UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

SIMULACIÓN Y CARACTERIZACIÓN DEL COMPORTAMIENTO DE ROBOTS VIBRACIONALES.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

PABLO ANDRÉS SILVA GUTIÉRREZ

PROFESOR GUÍA:

JUAN CRISTÓBAL ZAGAL MONTEALEGRE

MIEMBROS DE LA COMISIÓN:

CLAUDIO FALCÓN BEAS

BRUNO GROSSI CÓRDOVA

SANTIAGO DE CHILE

2019

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE: Ingeniero Civil Mecánico
POR: Pablo Andrés Silva Gutiérrez
FECHA: agosto de 2019
PROFESOR GUÍA: Juan Zagal Montealegre

SIMULACIÓN Y CARACTERIZACIÓN DEL COMPORTAMIENTO DE ROBOTS VIBRACIONALES.

Los enjambres de robots modulares se inspiran en la naturaleza, estos sistemas son capaces de lograr tareas complicadas como un conjunto, mientras que cada individuo realiza tareas sencillas. Algunos ejemplos son: la migración de aves, movimiento de cardúmenes de peces, colonias de abejas y hormigas y en una escala más fundamental el comportamiento de las células en organismos multicelulares.

El objetivo del presente estudio, es estudiar el comportamiento de un grupo de robots modulares que se desplazan mediante motores vibracionales. Cada robot puede moverse en línea recta y girar. Se estudia la capacidad del enjambre de formar figuras o patrones ingresadas por el usuario, para lograr este comportamiento, los robots provistos de sensores de luz, reciben comandos motrices mediante señales de color, dicha luz de color es enviada por un controlador central externo, este controlador se encarga de computar la acción a realizar por el robot en tiempo real. La implementación del control es hibrida, porque existe un solo controlador externo, no obstante el control es distribuido ya que depende de lazos de control local asociado a cada robot.

Se utiliza la cámara OptiTrack junto al software MovieTracker para determinar posición y velocidad de los robots, estos elementos permiten crear un lazo de control cerrado, que se utiliza para corregir los errores en el movimiento de los robots. Una parte central de este estudio es el diseño del sistema de control y la calibración del proyector de manera de apuntar la luz correctamente a las fotorresistencias.

Las líneas de estudio a desarrollar en esta memoria son dos: la primera línea corresponde a modelar y simular el comportamiento de los robots mediante el software Python, la segunda línea corresponde al estudio y caracterización experimental del comportamiento de los robots. Se estudian 3 casos: movimiento colectivo, Self Assembly formación de figuras tipo racimo y Self Assembly formación de figuras ingresadas por el usuario.

Finalmente se analiza cómo afecta la cantidad de elementos en el sistema, al comportamiento que exhibe el sistema (tiempo de Frame, tiempo total) y cómo afecta la calidad de las figuras formadas.

Tabla de contenido

1 Introducción

En la búsqueda del aumento de complejidad en las tareas de los robots, además del enfoque tradicional, donde una sola unidad realizaba tareas cada vez más complejas. Surge un nuevo enfoque, en donde los robots como unidad realizan tareas simples, mientras que como un conjunto realizan tareas complejas, este enfoque es la robótica de enjambre.

En la presente memoria se estudia el comportamiento de un enjambre robótico, el cual debe generar de manera autónoma figuras y/o patrones ingresados por el usuario. Para esto se provee a los robots con sensores y transmisores de manera que estos puedan comunicarse entre sí. La capacidad de un colectivo de formar figuras o patrones se conoce como self-assembly.

Para abordar los objetivos del presente estudio, se divide las líneas de trabajo en dos áreas principales: la primera área, corresponde a la parte del diseño, modelamiento y simulación y la segunda área es el desarrollo experimental. Dentro de la primera área se encuentra la mejora y/o acondicionamiento del prototipo de robot vibracional proveniente de trabajos previos en el laboratorio de síntesis de máquinas inteligentes, el siguiente paso es el diseño e implementación del sistema de control, se estudia la implementación de un controlador PID, para realizar el modelo y posterior simulación se hace uso de modelos de movimiento colectivos como el de Cucker-Smale. La segunda etapa comprende el estudio experimental del comportamiento de los robots, los resultados de estos experimentos serán comparados luego con el resultado de las simulaciones, para probar la precisión del modelo planteado.

1.1 Antecedentes Generales y motivación.

La robótica de enjambre, se inspira en la naturaleza, desde lo más básico como es la reparación de algún tejido por parte de las células, hasta interacciones de animales como es el caso de los cardúmenes de peces o el funcionamiento de las colonias de hormigas.

Un enjambre robótico se puede definir como un grupo de robots que trabaja de forma coordinada y descentralizada, los robots son simples y están sujetos a reglas de localidad. Se puede diferenciar la robótica de enjambre con respecto a la robótica multi-agente, en las reglas expuestas anteriormente, por ejemplo el futbol de robots aunque varios agentes trabajan de manera coordinada, estos no obedecen el principio de localidad, porque deben ver la pelota a cualquier distancia para poder reaccionar a tiempo y cada robot realiza tareas especializadas, en contraste con un enjambre en donde todos los individuos realizan tareas iguales.

1.2 Objetivos

1.2.1 Objetivo General

El objetivo del presente trabajo es implementar y controlar un sistema robótico compuesto por 50 módulos, cada módulo posee dos grados de libertad y es controlado mediante luz de color. Los individuos del sistema robótico, deben interactuar entre sí, de tal forma de producir figuras o patrones ingresados por el usuario.

1.2.2 Objetivos Específicos

- Construir 50 unidades de robots vibracionales.
- Definir las reglas de movimiento del sistema
- Diseñar e implementar la lógica de control
- Construir el setup experimental.
- Evaluar el desempeño del sistema con 10 unidades.
- Evaluar el desempeño del sistema con 50 unidades.
- Evaluar mediante una simulación el desempeño de 100 unidades.
-

1.2.3 Alcances

- Diseño de la filosofía de control.
- Conseguir que el enjambre robótico tenga la capacidad de self-assembly.
- Formación de figuras convexas y cóncavas bidimensionales
- Formación de figuras con agujeros bidimensionales

2 Antecedentes

2.1 Inspiración en la naturaleza

Desde la antigüedad la naturaleza ha servido como inspiración para los inventores, en el caso del presente estudio, los enjambres de insectos, cardúmenes de peces, bandadas de aves o rebaños de mamíferos, sirven de inspiración ya que estos sistemas muestran un asombroso movimiento colectivo.

A pesar de que cada individuo puede reaccionar al medio por su propia cuenta, a veces entrando en conflicto con otros miembros del grupo. Los grupos responden al medioambiente de una manera altamente coordinada, en donde los individuos no entran en conflicto entre sí. [1]

2.2 Self-assembly

Algunas definiciones de self-assembly:

Self-assembly es la organización autónoma en patrones o estructuras sin la intervención humana. Los procesos de self-assembly son comunes entre naturaleza y tecnología. [2]

Self-assembly puede ser definido como un proceso en el cual componentes preexistentes (separados o partes distintas de una estructura desordenada) se organizan en patrones o estructuras sin la intervención humana. [3]

Se distinguen 2 tipos de self-assembly, el primer tipo es estático, este tipo involucra sistemas que están en equilibrio global o local y no hay disipación de energía. Por ejemplo la formación de cristales, en este tipo de proceso, aunque se requiere energía para propiciar el self-assembly, una vez se forma la estructura, esta permanece en equilibrio, por lo tanto es estable.

El segundo tipo es dinámico, en donde las interacciones responsables de formar los patrones o estructuras solo ocurren si se disipa energía. [2]

El objetivo del presente estudio es que el sistema sea capaz de formar figuras o patrones ingresados por el usuario, en otras palabras el objetivo es dotar al sistema de la capacidad de self-assembly.

2.3 Prototipo Robot vibracional

El punto de partida de la presente memoria, es el trabajo previo desarrollado en el laboratorio de síntesis de máquinas inteligentes [4], en donde se construyen unas decenas de robots vibracionales, modelo cube, como el que se muestra en la Figura 2.1, en la imagen se observa el Robot ensamblado con todos sus elementos.

