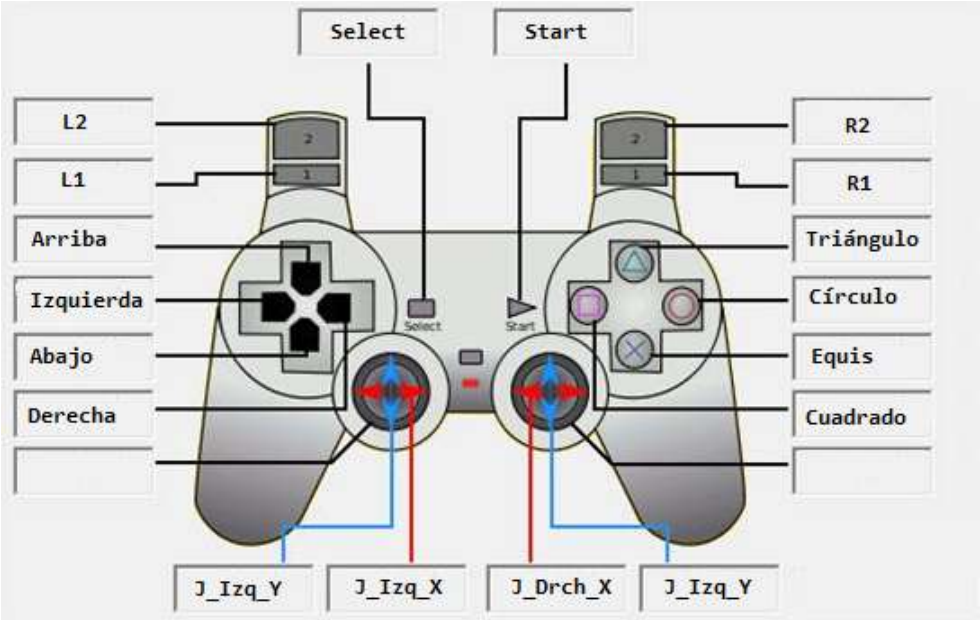


Figura 112: Configuración botones mando PS2

Los botones más importantes son el de *start* y el de *select,* el primero nos permite cambiar entre los modos de movimiento del cuerpo y desplazamiento del robot; mientras que con el segundo cambiaremos el tipo de secuencia para el desplazamiento del robot. A parte de esto, con los joystick controlaremos la dirección y sentido del movimiento del robot.

4.6.1 Modo Desplazamiento robot

En este modo el robot se mueve libremente por el suelo, pudiendo además girar y aumentar su altura respecto del suelo.

Comando	Función
Triángulo	-
Círculo	-
Equis	-
Cuadrado	-
Arriba	-
Izquierda	Disminuimos los pasos de la secuencia
Abajo	-
Derecha	Aumentamos los pasos de la secuencia
Select	Cambiamos de secuencia
Start	Cambiamos de modo
R1	Aumentamos la altura del robot
R2	disminuimos la altura del robot
L1	Aumentamos la altura de los pasos
L2	disminuimos la altura de los pasos
Joystick izquierdo eje Y	Desplazamiento hacia delante (positivo) y hacia atrás (negativo)
Joystick izquierdo eje X	Desplazamiento a la izquierda(negativo) o derecha (positivo)
Joystick derecho eje Y	-
Joystick derecho eje X	Giro horario en el sentido positivo del eje, y antihorario en el sentido opuesto

Tabla 25: Comandos desplazamiento robot

4.6.2 Modo control del cuerpo

En el modo de control del cuerpo se maneja el giro y traslación del robot sobre los tres ejes coordenados. Los botones asignados para el control del cuerpo del robot son los siguientes:

Comando	Función
Triángulo	Traslación positiva del cuerpo sobre el eje Y
Círculo	Traslación positiva del cuerpo sobre el eje X
Equis	Traslación negativa del cuerpo sobre el eje Y
Cuadrado	Traslación negativa del cuerpo sobre el eje Y
Arriba	-
Izquierda	-
Abajo	-
Derecha	-
Select	-
Start	Cambiamos de modo
R1	Aumentamos la altura del robot
R2	disminuimos la altura del robot
L1	-
L2	-
Joystick izquierdo eje Y	Giro del cuerpo del robot sobre el eje Y
Joystick izquierdo eje X	Giro del cuerpo del robot sobre el eje X
Joystick derecho eje Y	-
Joystick derecho eje X	Giro del cuerpo del robot sobre el eje Z

Tabla 26: Comandos movimiento del cuerpo del robot

4.7 Flujograma del código

En este subcapítulo se mostrarán los flujogramas de control para los procesos más importantes de la ejecución del código de Arduino.