2.3.1 Componentes y Ensamble

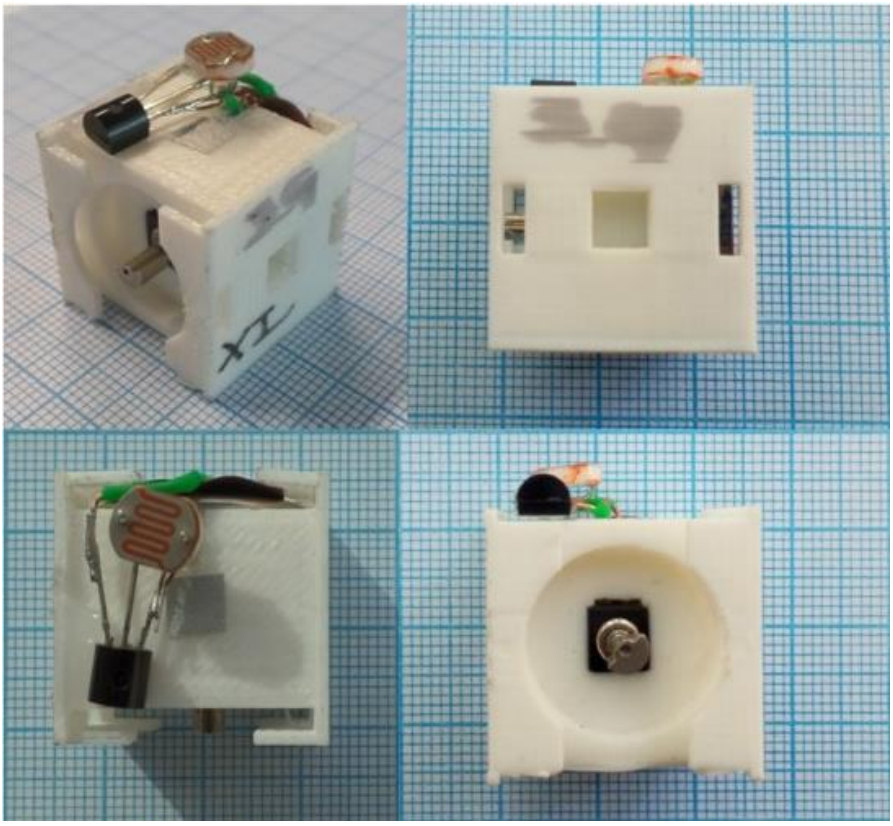


Figura 2.1 Robot vibracional Cube construido por Marco Vásquez [4]

El robot cuenta con los siguientes elementos:

- Estructura de soporte de los componentes.
- Porta-pila
- Motor vibracional.
- Batería.
- Fotorresistencia
- Transistor

El modelo de la Figura 2.1 es el cube, es un primer prototipo, que sirve de partida para desarrollar el Cheavibot. El esquemático del circuito del cube se observa en la Figura 2.2.

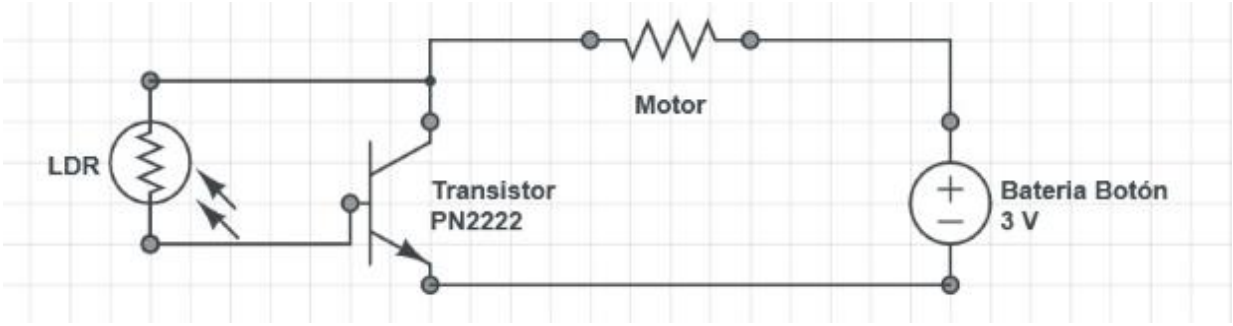


Figura 2.2 Esquema del circuito empleado en los robots Cube. Fuente: Memoria Marco Vásquez [4].

El Robot de partida de este trabajo es el Cheavibot, el cual es un modelo basado en el cube, que posee 2 grados de libertad. En la Figura 2.3 se observa el Cheavibot: en (a) se muestra el ensamble de todos los elementos en una vista isométrica, en (b) se muestran los componentes electrónicos ensamblados, se debe notar que se compone de un par de motores vibracionales, el circuito de cada motor corresponde al circuito de la Figura 2.2 y en (c) se muestra la base del Cheavibot, la cual sirve para apoyar los componentes eléctricos.

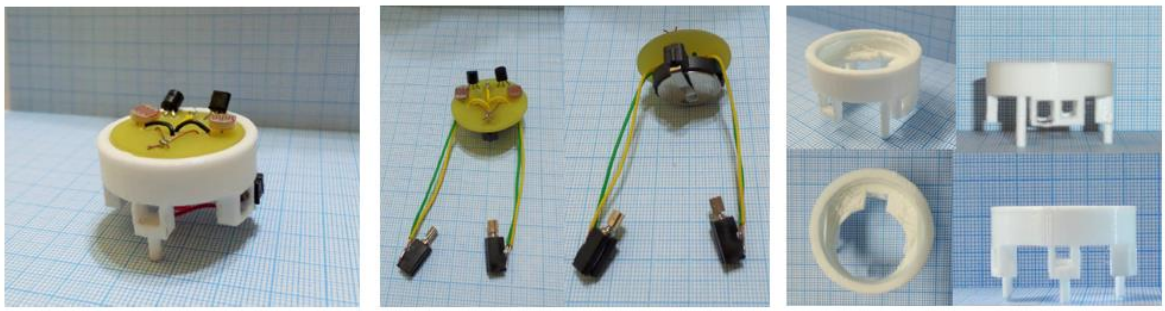


Figura 2.3 Robot Cheavibot. (a) Vista isométrica del Cheavibot ensamblado. (b) Ensamble componentes electrónicos del Cheavibot. (c) Estructura de soporte del Cheavibot [4].

2.3.2 Funcionamiento Cheavibot

El Cheavibot tiene dos fotorresistencias en la parte superior Figura 2.3 (a), (b), de forma que el circuito presentado en la Figura 2.2 funciona como un interruptor, en presencia de una señal de luz los motores se encienden, mientras en ausencia de dicha señal, permanecen apagados.

La idea es colocar filtros de color sobre las fotorresistencias, esto permite controlar al robot al dirigir distintos colores sobre la fotorresistencia. Las distintas señales de colores se transmiten mediante un proyector DLP ubicado sobre el área de prueba, la ubicación del proyector es en el techo del laboratorio de robótica.

2.4 Robótica de enjambre

La robótica de enjambre se basa en los insectos sociales, consiste en que un gran número de robots simples se coordinen entre sí. Los robots en este enfoque son simples y realizan tareas poco complejas, mediante las interacciones entre los miembros del enjambre se logra realizar tareas complejas.

Las características principales de la robótica de enjambre son:

- Escalabilidad: el número de individuos no debe representar un límite para el enjambre. Esto implica que el desempeño del enjambre debe ser independiente del número de individuos de este. Un enjambre robótico puede tener desde unas cuantas unidades hasta miles de unidades.
- Robustez: es la capacidad que tiene el enjambre de reaccionar frente a las perturbaciones externas, un sistema robusto es aquel al cual las perturbaciones no modifican su comportamiento. Algunos ejemplos de perturbaciones son: retirar miembros del enjambre, poner obstáculos en el camino del enjambre.
- Homogeneidad: las funciones y roles de los miembros del enjambre deben ser las menores posibles, al mismo tiempo el número de miembros asignados a cada tarea debe ser el mayor posible. Esta característica permite diferenciar enjambres robóticos de grupos de robot, por ejemplo el futbol de robots no se considera un enjambre, debido a que cada individuo tiene funciones y tareas únicas.
- Interdependencia: los robots deben colaborar para cumplir con la tarea especificada, cada unidad del enjambre es incapaz o ineficiente en cuanto a realizar la tarea global.
- Flexibilidad: el enjambre debe ser capaz de lograr diferentes soluciones para problemas diferentes.
- Comunicación y detección local: cada individuo del enjambre solo tienen conocimiento de lo que sucede con sus vecinos. Esta característica es la que permite la escalabilidad y flexibilidad, ya que a mayor número de individuos, crece exponencialmente el coste de la comunicación global.

2.5 Estado del arte en robots vibracionales

En la universidad de Harvard se desarrolló un modelo de enjambre robótico de bajo costo y alta escalabilidad, el kilobot es un robot del tipo vibracional [5], [6] El kilobot destaca por ser un enjambre con un alto número de individuos (1024 robots). El diseño del kilobot se considera importante, ya que tiene características comunes con el robot que empleado en esta memoria, en la Figura 2.4 se observan las principales partes del kilobot.

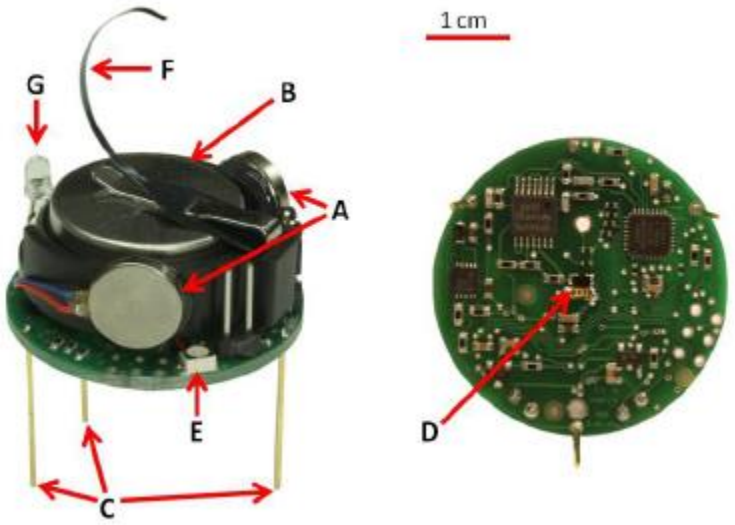


Figura 2.4 Vista isométrica (izquierda) e inferior (derecha) del kilobot, algunas partes claves son: (A) Motores vibracionales, (B) Batería ion-litio, (C) Soportes rígidos de las patas, (D) Transmisor/receptor infrarrojo, (E) LED tricolor (RGB), (F) lengüeta de carga, (G) Sensor luz ambiental. Notar la línea de 1 centímetro para escala.