4.7.1 Inicio del robot

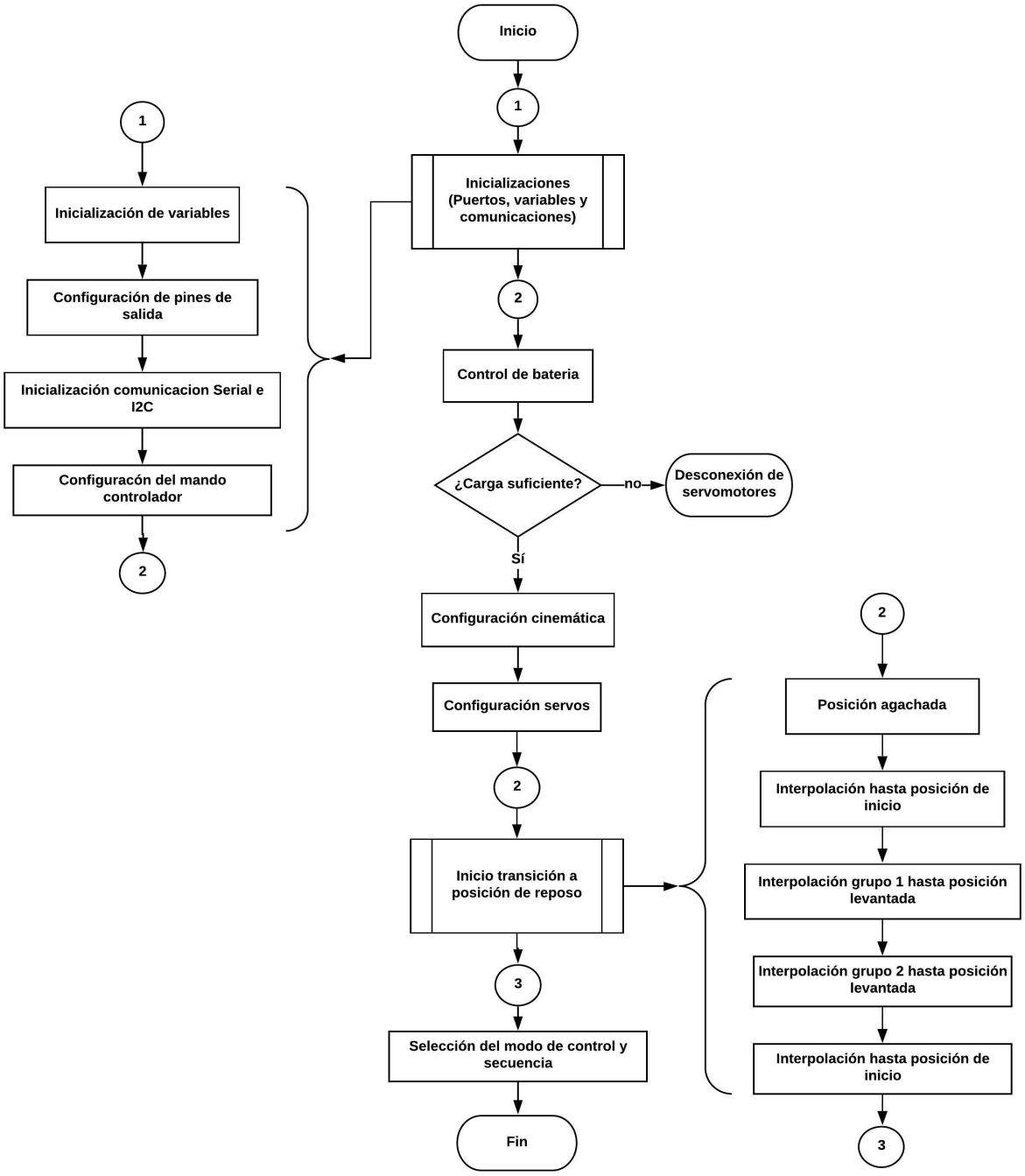


Figura 113: Flujograma del inicio del robot

4.7.2 Bucle Loop

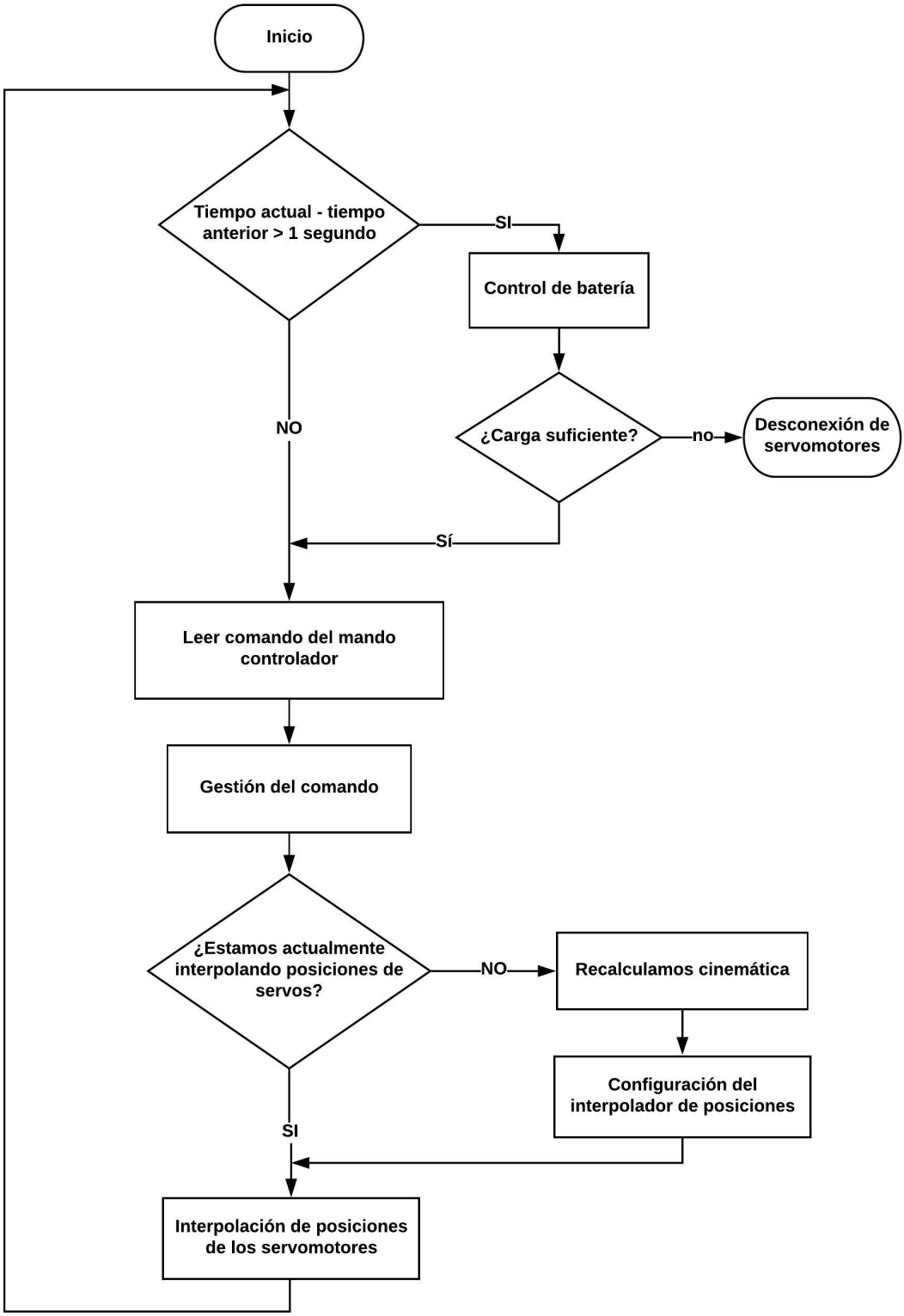


Figura 114: Flujograma del bucle loop

4.7.3 Cinemática

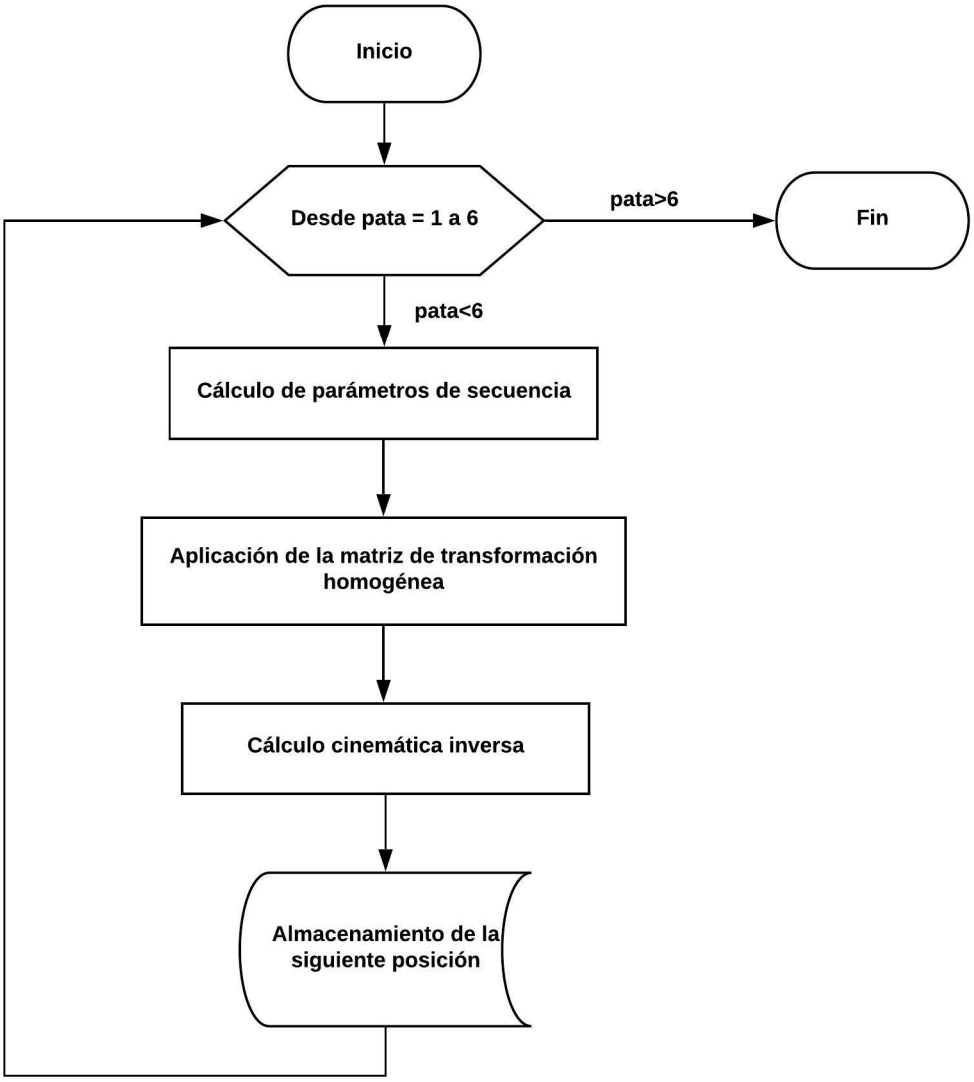


Figura 115: Flujograma del cálculo de la cinemática

5. RESULTADOS

En este capítulo se mostrarán los límites en el funcionamiento del robot. Esto se refiere a describir los rangos de movimiento permitidos por el mismo, antes de provocar un fallo de funcionamiento; por así decirlo una burbuja que delimita que si puede hacer el robot y qué no puede hacer.

- Altura máxima alcanzada hasta fallo

El fallo es provocado cuando se intenta ubicar a las patas en la posición la cual implica una altura mayor a la envergadura de estas, o que alguna de las articulaciones necesita orientarse a un cierto ángulo que está fuera de los límites establecidos para el servomotor en cuestión.

Para la altura máxima asignada, los servos no llegan a dar fallo, pero el robot comienza a moverse muy erráticamente a medida que se sobrepasan los 14 cm de altura. Esto evidencia que las patas están alcanzando su máxima envergadura y no consiguen alcanzar su posición correctamente. Al llevar al robot a estos límites, los errores que antes parecían no apreciarse, salen a la luz.

Figura 116: Altura máxima

- Altura máxima de paso hasta fallo

El robot permite una gran altura de paso sin que este tenga que llegar al fallo, por lo que limitaremos esta altura simplemente porque no tiene lógica elevar tanto las patas.

En las pruebas se observa que cuanto más incrementamos la altura de paso, el robot se vuelve más inestable. Esto es debido a que las patas tienen que recorrer más distancia en el mismo tiempo, incrementándose la inercia que adquieren estas y por lo tanto el balanceo del robot. También observamos que cuanto más dure el tiempo de ciclo, el robot es más estable a pesar de que incrementemos la altura de paso, esto se debe a que al aumentar el tiempo de ciclo, estamos disminuyendo la velocidad de desplazamiento del robot.

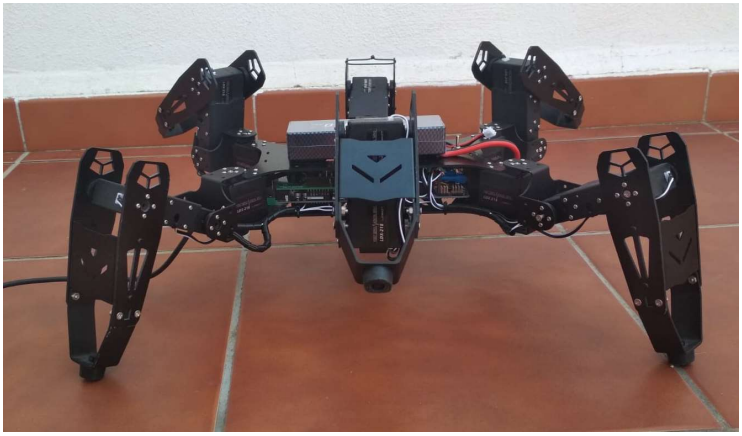


Figura 117: Altura máxima de paso

- Distancia máxima de zancada para evitar choque

La distancia máxima de zancada está limitada a 8cm como máximo en el eje X y 10 cm en el eje Y. Por encima de estos límites las patas podrían chocar entre sí, debido a alguna combinación aleatoria en los controles, producida por un movimiento muy brusco de ambos joysticks a la vez.

También se observa que aunque se seleccione una distancia de empuje de 8 y 10 cm para cada eje, físicamente se observa que la amplitud de movimiento es aproximadamente 1 cm o 1.5 cm inferior a la deseada.

- Los giros máximos del cuerpo

Respecto al giro del cuerpo (rotación sobre sí mismo), el valor máximo está establecido en 30º por zancada para que las patas no choquen unas con otras en su movimiento

Respecto del modo de movimiento del cuerpo, estas rotaciones están limitadas a 30º, ya que si aumentamos más de este valor algunos servomotores llegan a sus límites, provocando el fallo.

En las siguientes figuras podemos ver toda la cantidad de giros se le permite hacer al cuerpo del robot.

a) Giro horario

b) Giro antihorario

Figura 118: Giro sobre eje Z

a) Giro horario

b) Giro antihorario

Figura 119: Giro sobre eje Y

a) Giro horario

b) Giro antihorario

Figura 120: Giro sobre eje X

- Traslación del cuerpo del robot

Para la traslación del cuerpo en el plano xy, los valores máximos son 6cm de traslación para el eje X, en ambos sentidos y 8 cm en el eje Y en ambos sentidos. Si sobrepasamos estos valores las patas no conseguirán llegar hasta sus posiciones requeridas, provocando el fallo de los servomotores.

a) Traslación Y positiva

a) Traslación Y negativa

a) Traslación X negativa

b) Traslación X positiva

Figura 121: Traslaciones del cuerpo

6. PRESUPUESTO

En este capítulo se presenta el presupuesto del proyecto. En él se especifica el coste de fabricación de la PCB shield que se ha creado para Arduino, el hardware electrónico y el hardware mecánico, para finalmente mostrar el presupuesto total final.

Shield Arduino				
Descripción	Componente	Precio unitario	Ud	Total
Terminal de tornillo de 11 pines	J2	0,686€	1	0,686€
Conectores Arduino hembra (pack)	P1	1,9€	1	1,9€
	P2			
	P3			
	P4			
Resistencia 100k 5%	R5	0,05€	1	0,05€
Resistencia 100k 5%	R4	0,05€	1	0,05€
Resistencia 120k 5%	R6	0,05€	1	0,05€
Resistencia 150k 5%	R7	0,05€	1	0,05€
Resistencia 220 5%	R1	0,05€	1	0,05€
Resistencia 220 5%	R2	0,05€	1	0,05€
Resistencia 220 5%	R3	0,05€	1	0,05€
Resistencia 220 5%	R8	0,05€	1	0,05€
Transistor BJT 2N2222	Q1	0,105€	1	0,105€
P-Channel MOSFET,IRF5305	Q3	1,02€	1	1,02€
Capacitor cerámico 100nF	C1	0,94€	1	0,94€
Zener Diode 5V, BZX79-C5V1	D1	0,034€	1	0,034€
Buzzer	BZ1	0,8€	1	0,8€
Coste fabricación PCB	PCB	7,44	1	7,44€
Subtotal				13,325€

Tabla 27: Presupuesto PCB Shield Arduino

Componentes electrónicos				
Descripción	Componente	Precio unitario	Ud	Total
Fabricación PCB	Shield Arduino	13,325	1	13,325
Arduino Uno R3	Arduino	20,57	1	20,57€
Interruptor de palanca	interruptor	0,194	1	0,194€
Cables de conexionado	Cables	0,0583	10	0,583€
Controlador PWM	Controlador PWM	1,67	2	3,34€
Mando PS2 por cable	Mando PS2	15,9	1	15,9€
Batería HRB Lipo 7.4V 6000mAh, 2S 60C	Batería	35 €	1	35 €
Subtotal				88,912 €

Tabla 28: Presupuesto componentes electrónicos

Hardware mecánico				
Descripción	Componente	Precio unitario	Ud	Total
Robot hexápodo, 18 servos más piezas	Robot	217€	1	217€
Subtotal				217€

Tabla 29: Presupuesto hardware mecánico

Coste total del proyecto	
Descripción	Subtotal
Componentes electrónicos	88,912 €
Hardware mecánico	217€
Total	305,912€

Tabla 30: Presupuesto total

7. CONCLUSIONES Y LÍNEAS DE FUTURO

En este capítulo se expondrán las conclusiones finales sobre el proyecto y algunas ideas de cara a una posible mejora en el futuro.

- Montaje del robot

En el montaje del robot se han encontrado distintas dificultades. Uno de estos problemas se encuentra en la sujeción de las piezas y servomotores mediante sus respectivos tornillos; el tamaño de estos es muy pequeño por lo que no tienen mucho recorrido, lo cual hace que no se sujeten los elementos bien. No obstante, la cantidad de estos termina supliendo este defecto.

- Mandos controladores

Respecto al controlador se probaron distintas opciones. La primera fue la creación de una aplicación móvil con la ayuda de un programa desarrollado por el MIT, que facilita esta tarea para usuarios que no tengan los conocimientos de programación en Android. No obstante, esta idea fue descartada debido a que este programa era muy limitado y no podía crear las características necesarias para esta aplicación.

En segundo lugar un mando bluetooth PS3; con él se tuvieron dos problemas principalmente, el primero era referido al adaptador bluetooth usb necesario para la comunicación, ya que es difícil encontrar un adaptador compatible con estos mandos debido a que no hay un soporte oficial para este uso, y los creadores de la librería de Arduino no ofrecen ninguna garantía sobre esto. En segundo lugar, después de encontrar un receptor adecuado, nos encontramos con el problema de que estos solo funcionan con mandos originales, por lo menos para Arduino. Por lo que finalmente después de probar alrededor de entre 5 u 7 mandos sin un resultado satisfactorio, se descartó esta opción.

Como tercera opción que se barajó fue usar un mando PS2 inalámbrico, pero se acabó descartando debido a que el receptor se recibía comandos erróneos sin una causa coherente, por lo que podría ser que el mando estuviese defectuoso. Debido a esto se probó con un mando igual pero esta vez con cable, el cual funcionó correctamente. Por lo que se podría deducir que el anterior mando estaba defectuoso.

Finalmente se optó por usar este mando PS2 con cable, descartando la característica de controlar el robot inalámbricamente. A pesar de este cambio, no afecta a la motivación didáctica relativa a este proyecto, ya que el funcionamiento del robot no se ve restringido por este motivo.