Hay varios modelos de robots para enjambres robóticos, por ejemplo en [5] se hace una referencia a algunos modelos comerciales de estos robots, en la tabla 2.1, se comparan dichos modelos, se pueden observar los distintos tipos de locomoción, costos, operaciones escalables, etc.

Tabla 2.1 Comparación entre modelos de enjambres robóticos.

Robot	Cost	Scalable Operations	Sensing	Locomotion, speed	body size (cm)	battery life (hours)
Kilobot	$14*	charge, power, program	distance, ambient light	vibration, 1cm/s	3.3	3-24
E-puck[3]	$1,300	none	camera, distance, bearing	wheel, 13cm/s	7.5	1-10
Jasmine[11]	$130*	charging	distance, bearing, light color	wheel, N/A	3	1-2
R-One[9]	$220*	none	visible light, 3d accel, 2d gyro bump, IR sensors, encoders	wheel, 30cm/s	10	6
SwarmBot[8]	N/A**	charge, power, program	Range, bearing,camera, bump	wheel, 50cm/s	12.7	3

De la Tabla 2.1, se puede observar que los robots que se desplazan por medios vibracionales, tienen el menor avance. Aunque los sistemas vibracionales presentan un menor desplazamiento, esto se compensa por el menor costo del sistema motriz, en comparación con los robots que se mueven mediante ruedas.

Hay otros ejemplos de robótica de enjambre que no están relaciónanos con el self-assembly, por ejemplo en la Figura 2.5 se pueden observar aplicaciones tales como: una mesa que mantiene el balance aunque cambie la forma del terreno; un puente que se adapta al terreno, para siempre tener la calzada recta; la pinza modular que es una aplicación donde cada brazo de la pinza funciona de manera independiente, cada brazo solo puede medir la presión local; una pantalla en relieve 3D. [7]

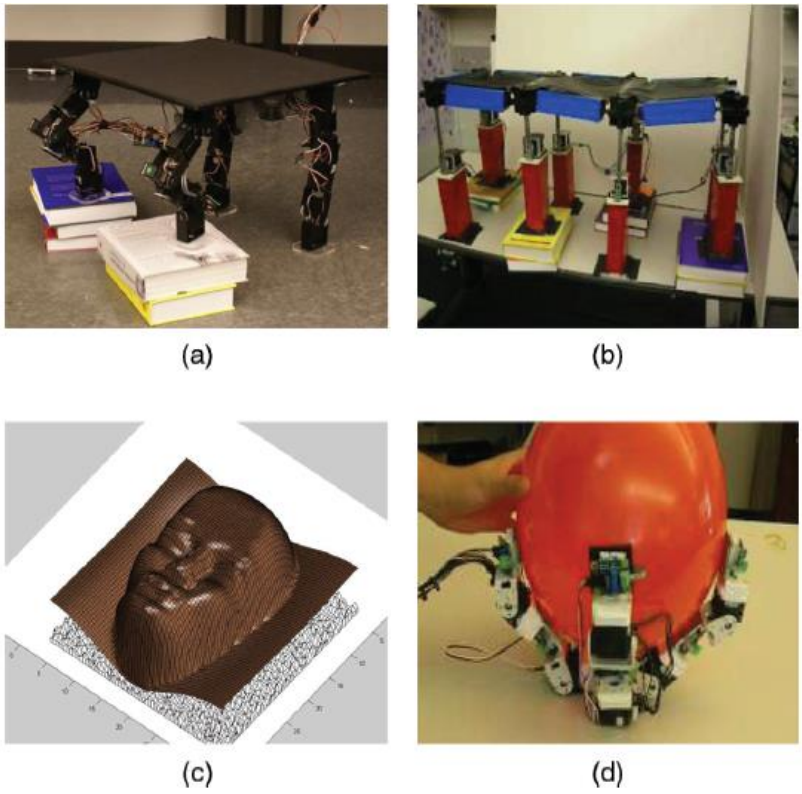


Figura 2.5 Diferentes tareas de robots modulares (a) Mesa auto-balanceada, (b) Puente que se adapta al terreno, (c) Pantalla 3D en relieve, (d) Pinza modular.

2.6 Sistema de captura de movimiento OptiTrack

Es un sistema de cámaras que permite capturar el movimiento del cuerpo humano o de cualquier objeto rígido. El modelo V120 Trio cuenta con 6 grados de libertad, tiene 3 cámaras en un soporte de aluminio, cada cámara cuenta con 26 led, en la Figura 2.6 se pueden observar a la izquierda el modelo de la cámara, mientras a la derecha se observa con los led encendidos. [8]

Figura 2.6 Cámara OptiTrack modelo V120 Trio. [8],[9]

OptiTrack funciona detectando la luz emitida o reflejada por los objetos, para obtener mejor medidas, se utilizan reflectores especiales, llamados marcadores. El sistema almacena los datos de las posiciones de los marcadores, cada cámara genera imágenes en 2D, luego se calcula la posición y superponiendo los datos se calcula la posición 3D del objeto. [10]

Un rígido se define como un sólido indeformable, que debe tener un mínimo de 3 marcadores, para poder rastrear el movimiento. El sistema OptiTrack cuenta con precisión milimétrica del orden de ±0.1mm.

2.7 Sistemas de control [11]

2.7.1 Sistema de control de lazo cerrado

Un sistema que mantiene una relación determinada entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control, se denomina sistema de control realimentado. Un ejemplo sería el sistema de control de temperatura de una habitación. Midiendo la temperatura real y comparándola con la temperatura de referencia (temperatura deseada), el termostato activa o desactiva el equipo de calefacción o de enfriamiento para asegurar que la temperatura de la habitación se mantiene en un nivel confortable independientemente de las condiciones externas.

Sistemas de control en lazo cerrado. Los sistemas de control realimentados se denominan también sistemas de control en lazo cerrado. En la práctica, los términos control realimentado y control en lazo cerrado se usan indistintamente. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación (que puede ser la propia señal de salida o una función de la señal de salida y sus derivadas y/o integrales), con el fin de reducir el error y llevar la salida del sistema a un valor deseado. El término control en lazo cerrado siempre implica el uso de una acción de control realimentado para reducir el error del sistema.

En la Figura 2.7 se observa un esquema de lazo cerrado, en la imagen de la izquierda se compara directamente la salida con la entrada, la función G(s) es conocida como función de transferencia y se puede interpretar como los cambios que sufre la señal de entrada en el sistema. En la imagen de la derecha a la señal de salida se le aplica una función H(s) que es conocida como función de transferencia de retroalimentación, esta función implica los cambios a los que se somete la señal de salida para poder ser comparada con la señal de entrada

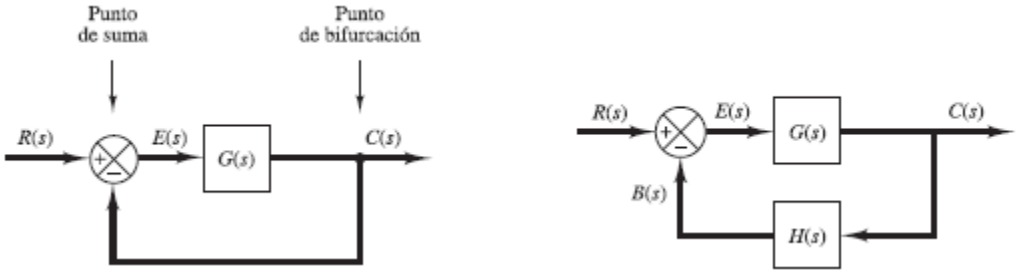


Figura 2.7 Esquema de un lazo cerrado de control.

2.7.2 Control PID

Un controlador PID ampliamente usado en la industria, es un mecanismo de control por realimentación, por lo tanto es en lazo cerrado. Consiste en 3 parámetros, los cuales son el proporcional, el integral y el derivativo. Cada uno de estos controladores está representado por una constante, en la Figura 2.8 se observa la representación completa de un control PID, donde Kp es la constante proporcional, Ti es el tiempo integral y Td es el tiempo derivativo.

La parte proporcional depende del error actual, la parte integral de los errores pasados y la parte derivativa predice errores futuros.

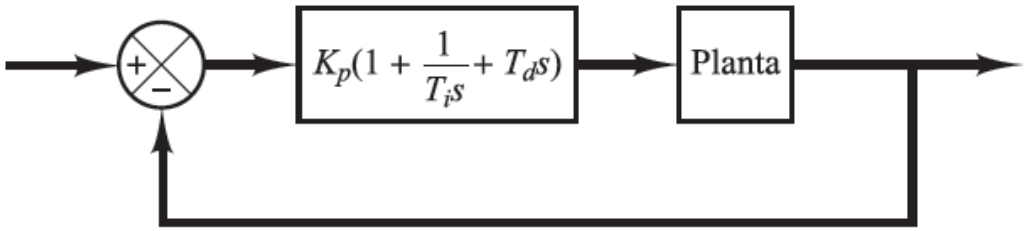


Figura 2.8 Esquema de un controlador PID.

Un control PID se logra sumando las tres componentes, a continuación se estudian en detalle estas componentes.

2.7.3 Control Proporcional

El control proporcional en multiplicar la señal de error por alguna constante, la cual recibe el nombre de constante proporcional K_p. El control proporcional permite disminuir el error en estado estacionario, sin embargo no lo elimina del todo, sino que hay una pequeña variación entre el valor deseado (set point) y el valor de salida, dicha variación conocida como offset se presenta en la Figura 2.9.

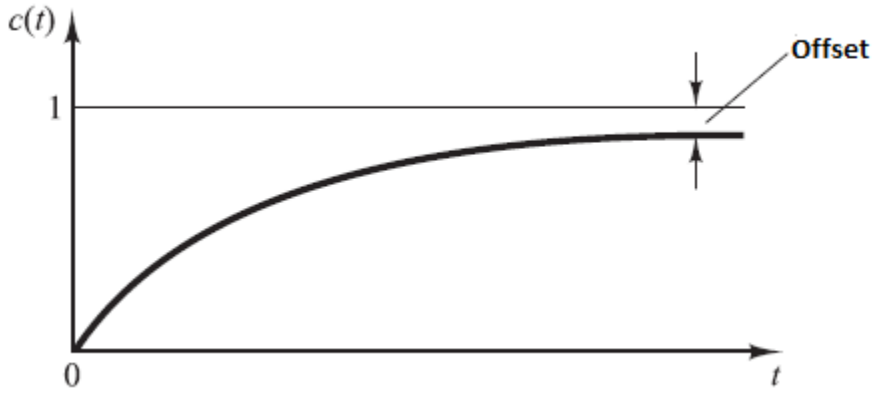


Figura 2.9 Respuesta a un escalón unitario de un controlador proporcional, la diferencia entre el set-point y la salida es el offset.

La representación del controlador proporcional se presenta en (2.1)

$$P = K_p e(t) \tag{2.1}$$

2.7.4 Control Integral

Un controlador integral permite eliminar el offset que se produce debido a un controlador proporcional. En el control integral la señal de control u(t), que es la señal que sale desde el controlador es en todo momento el área bajo la curva de la señal de error e(t) hasta tal momento. Como se puede apreciar en la Figura 2.10, la señal de control es diferente de cero cuando el error es cero, en el caso de un controlador proporcional esto no es posible, ya que cuando la señal de control es cero, el error también es cero

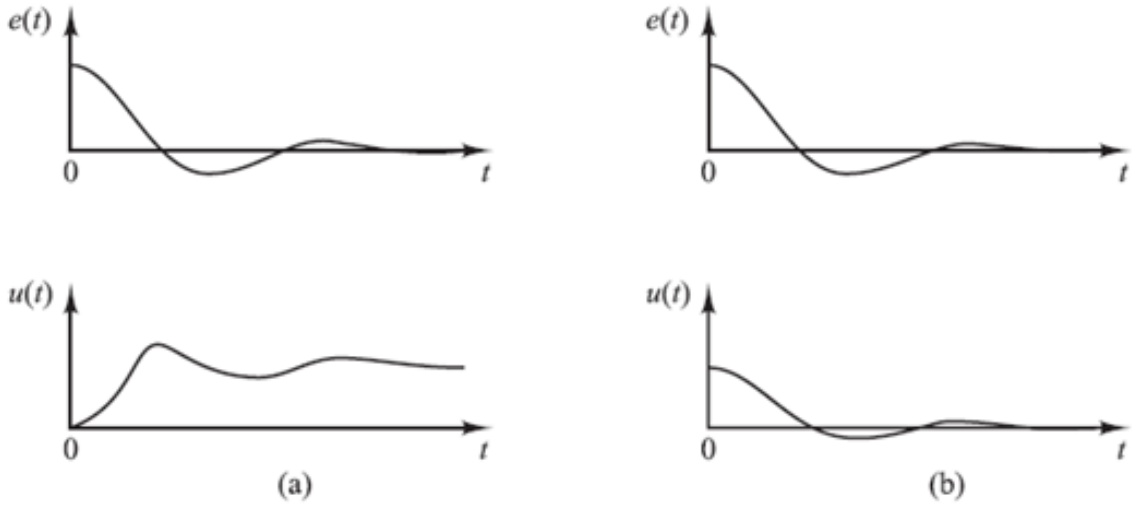


Figura 2.10 Graficas de las curvas e(t) y u(t). (a) Control integral, la curva de control es distinta de cero cuando la señal de error es cero (b) Control proporcional, la curva de control es cero cuando la señal de error es cero.

Un controlador integral se puede representar mediante (2.2).

$$I = K_i \int_0^t e(t)dt \tag{2.2}$$

2.7.5 Control Derivativo

El control derivativo responde a la tasa de cambio del error, de modo que si el error no cambia su valor el control derivativo no entra en acción. Un control derivativo tiene una alta sensibilidad, tiene la ventaja de producir una corrección significativa del error antes que este se vuelva demasiado grande, esto sucede porque responde a las variaciones del error.

El control derivativo se anticipa al error, iniciando una acción correctiva oportuna aumentando la estabilidad del sistema.

El control derivativo siempre se usa en conjunto con un control proporcional PD o con un control integral proporcional PID, porque de lo contrario el sistema oscilaría infinitamente como se muestra en la Figura 2.11.

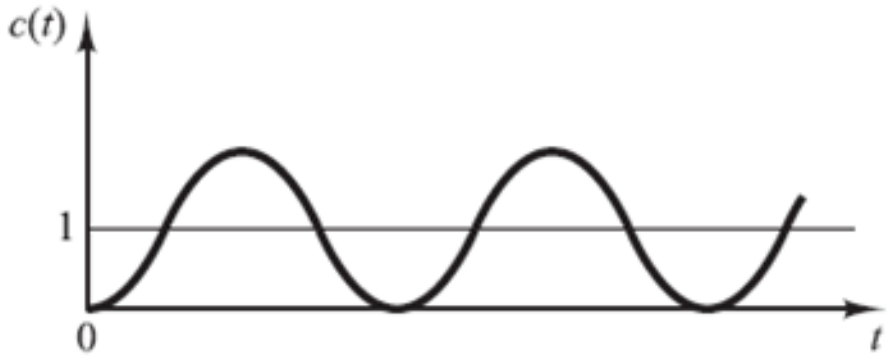


Figura 2.11 Respuesta a una entrada escalón unitaria de un controlador derivativo.

El controlador derivativo se puede representar mediante (2.3).

$$D = K_d \frac{de}{dt} \tag{2.3}$$

2.8 Conceptos básicos de mecánica estadística en el movimiento de enjambres

Para simular el movimiento de enjambres se utilizan modelos basados en la mecánica estadística. A diferencia de la mecánica estadística en donde se analizan grandes conjuntos de partículas, en los enjambres usualmente se trabaja con algunas decenas de elementos, aunque hay casos en donde se trabaja con miles.

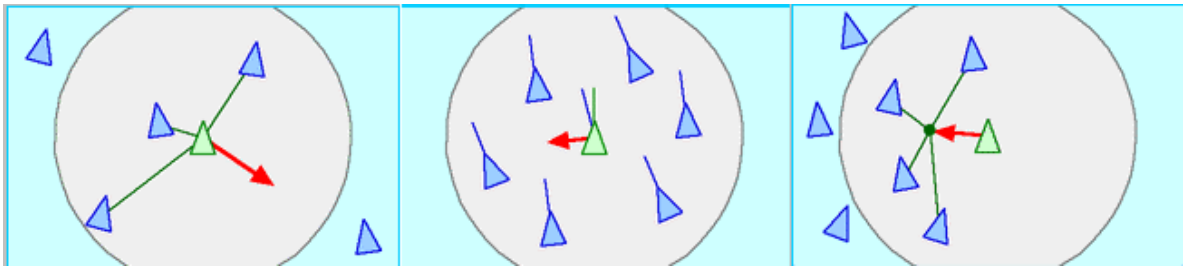
Se asume que los elementos del enjambre tienen las siguientes características:

- Son similares
- Se mueven con una velocidad constante y son capaces de cambiar de dirección
- Interactúan dentro de cierto rango, cambiando su dirección de movimiento, de tal manera de conseguir un alineamiento con los vecinos.
- Existe cierto ruido de magnitud variable

2.9 Modelos Básicos para partículas auto-propulsadas (SSP)

Para el modelamiento de sistemas colectivos las trayectorias de los individuos son determinadas mediante ecuaciones diferenciales, se consideran 3 tipos de interacciones: evitar las colisiones, mantener la orientación de los vecinos y tratar de mantenerse dentro

del centro de masas del grupo. En la Figura 2.12 se muestran las interacciones. El modelo es determinista y tiene parámetros ajustables.