Como propuesta de mejora sería muy interesante la creación de una aplicación móvil para realizar el control del robot de forma inalámbrica y sin depender de un hardware adicional que además, encarece el proyecto.

- Microcontrolador

Este proyecto consta de varios módulos electrónicos, los controladores PWM, Arduino y su shield. Sería muy útil fusionar todos los elementos en una misma placa PCB, para así ahorrar cableado y espacio. Además, ya que se diseñaría una nueva PCB, se podría buscar un microcontrolador con más puertos PWM para no depender de una controladora externa.

- Inclusión de un sistema de visión para navegación autónoma

A pesar de que el proyecto comentado en la introducción: “Sistema de navegación autónoma de robot hexápodo” del alumno José Antonio Guirado Cárdenas, que servía de complemento a este proyecto, no se ha conseguido implementar conjuntamente debido a que los desarrollo de ambos proyectos los proyectos no han coincidido en el mismo periodo de tiempo. Por este motivo, queda pendiente su implementación para ampliar este proyecto.

Así mismo también se podrían añadir distintos sensores al robot para que interactuase mejor con su entorno.

- **Conclusión final**

Finalmente el resultado global del robot ha sido mejor de lo esperado, tanto en el apartado de la construcción, como en el control del mismo. Por lo que a nivel personal se ha avanzado en gran medida en el área de la robótica y la programación.

Por otra parte es muy gratificante ver como de una vaga idea inicial, esta va tomando forma progresivamente y se va completando paso a paso hasta dar forma al robot final. También es impresionante ver como los cálculos realizados gracias a las matemáticas y disciplinas estudiadas, verdaderamente se comportan como se esperaba en el mundo real. Por lo que la satisfacción final es doblemente reconfortante.

8. BIBLIOGRAFÍA

Asif, U., iqbal, J., & khan, A. (2011). Kinematic Analysis of Periodic Continuous Gaits for a Bio-Mimetic. *researchgate*.

Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil Santoja, R. (2007). *Fundamentos de robótica.* McGraw-Hill.

Baturone, Á. O. (2005). *Robótica: Manipuladores y Robots Móviles.* Marcombo.

Fielding, M. (Junio 2002). *Omnidirectional Gait Generating Algorithm for Hexapod Robot.*

Fujii, S., Inoue, K., Takubo, T., & Arai, T. (2006). *Climbing up onto Steps for Limb Mechanism Robot "ASTERISK".* Osaka: Osaka University.

Khandaker, S. M. (2018). *An Intelligent Hexapod Rescue Robot.* Dhaka, Bangladesh: BRAC University.

llamas, L. (2019). *Web Electrónica*. Obtenido de https://www.luisllamas.es/.

López, L. K. (2015). *Robot caminante para aplicaciones de desminado humanitario.*

montes, H., Mena, L., Fernández, R., Sarria, J., González Santos, P., & armada, M. (s.f.). *Hexapod robot for humanitarian demining.* Panama.

Super robótica. (2019). Obtenido de http://www.superrobotica.com/servosrc.htm.

Torres, J. t. (2016). *Diseño y simulación del sistema de Locomoción de un Robot Hexápodo para tareas de búsqueda y rescate.* Madrid.

Velasco, J. G. (2009). *Energías renovables.* Editorial Reverte.

Verdugo, M. Á. (2011). *Diseño, construcción y control de un robot hexápodo.* México.

Web Arduino. (2019). Obtenido de https://www.arduino.cc/.

Web de robotica . (2019). Obtenido de https://www.robotshop.com/.

Web de robotica. (2018). Obtenido de https://www.trossenrobotics.com/.

Web software libre. (s.f.). Obtenido de https://www.eclipse.org/.

ANEXO 1: CINEMÁTICA

1. Cinemática en robótica

La cinemática del robot estudia el movimiento del mismo respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función de las relaciones entre la posición y orientación del extremo final del robot con los valores que toman sus coordenadas articulares. De esta manera la cinemática nos ayudará a establecer un método matemático para el control de la posición de los extremos del robot.

Existen dos problemas fundamentales a la hora de resolver la cinemática del robot; el primero se conoce como cinemática directa, y consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas y conociendo las coordenadas articulares; el segundo es la cinemática inversa que consiste en hallar las coordenadas articulares necesarias para adoptar una posición y orientación conocidas de antemano.

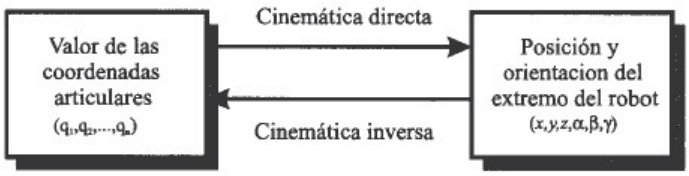


Figura 122: Relación entre cinemática directa e inversa

1.1 Matemáticas para la localización espacial

Para el movimiento espacial del extremo de un robot, es necesario conocer la posición y orientación de esta con respecto a la base del robot. Se aprecia entonces la necesidad de contar con una serie de herramientas matemáticas que permitan especificar su posición y orientación.

1.1.1 Representación de la posición

Aunque existen otros sistemas de representación como las coordenadas cilíndricas o las esféricas, se utilizará el sistema cartesiano por ser la forma más intuitiva y utilizada.

1.1.1.1 Sistema cartesiano de referencia

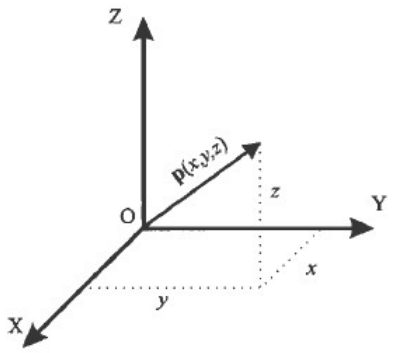


Figura 123: Vector en coordenadas cartesianas de 3 dimensiones

Los ejes cartesianos se componen por una terna ortonormal de vectores coordenados OX, OY, OZ.

$$\vec{p}=\begin{bmatrix}\overrightarrow{p_x}\\ \overrightarrow{p_y}\\ \overrightarrow{p_z}\end{bmatrix}$$

1.1.2 Representación de la orientación

Un punto queda totalmente definido en el espacio a través de su posición, pero en el caso de un sólido se necesita además su orientación con respecto a un sistema de referencia.

1.1.2.1 Matrices de rotación

Las matrices de rotación son el método más fácil y extendido de describir orientaciones, debido a las facilidades que proporciona el uso de álgebra matricial.

En un espacio tridimensional supóngase los sistemas OXYZ y OUVW, coincidentes en el origen, siendo el OXYZ el sistema de referencia fijo, y el OUVW solidario al objeto cuya orientación se desea definir. Los vectores unitarios del sistema OXYZ serán i_x, j_y, k_z mientras que los del OUVW serán i_u, j_v, k_w . Un vector *p* del espacio podrá ser referido a cualquiera de los sistemas como:

$$P_{uvw} = [p_u, p_v, p_w]^T = p_u * i_u + p_v * j_v + p_w * k_w$$

$$P_{xyz} = [p_x, p_y, p_z]^T = p_x * i_x + p_y * j_y + p_z * k_z$$

Realizando una serie de transformaciones se llega a la equivalencia:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

Donde:

$$\boldsymbol{R} = \begin{bmatrix} i_x i_u & i_x i_v & i_x k_w \\ j_u i_u & j_y j_v & j_y k_w \\ k_z i_u & k_z j_v & k_z k_w \end{bmatrix}$$

Es la matriz de rotación que define la orientación respecto del sistema OUVW con respecto al sistema OXYZ.

- Giro respecto al eje OX
-

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ 0 & \operatorname{sen}\alpha & \cos\alpha \end{bmatrix}$$

- Giro respecto al eje OY
-

$$\boldsymbol{R} = \begin{bmatrix} \cos\phi & 0 & \operatorname{sen}\phi \\ 0 & 1 & 0 \\ -\operatorname{sen}\phi & 0 & \cos\phi \end{bmatrix}$$

- Giro respecto al eje OZ
-

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.1.2.2 Composición de rotaciones

Las matrices de rotación pueden componerse para expresar la aplicación continua de varias rotaciones. Pero hay que tener en cuenta el orden en que se realizan las rotaciones, pues el producto de matrices no es conmutativo.

$$T = R(z,\theta)R(y,\phi)R(x,\alpha) = \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C\phi & 0 & S\phi \\ 0 & 1 & 0 \\ -S\phi & 0 & C\phi \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} =$$

$$= \begin{bmatrix} C\theta C\phi & -S\theta C\alpha + C\theta S\phi S\alpha & S\theta S\alpha + C\theta S\phi C\alpha \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha \\ -S\phi & C\phi S\alpha & C\phi C\alpha \end{bmatrix}$$

Donde $C\theta$ se expresa $cos\theta$ y $S\theta$ expresa $sen\theta$.

En aeronáutica existe una nomenclatura para los giros: Roll, Pitch and Yaw (alabeo, cabeceo y guiñada)

1. Girar un ángulo ψ con respecto al eje OX. Es denominado Yaw o guiñada.
2. Girar un ángulo θ con respecto al eje OY. Es denominado pitch o cabeceo.
3. Girar un ángulo φ con respecto al eje OZ. Es denominado Roll o alabeo.

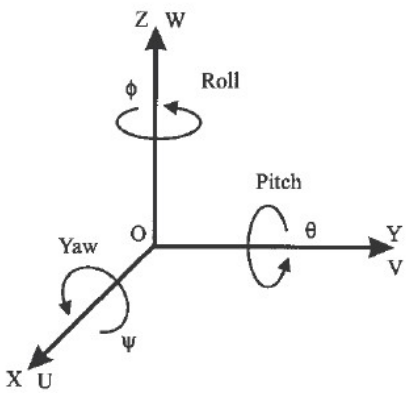


Figura 124: Ángulos de Euler: roll, pitch y yaw

1.1.3 Matrices de transformación homogénea

Gracias a las denominadas coordenadas homogéneas se puede representar conjuntamente posición y orientación de un sistema, que ha sido girado y trasladado con respecto a un sistema fijo de referencia.

1.1.3.1 Coordenadas y matrices homogéneas

La representación mediante coordenadas homogéneas de la localización de sólidos en un espacio n-dimensional se realiza a través de coordenadas de un espacio (n+1) dimensional, de tal forma que un vector $p\,(x,y,z)$ vendrá representado por $p(wx,wy,wz,w)$, donde w tiene un valor determinado y representa un factor de escala.

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} aw \\ bw \\ cw \\ w \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}$$

A partir del concepto de coordenadas homogéneas se deduce el concepto de matriz de transformación homogénea. Se define como matriz de transformación homogénea T a una matriz de dimensión 4 x 4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ Perspectiva & Escalado \end{bmatrix}$$

Donde R_{3x3} corresponde a la matriz de rotación; una submatriz p_{3x1} que corresponde al vector de translación; otra submatriz f_{3x1} que representa una transformación de persectiva que en robótica suele ser nula, y una submatriz w_{1x1} que representa el escalado global y suele ser 1.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ 0 & 1 \end{bmatrix}$$

1.1.3.2 Traslación y rotación

- Matriz homogénea de traslación

$$\boldsymbol{T}(\boldsymbol{p}) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vector cualquiera $\vec{r}$ trasladado un vector $\vec{p}$:

$$\begin{bmatrix} \boldsymbol{r_x} \\ \boldsymbol{r_y} \\ \boldsymbol{r_z} \\ \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_u + p_x \\ r_v + p_y \\ r_w + p_z \\ 1 \end{bmatrix}$$

- Matriz homogénea de rotación sobre el eje OX:

$$\boldsymbol{T}(\boldsymbol{x}, \boldsymbol{\alpha}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha & 0 \\ 0 & \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz homogénea de rotación sobre el eje OY:

$$\boldsymbol{T}(\boldsymbol{y}, \phi) = \begin{bmatrix} \cos\phi & 0 & \operatorname{sen}\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz homogénea de rotación sobre el eje OZ:

$$\boldsymbol{T}(\boldsymbol{z}, \theta) = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.1.3.3 Rotación seguida de translación

En el caso de realizar primero una rotación sobre uno de los ejes coordenados del sistema OXYZ seguida de una translación, las matrices homogéneas serán:

- Rotación de un ángulo α sobre el eje OX seguido de una translación.

$$\boldsymbol{T}((\boldsymbol{x},\boldsymbol{\alpha}),\boldsymbol{p}) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & \cos\alpha & -\operatorname{sen}\alpha & p_y \\ 0 & \operatorname{sen}\alpha & \cos\alpha & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación de un ángulo ϕ sobre el eje OY seguido de una translación.

$$\boldsymbol{T}((\boldsymbol{y},\phi),\boldsymbol{p}) = \begin{bmatrix} \cos\phi & 0 & \operatorname{sen}\phi & p_x \\ 0 & 1 & 0 & p_y \\ -\operatorname{sen}\phi & 0 & \cos\phi & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación de un ángulo θ sobre el eje OZ seguido de una translación.

$$\boldsymbol{T}((\boldsymbol{z},\theta),\boldsymbol{p}) = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 & p_x \\ \operatorname{sen}\theta & \cos\theta & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.1.3.4 Composición de matrices homogéneas

La composición de matrices homogéneas permite describir diversos giros y traslaciones consecutivos sobre un mismo sistema de referencia. De esta forma una transformación compleja puede descomponerse en la aplicación consecutiva de transformaciones simples.

Debido a que el producto de matrices no es conmutativo, tampoco lo es la composición de transformaciones. La composición del giro de un vector sobre los 3 ejes coordenados y una traslación quedaría como se muestra a continuación.

$$T = T(x,\alpha)T(y,\phi)T(z,\theta) =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\phi & 0 & S\phi & 0 \\ 0 & 1 & 0 & 0 \\ -S\phi & 0 & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\phi C\theta & -C\phi S\theta & S\phi & 0 \\ S\alpha S\phi C\theta + C\alpha S\theta & -S\alpha S\phi S\theta + C\alpha C\theta & -S\alpha C\phi & 0 \\ C\theta C\alpha S\phi + S\alpha S\theta & C\alpha S\phi S\theta + S\alpha C\theta & C\alpha C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.2 Cinemática inversa

El objetivo de la cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $\theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$ para que su extremo se posicione y oriente según una determinada posición espacial.

Sin embargo, al contrario que la cinemática directa, que se puede abordar de manera sistemática mediante el uso de matrices de transformaciones homogéneas e independientes de la configuración del robot, en la cinemática inversa el procedimiento de obtención de las ecuaciones es muy dependiente de la configuración del robot.

Se han creado algunos métodos genéricos que pueden ser programados, de modo que un ordenador pueda, a partir del conocimiento de la cinemática del robot (parámetros Denavit-Hartenberg), los valores articulares que posicionan y orientan el extremo. El problema de estos métodos es que se basan en procesos iterativos, cuya velocidad y convergencia no está siempre garantizada.

A la hora de resolver el problema cinemático inverso es mucho más adecuado encontrar una solución cerrada. Esto es, encontrar una relación matemática explicita de la forma:

$$\theta_k = f_k(x, y, z, \alpha, \beta, \gamma,)$$

$$k = [1, \dots, n]\ (grados\ de\ libertad)$$

Este tipo de solución presenta las siguientes ventajas:

- En la mayoría de aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real, por lo que una solución de tipo iterativo no garantiza tener la solución en el tiempo requerido.

- Al contrario de lo que ocurría en la cinemática directa, hay veces que la solución de la cinemática inversa del robot no es única. Existen diferentes orientaciones de las articulaciones para una misma posición y orientación del extremo del robot. En estos casos de solución cerrada se pueden incluir determinadas reglas o restricciones para asegurarse que la solución es la buscada.

No obstante, a pesar de todos los inconvenientes comentados, la mayor parte de los robots poseen cinemáticas relativamente simples que facilitan su resolución.

1.2.1 Distintos métodos de resolución

1.2.1.1 Método geométrico

Este método es adecuado para robots que poseen pocos grados de libertad. El procedimiento se basa en encontrar las suficientes relaciones geométricas en las que estarán involucradas las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.