(a) (b) (c)

Figura 2.12 Interacciones del modelo SPP (a) separación: evitar colisiones, (b) alineación: orientarse de acuerdo a los vecinos, (c) Cohesión: moverse hacia la posición promedio de los vecinos. Fuente: http://www.red3d.com/cwr/boids/

2.9.1 Modelo de Vicsek estándar (SVM) [12]

En este modelo las perturbaciones son consideradas como una consecuencia natural de una variedad de factores. En este modelo las partículas auto-propulsadas se mueven con una velocidad absoluta ajustada y se asume una dirección promedio de los individuos dentro de una distancia R.

La velocidad y la posición de la partícula i con vecinos j, se puede determinar por las ecuaciones (2.4) y (2.5) respectivamente.

$$\vec{v}_i(t+1) = v_0 \frac{\langle \vec{v}_j(t) \rangle_R}{\left| \langle \vec{v}_j(t) \rangle_R \right|} + perturbación \tag{2.4}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{2.5}$$

Aquí $\langle \ldots \rangle_R$ denota el promedio de las velocidades dentro de un círculo de radio R que rodea a la particular i.

2.9.2 Modelo de Cucker-Smale (CS) [13]

En este modelo se plantea que los elementos procurar alcanzar una dirección común, para un grupo de i= 1,…, k elementos moviéndose en un espacio tridimensional, donde la posición de cada elemento es $x_i = x_i(t)$, se define la matriz de proximidad $A = (a_{ij})$, donde el elemento a_{ij} mide la influencia de los otros elementos sobre el elemento i, el parámetro β sirve para ajustar la magnitud de la influencia de los otros elementos. Lo anterior se resume en (2.6)

$$a_{ij} = \frac{1}{\left(1 + \left\| x_i + x_j \right\|^2\right)^{\beta}} \tag{2.6}$$

2.10 Problema vecinos cercanos con radio fijo

El problema de los vecinos cercanos con radio fijo (fixed radius near neighbors) es una variante del problema de búsqueda de vecinos cercanos (near neighbor search-NNS-). El problema de la búsqueda de vecinos cercanos es un problema de optimización: el objetivo es encontrar un punto que minimice cierta función objetivo (en este caso, la distancia desde el punto consultado). El problema NNS se define como: Dado un conjunto de puntos S en un espacio métrico M, y un punto q en M, encontrar los K puntos más cercanos en S a q.

El problema de encontrar vecinos cercanos en un radio fijo se define como: dado un conjunto de puntos S en un espacio métrico M (frecuentemente euclidiano), una distancia D, un radio R y un punto q en M, encontrar el conjunto S' ⊆ S tal que:

S' = {p ∈ S: D (q, p) <= R} [14]

2.11 Métodos de resolución del problema. [15]

Básicamente existen 4 métodos principales, para resolver el problema de encontrar los vecinos próximos dentro de un radio fijo:

1. Fuerza bruta
2. Proyección
3. Celdas
4. K-d tree

2.11.1 Fuerza Bruta.

Este método consiste en guardar cada uno de los n elementos en un array, lista u otro tipo de secuencia. Cuando se produce una consulta, todos los elementos son escaneados y todos los vecinos cercanos en un radio fijo son enumerados. Esta técnica involucra espacio lineal para almacenar la estructura, tiempo de pre procesado lineal en n y cada consulta es respondida con n cálculos de distancia. Este algoritmo tiene un orden de n^2, $O(n^2)$.

Aunque este método es poco sofisticado, en el sentido de que realiza muchos cálculos de distancias, para un pequeño set de datos (especialmente en altas dimensiones), el método de fuerza bruta es difícil de batir.

2.11.2 Proyección

El método consiste en mantener, por cada dimensión, una secuencia de puntos en el espacio ordenados por aquella dimensión. Estas k listas pueden ser obtenidas usando algoritmos estándar de ordenamiento en un tiempo de O(k*n*log n) y guardados usando O(n*k) palabras de almacenamiento. Después del pre proceso, una consulta por todos los vecinos cercanos con radio fijo al punto x, puede ser respondida con el siguiente método de búsqueda: se escoge una dimensión, por generalidad se denomina a la dimensión seleccionada como la dimensión i, se busca la posición de x en la secuencia i, usando una búsqueda binaria (de costo O(log n). Luego se escanea la lista hacia abajo hasta encontrar un valor menor a xi - r, de manera análoga se escanea los valores hacia arriba hasta encontrar un valor mayor a xi + r. Todos los puntos que están a menos de una distancia r de x, están entre los valores extremos de los puntos ya encontrados.

Si los puntos se encuentras distribuidos de manera aleatoria entre todas las dimensiones, entonces basta guardar solo una secuencia ordenada. Sin embargo, si existe mucho agrupamiento en ciertos lugares del espacio. Entonces es más conveniente, guardar secuencias ordenadas para cada una de las dimensiones. Para escoger que dimensión utilizar para una consulta dada, se examina la densidad local alrededor de x para cada dimensión, y se escoge la dimensión que tiene la menor densidad.

2.11.3 Celdas

El método de celdas es apropiado cuando el conjunto de puntos está distribuido de manera uniforme en un espacio euclidiano. Por ejemplo, el método de celdas, puede ser apropiado para representar las ciudades en un mapa, con dos dimensiones en un espacio euclidiano siendo estas la longitud y la latitud. En el método de celdas los datos se estructuran colocando un “tablero de ajedrez” sobre el mapa y asignando cada cuidad a una casilla del tablero. Una estructura de celdas se puede ampliar a dimensiones mayores, en este caso se considera que los puntos están dentro de hipercubos.

El método de celdas tiene un orden que es proporcional al número de puntos n y a los pares de puntos a reportar k, O(n+k). Este método es adecuado cuando el conjunto de datos esta uniformemente distribuido y se trabaja en espacios de baja dimensión.

2.11.4 K-d tree

El árbol de búsqueda binaria multidimensional (en k dimensiones k-d tree), es una generalización de un árbol de búsqueda binaria estándar, los k-d tree se describen en más detalle en [16]. En un árbol de búsqueda binaria estándar, la decisión de proceder a uno de los dos hijos de un nodo es hecha comparando la clave de consulta con el valor de la clave guardada en el nodo. Debido a que hay k claves asociadas a cada punto, este esquema se puede extender a k dimensiones especificando en el nodo, no solo el valor contra cual se hace la comparación, sino también cual dimensión debe ser comparada.

El almacenamiento de un k-d tree es proporcional a n, mientras el tiempo de pre procesado requerido para construir el árbol es de O(k*n*log n)

2.12 Distintos tipos de consultas

Existen distintos tipos de consultas, las cuales hacen que el algoritmo a usar varié de un caso a otro, ya que la respuesta debe estar en un tiempo óptimo. Los tipos de consulta son:

1. Modelo Online
2. Modelo en Batch
3. Problema de todos los pares cercanos
4. Conjunto dinámico de puntos

2.12.1 Modelo Online

En el modelo online aplicado a la búsqueda de los vecinos cercanos con radio fijo, en el procedimiento de búsqueda, se entrega la colección de puntos y suficiente tiempo para organizar los puntos una estructura de datos adecuada. Después de esto el sistema es consultado repetidas veces para buscar todos los vecinos cercanos con radio fijo al punto consultado.

2.12.2 Modelo en Batch (lotes)

En este modelo, las consultas de los vecinos cercanos con radio fijo a un punto, no llegan al sistema de forma aleatoria, sino que llegan en lotes. Esto ocurre cuando las consultas son generadas por un número de usuarios en terminales remotos y luego son reunidas en concentradores antes de ser mandados en un lote al computador central.

Las consultas en lote se pueden resolver iterando el modelo online, sin embargo existen algoritmos más sofisticados para tratar este problema.

2.12.3 Problema de todos los pares cercanos

El problema de todos los pares cercanos se puede enunciar como: dada una colección de n puntos en el espacio k, enumerar todos los pares entre n dentro de una distancia r entre cada uno. Este problema se puede abordar usando el modelo online o en batch, pero hay mejores métodos que pueden ser usados. Una razón para esto es que las anteriores soluciones enumeran todos los pares cercanos dos veces: una cuando x está dentro de una distancia r de y; dos cuando y está dentro de una distancia r de x. Evitar esta redundancia puede significar un factor de 2 en el aumento de la velocidad de procesamiento.

2.12.4 Conjunto dinámico de puntos

Muchas aplicaciones de búsqueda de vecinos cercanos con radio fijo tratan con conjuntos de puntos que son dinámicos, esto significa, los puntos pueden ser insertados, eliminados o cambiar su posición. Cuando estos eventos ocurren, es deseable cambiar lo menos posible la estructura de datos.