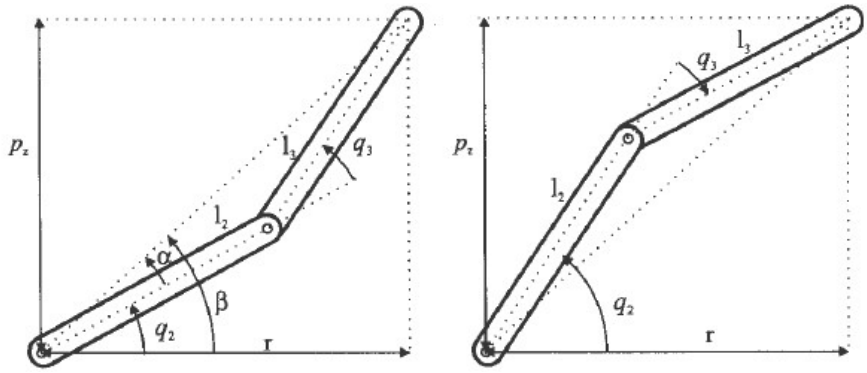


Figura 125: Distintas soluciones geométricas para una misma orientación

1.2.1.2 Resolución a partir de las matrices de transformación homogénea

Trata de hallar el modelo cinemático inverso de un robot a partir del conocimiento de su modelo directo. Es decir, conocidas las relaciones que expresan el valor de la posición y orientación del extremo del robot en función de sus coordenadas articulares, obtener por manipulación de aquellas las relaciones inversas. En la práctica esta tarea no es trivial, resultando en muchas ocasiones tan compleja que obliga a desecharla.

1.2.1.3 Desacoplamiento cinemático

Este método permite, para determinados tipos de robots, resolver de manera separada la posición y la orientación. Los grados de libertad, dedicados al posicionamiento, se resuelven de una manera independiente a la de grados de libertad, dedicados a la orientación. Este método es típico en robots de 6 grados de libertad, donde los tres primeros están destinados a posicionar el brazo y los tres últimos, a orientar el extremo del robot.

2. Trayectorias

Para que el robot se oriente de manera determinada este tiene que moverse desde un punto inicial a un punto final. Este movimiento puede ser realizado según infinitas trayectorias espaciales. De las cuales solo algunas por su sencillez de implementación o bien por su adaptación a diversas tareas son las que verdaderamente se implementan en robots comerciales. De este modo se pueden encontrar trayectorias de tipo punto a punto, coordinadas y continúas.

2.1 Trayectorias punto a punto

En este tipo de trayectoria cada articulación transiciona desde su posición inicial a la final sin tener en cuenta el estado o la evolución de las demás en este proceso. De este modo cada actuador lleva a su articulación hasta su punto de destino en el menor tiempo posible, pudiéndose distinguir dos variantes: movimiento simultáneo de ejes o movimiento eje a eje.

2.1.1 Movimiento eje a eje

En este tipo de movimiento solo cambia de posición una articulación a la vez, y una vez que se alcance su punto final, empezará su movimiento la segunda, y así sucesivamente. Este tipo de movimiento da lugar a un mayor tiempo de ciclo, pero como punto a favor, se obtiene un menor consumo de potencia instantánea por parte de los actuadores.

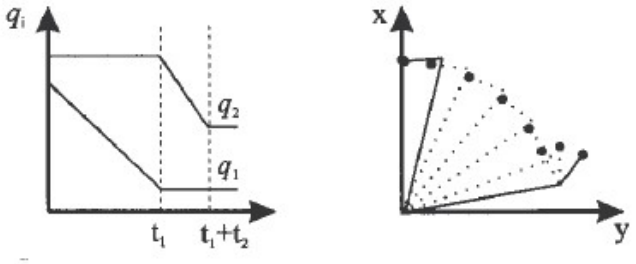


Figura 126: Movimiento eje a eje

2.1.2 Movimiento simultáneo de ejes

En este caso todas las articulaciones se mueven a la vez a una determinada velocidad específica para cada una de ellas. Debido a que la distancia a recorrer y las velocidades serán en general diferentes, cada articulación terminará su movimiento en un tiempo diferente.

El movimiento del robot no acabará hasta que se alcance el punto final, esto se producirá cuando la articulación que más tarde concluya su movimiento. De esta forma, el tiempo total invertido coincidirá con la articulación que más tarde, viendo como las demás han sido forzaras a ir más rápido de lo requerido, innecesariamente.

Por estos motivos las trayectorias punto a punto solo están implementadas en robots simples o con unidades de control muy limitadas

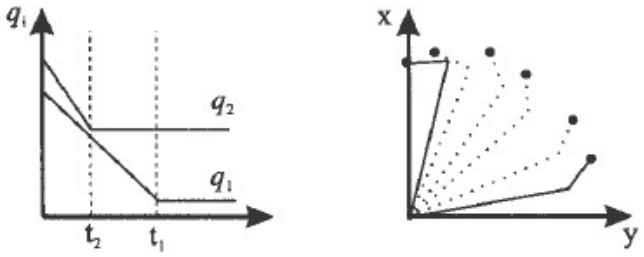


Figura 127: Movimiento simultaneo de ejes

2.2 Trayectorias coordinadas o isocronas

Para evitar que algunos actuadores trabajen forzando sus velocidades y aceleraciones, teniendo además que esperar la finalización del movimiento de la articulación más lenta, puede hacerse un cálculo previo para averiguar cuál será esta articulación y cuánto tiempo tardará. Se procederá entonces al movimiento de los ejes simultáneamente, acabando todos ellos al mismo tiempo. De esta forman, el tiempo total invertido es el mínimo posible y no se requieren velocidades ni aceleraciones innecesarias para los actuadores.

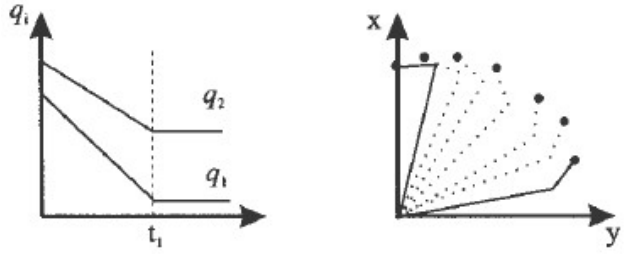


Figura 128: Trayectoria coordinada

2.3 Trayectorias continuas

Cuando se pretende que la trayectoria que sigue el extremo del robot sea conocida por el usuario, es necesario calcular de manera continua las trayectorias articulares.

Normalmente las trayectorias que el usuario implementa son en línea recta o en forma de arco. Para conseguir esto habrá que implementar unas determinadas funciones de control cinemático. El resultado será que cada articulación seguirá un movimiento aparentemente caótico, con incluso cambios de dirección y velocidad por parte de alguna articulación. Sin embargo, el resultado general en el extremo del robot es el seguimiento exacto de la trayectoria.

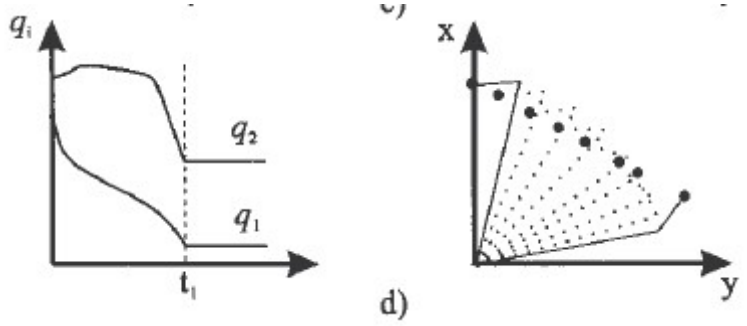


Figura 129: Trayectoria continua rectilínea

ANEXO 2: SERVOMOTORES

Estos se componen por un motor eléctrico de corriente continua, un sistema de regulación que actúa sobre el motor y un sistema de sensor que controla la posición del motor. Este sensor es el que marca la diferencia con respecto a un motor controlado electrónicamente, ya que gracias a la información obtenida del sensor, se puede saber en tiempo real en qué posición se encuentra el servomotor.

2 Partes de un servomotor

El motor eléctrico, juego de engranes y tarjeta de control se encuentran todos confinados en una carcasa de plástico.

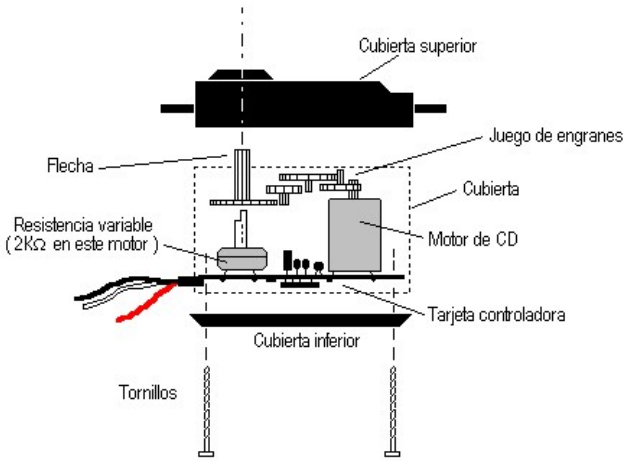


Figura 130: Componentes de un servomotor

El motor en sí puede ser AC o DC, y si es DC, puede ser cepillado o sin escobillas. Lo que distingue a un servo de otros tipos de motor es que no está diseñado para una rotación continua. Es un dispositivo de búsqueda de posición.

Los más utilizados para un rango de precio reducido son los motores DC, ya que son pequeños y fáciles de controlar. No obstante, el motor mencionado ofrece mejores características de velocidad que de torque, por lo que se utiliza un juego de engranajes para obtener un mayor par en el eje.

Estos engranajes de reducción pueden ser de nylon o de metal, siendo los primeros más baratos pero con la contrapartida de ser más débiles frente a grandes cargas de trabajo, con su consecuente desgaste.

La electrónica que se encuentra dentro de la carcasa del servomotor interpreta los comandos de un controlador externo. La señal de control especifica el ángulo de giro deseado medido como un desplazamiento a ambos lados de la posición central del rango del motor. Como respuesta a una señal de control, el motor girará rápidamente a la posición especificada y se detendrá allí. Mientras la señal de comando continúe y la potencia del motor se mantenga, el motor mantendrá su posición y "empujará" contra cualquier fuerza de giro externa. En ausencia de tal fuerza, mientras el motor está parado, utilizará muy poca corriente.

Para saber si el servomotor ha alcanzado la posición deseada, se incorpora un sensor el cual es un potenciómetro conectado al eje y que variará su resistencia dependiendo de la orientación que se alcance. Esta información se retroalimenta hacia el sistema electrónico de control, para ajustar el error que pueda haber en la posición.

2.1 Tipos de servomotores

Existen muchos tipos y tamaños de servomotores, como los servos hidráulicos, los basados en motores de corriente alterna empleados en sistemas de gran potencia y los basados en motores de corriente continua que son los más utilizados en máquinas herramientas, robots industriales, sistemas de producción, etc. El coste de un servo industrial comienza en los 500 euros y pueden llegar fácilmente a varios miles de euros por cada servo.

Los servomotores empleados en estos tipos de proyectos de robótica son un tipo de servo motor muy específico que se basan en los mismos principios que los servos industriales pero con significativas diferencias en cuanto al control y al movimiento que producen y, sobre todo, al coste de los mismos. Por lo que en nuestro caso, todo lo relacionado con servomotores, se referirá a este tipo de servos, los cuáles son mucho más económicos.

El rango de movimiento típico para esta categoría de servos se encuentra de 0º a 180º, aunque existen otras opciones más caras que consiguen un rango de movimiento de hasta 270º las cuáles aportan mayor capacidad de movimiento a las articulaciones de los robots.

2.1.1 Dimensiones y potencia

Hasta hace no mucho tiempo los servos se clasificaban básicamente por su tamaño. Todos los servos funcionaban de la misma forma y en lo único que se diferenciaban era en el tamaño de los mismos. Dentro de esta clasificación están los sevos estándar que incluye a la mayoría de los servos y que se trata de servos cuyas medidas son 40 x 20 x 37 mm. Las pueden variar un poco de unos modelos a otros, pero por lo general las medidas de los orificios de fijación y la posición del eje de salida coinciden en casi todos los modelos, lo que ha permitido crear soportes y fijaciones universales que pueden utilizarse con la mayoría de los servos.

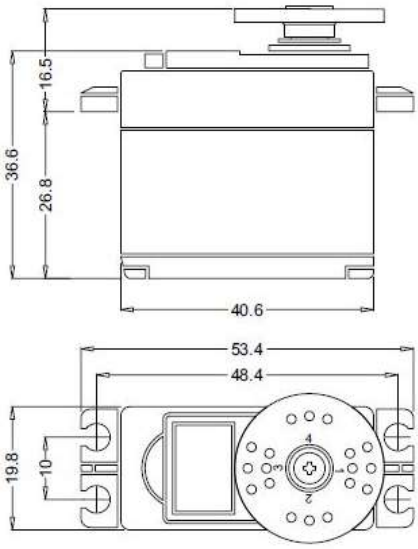


Figura 131: Servo estándar

Figura 132: Fijaciones universales

La potencia de estos servos estándar puede variar entre los 3 Kg. y los 24 Kg.

Existen servos de mayor tamaño, utilizados en coches a radiocontrol de escala 1:4 que utilizan servos a escala 1:4, los cuales son más robustos y poseen mayor potencia. Estos modelos se emplean en casos donde el peso no es tan problemático y se encesita mucha potencia, como en el caso de coches Rc 4x4.

En principio los servos de menor tamaño se empleaban sobre todo en los aviones, que se ubican en los flaps de las alas o en el tren de aterrizaje, en donde el peso y el tamaño son muy críticos.

Hoy en día se encuentran sustitutos de estos servos con igual o superior potencia en tamaños estándar, gracias al control digital. La potencia de estos servos varía entre 9 kg.cm y 24 kg.cm

Otra aplicación es la construcción de robots miniaturas en la que los servos estándar son demasiado grandes. Dentro de los servos miniaturas se encuentran hoy en día muchos tamaños desde los mini, los ultra mini y los microservos con apenas 8 gramos de peso. Su rango de potencia está entre los 0,5 kg.cm y los 3,5 kg.cm.

2.1.2 Servos analógicos y digitales

Hasta hace no mucho tiempo, todos los servomotores eran analógicos, es decir, su control se basaba en un circuito electrónico basado en comparadores y amplificadores operacionales que siempre funcionaban de la misma forma. Con la aparición de los servos digitales se han conseguido avances tanto en prestaciones como en opciones de control.

La mejora de rendimiento se produce, en mayor medida, gracias a que la electrónica de control la realiza un microcontrolador. Esto permite mandar con mayor frecuencia pulsos al motor, aumentando la precisión de su movimiento y su rendimiento. También se realizan más lecturas por segundo del potenciómetro, obteniendo una mayor rapidez en el feedback.

A parte de esto, también se utilizan nuevos drivers para el motor más eficaces, con mayor capacidad de integración, menor consumo y eficacia para entregar más potencia al motor en un corto periodo de tiempo.

Además el microcontrolador tiene otras ventajas, como la capacidad de programar algunos parámetros como la posición central, el margen de recorrido, la zona neutra etc… No obstante, cabe destacar que el torque máximo viene determinado por el motor y el equipamiento de engranajes reductores; la ganancia en este aspecto recae en la rapidez con la que se entrega el par máximo y la aceleración que se logra para alcanzar la velocidad esperada de giro.

2.2 Control de servomotor

2.2.1 Modulación por ancho de pulso (PWM)

Los servomotores son generalmente controlados por modulación de ancho de pulso (PWM), esto quiere decir que desde un controlador, como puede ser un microcontrolador, se le envía una señal de control la cual es interpretada por la electrónica del servomotor para orientarlo según se busque. Para el tipo de servos que utilizaremos los pulsos se realizan a una frecuencia constante de 50 hercios o cada periodo de 20 ms, de manera que si el ancho de pulso se mantiene en 1,5 ms el servo se localizará posición centrada. Si el pulso fuese más corto, por ejemplo 1 ms el servo girará hacia la izquierda, si el pulso es mayor, por ejemplo 2 ms, el servo gira a la derecha.

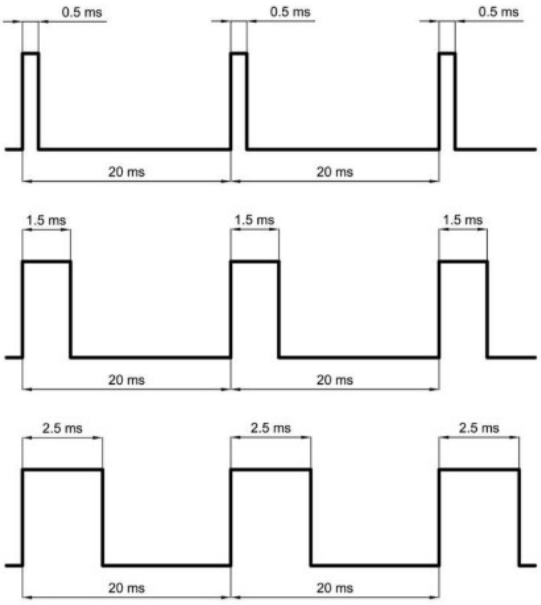


Figura 133: Ancho de pulso

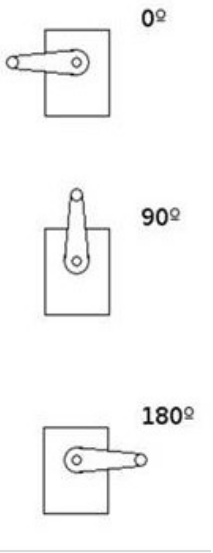


Figura 134: Orientación

El movimiento del servomotor depende del ancho de pulso que se le aplica, por lo que para el control del mismo habrá que hallar la constante de proporcionalidad que nos permite cambiar de ancho de pulso a grados.