El tipo de cambio más simple es cuando un punto es modificado. Cuando la modificación es una inserción o eliminación, la correspondiente estructura puede ser fácilmente modificada usando los algoritmos de fuerza bruta, proyección o celdas. Inserciones o eliminaciones con k-d tres son más difíciles. Las inserciones o eliminaciones en k-d tres, se pueden realizar insertando o eliminando ítems en la celda, pero esto es desastroso si sucesivas inserciones se realizan en solo pocas celdas, ya que el árbol crecería fuera de balance. En este caso el árbol no debería usar celdas sino guardar los puntos en los nodos del árbol usando las rutinas dinámicas de inserción y eliminación como se describen en [16]. Para cualquier de las estructuras, si el conjunto de todos los pares cercanos en el conjunto de puntos es conocido antes de un cambio de posición del punto x, el nuevo conjunto de todos los pares cercanos puede ser fácilmente calculado después del cambio de posición de x. Los únicos pares cercanos previos que ya no son pares cercanos son los puntos que estaban cerca de x antes de que x cambiara su posición, y los únicos nuevos pares cercanos son los nuevos vecinos de x.

Cualquier lote de cambios puede ser manejado iterando sobre las técnicas anteriores. Una situación más radical ocurre cuando todos los puntos en la estructura cambian sus posiciones. Si las nuevas posiciones están totalmente no relacionadas con las antiguas, no hay mucho que se pueda hacer además de desechar la antigua estructura y empezar nuevamente desde cero.

Frecuentemente se da el caso, que las nuevas posiciones se encuentran relacionadas con las antiguas. Por ejemplo en el sistema de tráfico aéreo, el cual actualiza la posición de los aviones cada 5 segundos. Es seguro asumir que ningún avión habrá viajado más que una milla desde su posición previa dentro de esos 5 segundos. Usando esta restricción (un punto

no se habrá desplazado más que cierta distancia constante de su antigua posición), es posible desarrollar algoritmos de actualización muy rápidos. Para la técnica de proyección, la nueva proyección se parecerá a la antigua proyección, y una variación del algoritmo bubble sort puede ser usado para modificar la proyección. Para k-d tree se puede usar una técnica similar, cuando la posición de cada punto cambia, probar si el cambio es lo suficientemente grande para causar que el punto se mueva de la presente celda a una nueva celda.

2.13Carga y descarga de la batería [17]

Al descargar una batería tipo CR2032 con una corriente mayor a la nominal (200µA), se produce un fenómeno de descarga rápida en donde el voltaje de la batería disminuye rápidamente. sin la presencia de este fenómeno el tiempo de vida útil de la batería se obtiene dividiendo la carga por la corriente de descarga, sin embargo debido a que la química de la batería no puede seguir el paso cuando se tiene una descarga con una corriente mayor a la nominal, el tiempo de uso se reduce drásticamente y se produce la caída de voltaje de la pila, sin embargo la pila vuelve a recuperar el voltaje nominal si se deja de descargar, el tiempo de recuperación del voltaje es igual al tiempo que se tarda en la descarga, esto se observa en el gráfico de la Figura 2.13.

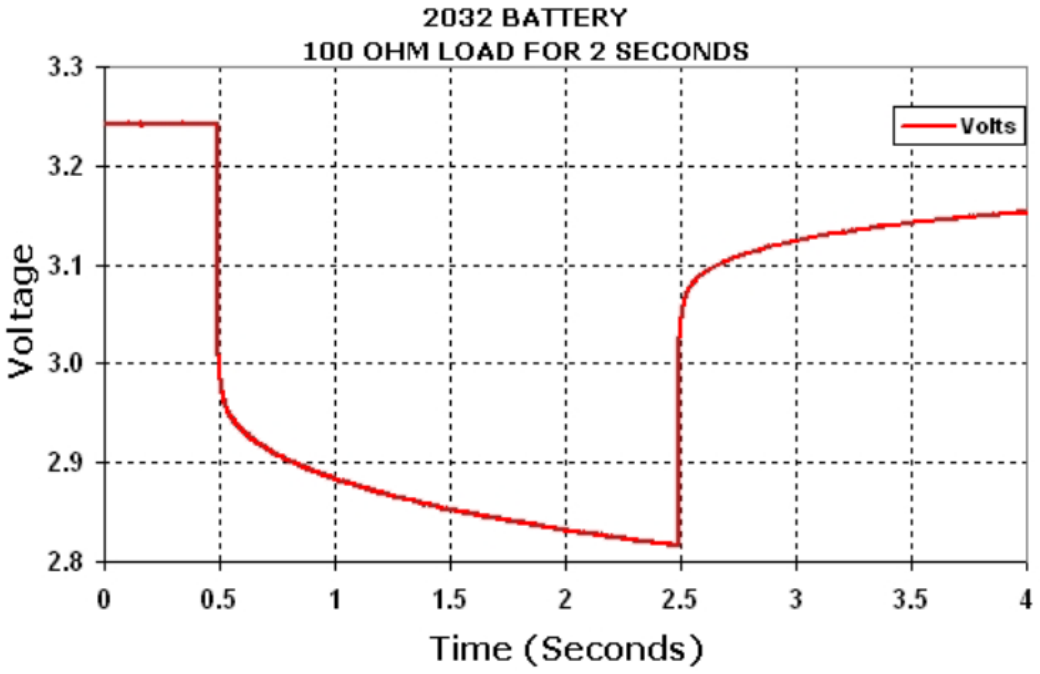


Figura 2.13 Voltaje instantáneo en una batería CR2032, la batería se descarga por 2 segundos y luego se deja en reposo.

2.14 Dinámica robot vibracionales. [18]

Para describir la dinámica de una plataforma robótica que se mueve mediante el mecanismo de slip and stick. Se parte de un modelo simplificado de un grado de libertad, luego se describe a la plataforma con 2 grados de libertad. Se analizan la plataforma robótica como cuerpo rígido y posteriormente como cuerpo deformable.

2.14.1 Principio de movimiento

El principio de movimiento es primeramente demostrado usando un modelo simplificado de una plataforma móvil de un grado de libertad de masa M. El mecanismo de movimiento emplea una masa excéntrica de masa m que rota debido al motor O montado en la plataforma según se observa en la Figura 2.14.

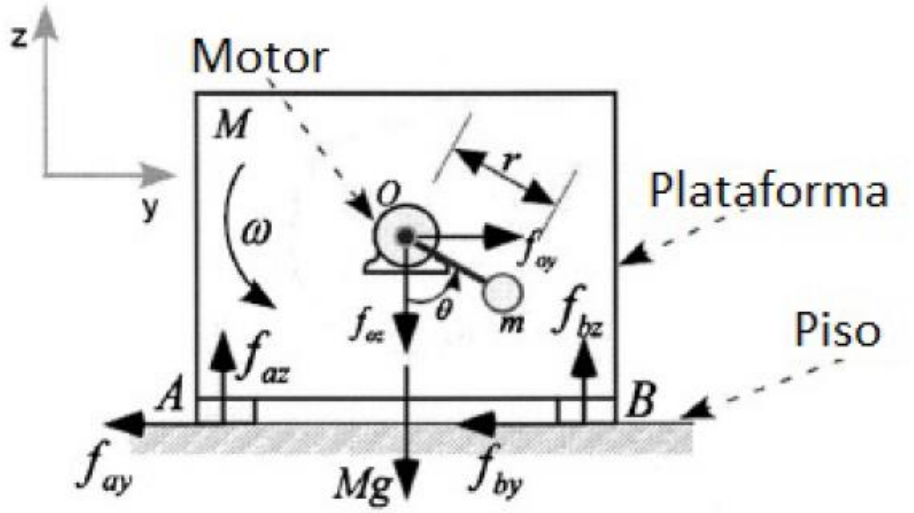


Figura 2.14 Modelo simplificado plataforma robótica de un grado de libertad con masa rotatoria m

Se asume que la masa m rota en un plano vertical con velocidad angular constante, en torno al punto O, y que la plataforma está restringida a moverse solo sobre el eje y. Un ciclo de operación se completa cuando la masa ha descrito un ángulo de 360°. Las fuerzas gravitacional y centrifuga ejercidas en la masa rotatoria se resuelven sobre los ejes y, z para producir (2.7) y (2.8).

$$f_{oy} = mr\omega^2 \sin\theta \tag{2.7}$$

$$f_{oz} = -mg - mr\omega^2 \cos\theta \tag{2.8}$$

Donde g es la aceleración de gravedad y r es la longitud entre m y O. Estas fuerzas también son aplicadas a la plataforma en el punto O, mientras el momento debido a la pequeña masa excéntrica es despreciado. Cuando la velocidad angular ω es baja, la plataforma no se mueve debido a que la fuerza horizontal actuante f_{oy} se cancela por la fuerza de fricción que

ocurre en los puntos de contactos de la plataforma A y B. Sin embargo, si la velocidad angular ω excede el valor crítico, luego f_{oy} supera la fuerza de fricción de Coulomb aplicada en los dos puntos de soporte, y como resultado, la plataforma empieza a deslizar.