Los servos modernos dejan de controlar el motor, tan pronto como se dejan de mandar los pulsos de control. Por eso, para controlar un servo hay que mandar los pulsos de control unas 50 veces por segundo, con un margen del 20 % aproximadamente. Si se pasa más tiempo sin mandar pulsos, el servo entra en reposo y como consecuencia su consumo baja a apenas 8 mA dependiendo del tipo y marca de servo, lo cual puede ser interesante para algunas aplicaciones.

2.2.2 Flujograma de control

En primer lugar, como se ve en la figura inferior, la señal de control se transforma en un voltaje proporcional a la posición que se quiere obtener, teniendo como referencia la tensión de alimentación del servomotor

En segundo lugar, tenemos una retroalimentación de la posición actual del servo, gracias a un potenciómetro el cual está conectado al eje del servo y a un sistema de engranajes; de este modo, se nos proporciona un voltaje proporcional a la orientación actual.

Estas dos señales se mandan a un amplificador operacional, que está configurado en modo comparador; la diferencia resultante será el error que hay entre la orientación que se busca y en la que se encuentra actualmente.

Esta señal de error se introduce en un microcontrolador y genera una respuesta o salida en base a un programa realizado previamente, que controla la dinámica de movimiento del motor.

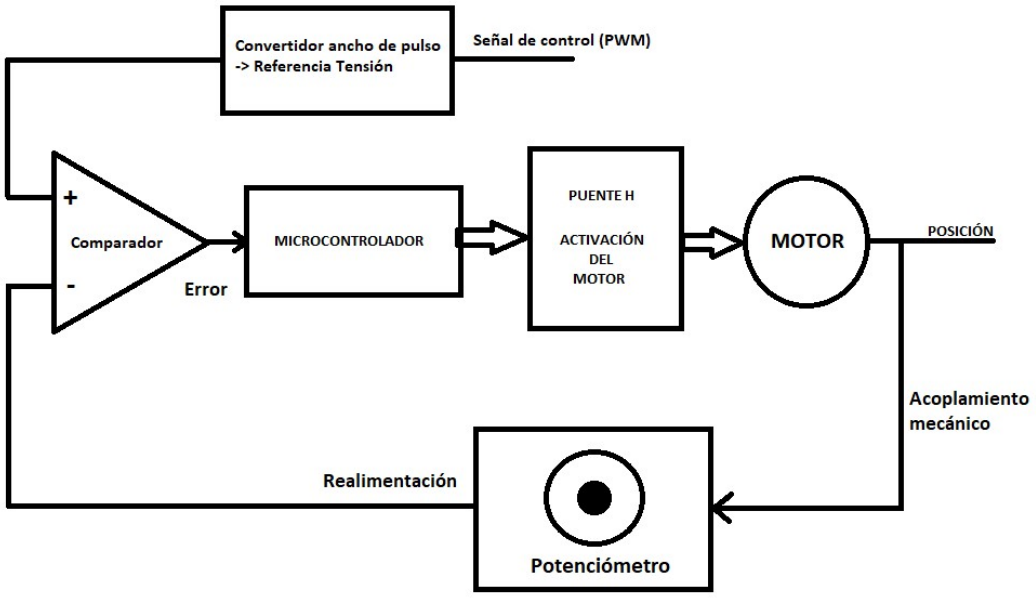


Figura 135: Diagrama de control de un servomotor

Estas salidas controlan el funcionamiento del puente H, que es un circuito que controla el sentido de giro del motor. Ubicándolo de nuevo con la orientación deseada.

2.2.3 Puente H

Se le da el sobrenombre de puente H debido a la forma que presenta dentro de un circuito esquemático simplificado. En la rama central se encuentra ubicado el motor y en cada rama lateral ascendente o descendente se ubican los conmutadores que, activados de manera apropiada, brindarán al sistema los movimientos necesarios para que el motor utilizado pueda girar en un sentido u otro.

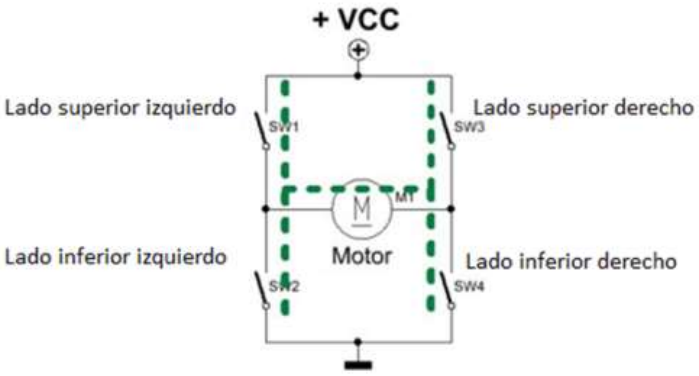


Figura 136: Circuito puente H

Si por ejemplo, se cierran los contactos SW3 y SW2 tendremos un sentido en el motor, mientras que si activamos SW4 y SW1, tendremos un sentido contrario al anterior. En caso de cerrar SW2 y SW4 ó SW1 y SW3, tendremos una condición de paro rápido, debido a que la fuerza electromotriz generada en las bobinas debido a la inercia del movimiento del rotor queda anulada por el cortocircuito en dichas bobinas. Esto provoca un desaceleramiento en el motor.

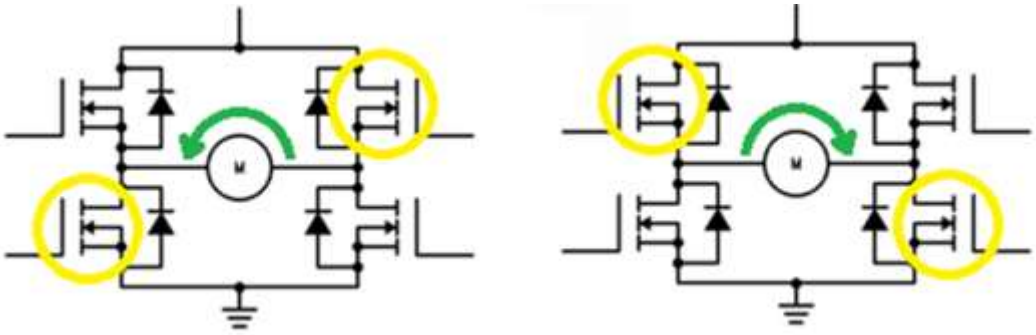


Figura 137: Giro del motor al activar Mosfets alternos

En realidad estos conmutadores son transistores tipo Mosfet, los cuales son controlados por voltaje. El microcontrolador manda una señal a la puerta del Mosfet para que éste deje pasar la corriente o no, controlando de este modo la orientación del eje del motor.

El diodo que vemos en paralelo al transistor se denomina **diodo Flyback** y su función es la protección del Mosfet frente a voltajes extremadamente altos, que romperían su estructura interna dejándolos inservibles. Este diodo proporciona un recorrido de mínima resistencia, que permite disipar las corrientes inducidas producidas por el campo magnético de la carga inductiva (Motor) cuando ésta es desconectada y que dañarían al transistor.

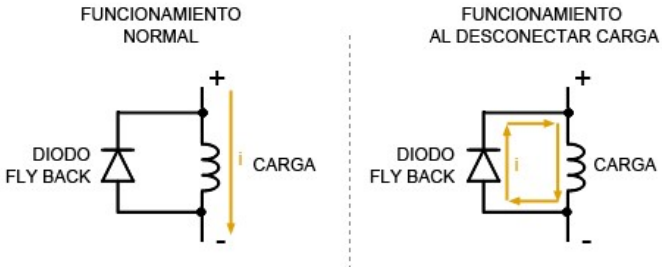


Figura 138: Funcionamiento diodo flyback

ANEXO 3: ESQUEMÁ ELECTRÓNICO SHIELD PCB