Usando un modelo simplificado sobre la fricción estática – cinética [19], el movimiento de la plataforma sobre los ejes z e y se describe por las ecuaciones (2.9) y (2.10), una simulación numérica se presenta en la Figura 2.15.

$$M\ddot{y} = f_{oy} - f_{fr} \tag{2.9}$$

$$0 = f_{az} + f_{bz} + (-Mg + f_{oz}) \tag{2.10}$$

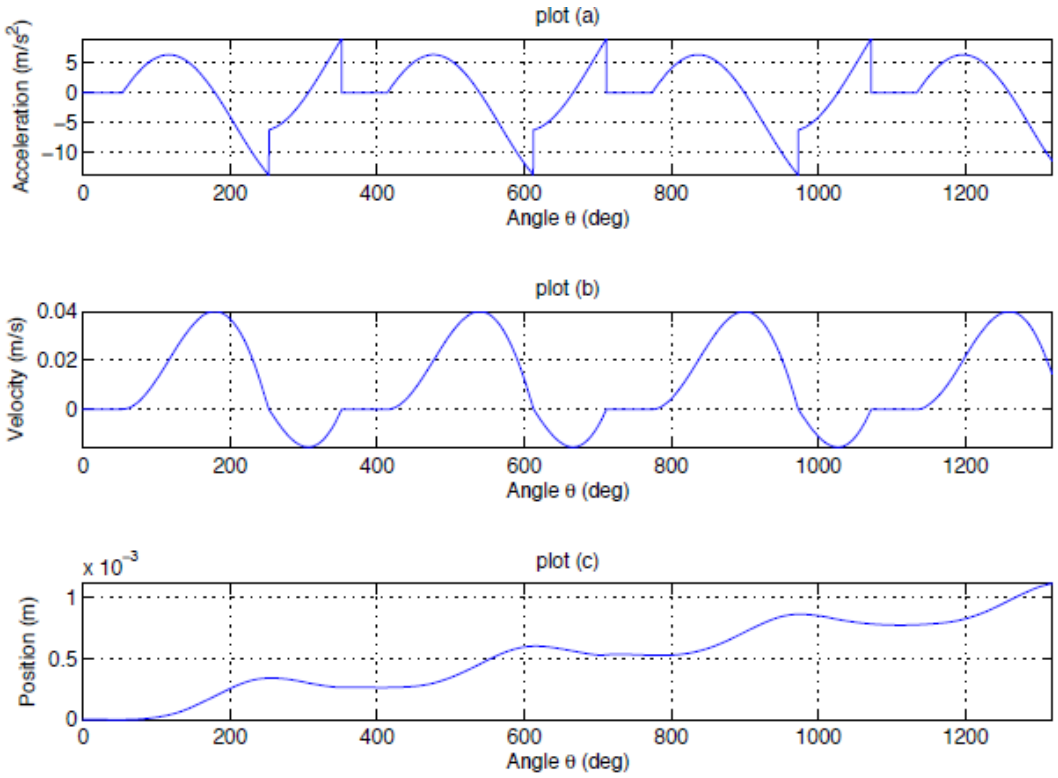


Figura 2.15 Simulación del movimiento de la plataforma con un grado de libertad.

2.14.2 Movimiento planar de una plataforma de dos grados de libertad

El principio de actuación mencionado anteriormente es empleado para diseñar un micro robot de 2 grados de libertad. El diseño debe cumplir un número de objetivos para el pertinente micro posicionamiento [20]. El primer diseño cuenta con 3 actuadores y permite movimiento planar con 3 grados de libertad. Sin embargo, mediante simulaciones e investigación analítica se muestra que cuando los 3 actuadores operan, las componentes de la fuerza se cancelan entre sí resultando en cero trabajo y así reduciendo la eficiencia del robot. Además durante el movimiento de 3 grados de libertad, aparece un acoplamiento en

el accionamiento y el movimiento inducido requiere algoritmos de control más sofisticados. Por lo tanto, se decidió que los primeros prototipos deberían emplear 2 actuadores en lugar de 3, permitiendo solo movimiento planar local con 2 grados de libertad. De esta manera se gana simplicidad operacional y eficiencia energética a expensas de un grado de libertad. La base del diseño de 2 grados de libertad se muestra en la Figura 2.16.

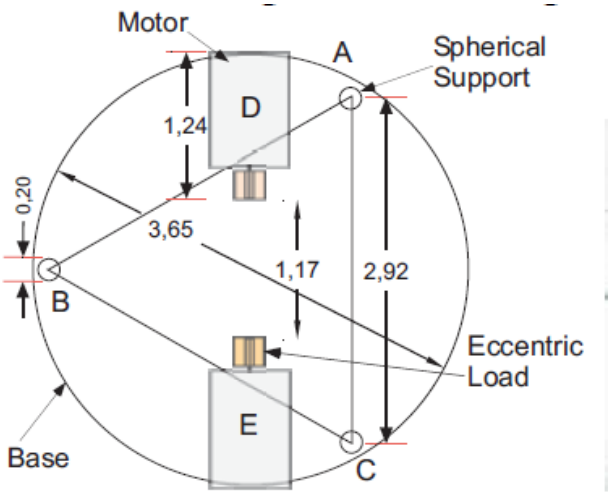


Figura 2.16 diseño de la base

La geometría de la base es un disco. Los puntos de contacto (soporte) A, B y C están dispuestos en un triángulo equilátero. Se asume que no hay simetría en la plataforma y así el centro de masa no necesariamente coincide con el centro geométrico. Los actuadores están alineados con el diámetro de la base y están con direcciónanos opuestamente entre sí. La carga excéntrica es capaz de rotar en ambos sentidos sobre el eje del motor. El movimiento inducido por la plataforma depende de la velocidad de giro del actuador, la distribución de la inercia de la plataforma y la distribución de la fricción con el piso.

2.14.3 Análisis dinámico

La descripción de la dinámica de la plataforma robótica requiere el uso de 3 modelos dinámicos: Primero, la dinámica de la plataforma, la cual es expresada por las correspondientes ecuaciones de movimiento. Segundo, un sistema masa – resorte que modela la deformación de la base de la plataforma. Tercero, un modelo dinámico que describe la respuesta de los motores vibracionales.

2.14.3.1 Dinámica de la plataforma

Las únicas suposiciones usadas para la dinámica de la plataforma son: (i) la carga desbalanceada puede ser modelada como un punto de masa m, rotando a una distancia r del eje del motor. (ii) Los puntos de contacto de la plataforma experimentan fricción de Coulomb. No se hacen suposiciones con respecto a la velocidad relativa y la fase de los actuadores o la distribución de inercia de la plataforma. El análisis de la plataforma

involucra el marco de referencia fijo en el cuerpo {B, x, y} y el marco de referencia inercial {O, x, y}, mostrado en la Figura 2.17.

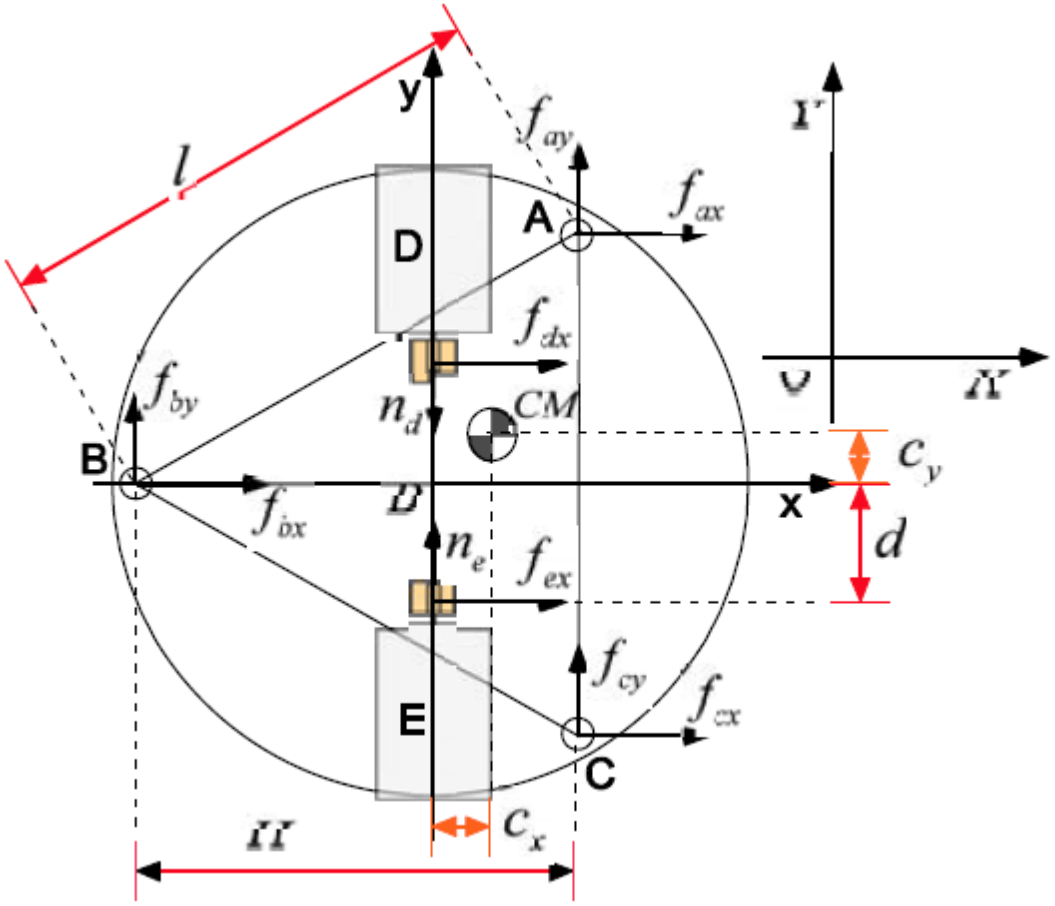


Figura 2.17 Fuerzas aplicadas en la plataforma

La notación adoptada es ${}^{i}f_{j}$ donde el superíndice i corresponde al marco de referencia y el subíndice j corresponde a la componente x, y, z. el superíndice b denota el marco de referencia B. el marco de referencia O no usa superíndice. Los vectores de posición de los puntos de contacto A, B y C están denotados por ${}^{b}\mathbf{r}_{a}$, ${}^{b}\mathbf{r}_{b}$, ${}^{b}\mathbf{r}_{c}$ y el vector de posición del eje del motor D y E en donde las cargas desbalanceadas son aplicadas están denotados por ${}^{b}\mathbf{r}_{d}$, ${}^{b}\mathbf{r}_{e}$. Fuerzas ${}^{b}\mathbf{f}_{a}$, ${}^{b}\mathbf{f}_{b}$, ${}^{b}\mathbf{f}_{c}$ incluyen la fuerza normal y friccional por contacto en los puntos de contacto A, B y C, respectivamente. El ángulo θ, es el ángulo de la masa excéntrica con respecto al ángulo vertical ver Figura 2.14. Debido a la excentricidad en la rotación, fuerzas ${}^{b}\mathbf{f}_{d}$, ${}^{b}\mathbf{f}_{d}$, son aplicadas en los puntos D y E de la plataforma, y momentos ${}^{b}\mathbf{n}_{d}$, ${}^{b}\mathbf{n}_{e}$ son aplicados sobre el eje del motor, ver Figura 2.17. Las componentes están dadas por (2.11).

$$
\begin{aligned}
{}^{b}f_{ix} &= -m_i r_i \omega_i^2 \sin\phi_i \sin\theta \\
{}^{b}f_{iy} &= 0 \\
{}^{b}f_{iz} &= -m_i g - m_i r_i \omega_i^2 \cos\theta \\
{}^{b}n_{ix} &= 0 \\
{}^{b}n_{iy} &= -m_i g r_i \sin\phi_i \sin\theta \\
{}^{b}n_{iz} &= 0
\end{aligned}
\qquad (2.11)
$$

Donde i = {d, e}, ω_i es la velocidad angular del motor i, r_i es la excentricidad de la masa m_i y φ_i= {90,-90} son los ángulos de posición de los vectores ${}^b\mathbf{r}_d$, ${}^b\mathbf{r}_e$. Luego las ecuaciones de Newton – Euler son dadas por (2.11) y (2.12). [21].

$$M\dot{v} = R\sum_i {}^b f_i \qquad \mathrm{i} = \{\mathrm{a,b,c,d,e}\} \tag{2.12}$$

$${}^b I\dot{\omega}_p + {}^b\omega_p \times {}^b I\,{}^b\omega_p = \sum_i ({}^b r_i \times {}^b f_i) + \sum_i {}^b n_j \qquad \mathrm{i} = \{\mathrm{a,b,c,d,e}\},\ \mathrm{j} = \{\mathrm{d,e}\} \tag{2.13}$$

Donde R es la matriz de rotación entre los marcos de referencia B y O, ω_p es la velocidad angular de la plataforma. ${}^b\mathbf{I}$ es la matriz de inercia, y $v = [\dot{x}, \dot{y}, \dot{z}]^T$ es la posición del centro de masa en el marco de referencia inercial.

2.14.3.2 Dinámica de cuerpo deformable

Mientras la plataforma es estática y las fuerzas actuantes gradualmente se incrementan, las fuerzas distribuidas a las patas de la plataforma alcanzan el nivel de Coulomb y el movimiento es impedido. Se requiere tener conocimiento de la distribución de fuerza en cada uno de los soportes A, B y C de la plataforma durante condiciones estáticas. Estas 6 incógnitas pueden ser determinadas considerando pequeñas deformaciones sobre la base de la plataforma. Para este propósito, la plataforma es modelada como un sistema de masa distribuido, constituido por 3 masas puntuales conectadas mediante resortes como se muestra en la Figura 2.18.

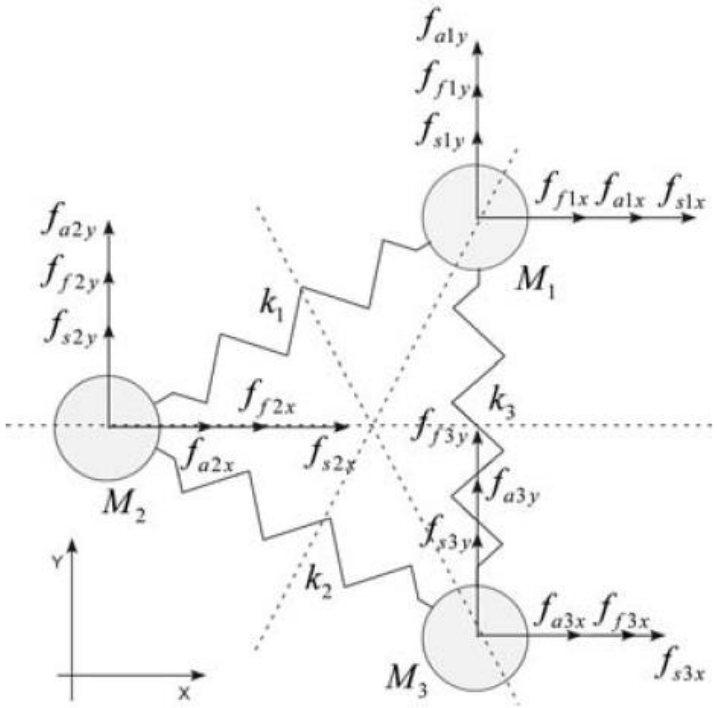


Figura 2.18 Sistema masa – resorte de la plataforma

Resolviendo las ecuaciones de equilibrio estático de la plataforma se encuentran las reacciones verticales en los puntos de contacto A, B y C y consecuentemente las masas distribuidas M_1, M_2, M_3 son calculadas mediante *(2.14)*.

$$\begin{aligned} M_1 &= (2Hl(M + m_d + m_e) - 3lMc_x - 6H(Mc_y - (m_d - m_e)d))/(6Hl) \\ M_2 &= \frac{1}{3}(m_d + m_e + M + \frac{3Mc_x}{H} \\ M_3 &= (2Hl(M + m_d + m_e) - 3lMc_x + 6H(Mc_y - (m_d - m_e)d))/(6Hl) \end{aligned} \tag{2.14}$$

Donde l, H, d son parámetros dimensionales mostrados en la figura 4 y (c_x, c_y) son las coordenadas del centro de más con respecto al marco de referencia B. Las constantes k_1, k_2, k_3 representan las respectivas rigideces de los resortes y m_d, m_e son las cargas excéntricas de los motores D y E respectivamente. Las respectivas deformaciones son pequeñas por lo tanto el cambio en el ángulo en los resortes se considera despreciable. Fuerzas f_{ai}, f_{fi}, f_{si}, con i = {1, 2,3}, son la actuación, fricción y fuerza de resorte ejercida a la masa M_i. Las ecuaciones dinámicas del sistema masa – resorte están dadas por (2.15) y (2.16):

$$M\ddot{x} = Ax + f_a + f_f, \tag{2.15}$$

$$x(0) = 0 \ , \ \dot{x}(0) = 0 \tag{2.16}$$

Donde M es la matriz de masa, A es la matriz que contiene las constantes de los resortes, y $x = [x_1, x_2, x_3, y_1, y_2, y_3]^T$ represente el desplazamiento x – y de las 3 masas. Cuando las masas están en un estado estático, las incógnitas del sistema son las 6 fuerzas de fricción, las cuales están determinadas al resolver las 6 ecuaciones de equilibrio estático. Cuando alguna o todas las masas están en movimiento, la magnitud de la correspondiente fuerza de fricción está determinada por el límite de la fricción de Coulomb, mientras la dirección de la fuerza de fricción está determinada por la velocidad de la correspondiente masa.

2.14.3.3 Dinámica del actuador

Los actuadores están modelados como un sistema compuesto por un motor de corriente continua con un magneto permanente y una carga desbalanceada. La entrada del actuador es el voltaje V_s. La dinámica del actuador se expresa mediante la ecuación (2.17).

$$\ddot{\theta} = -\frac{k_T^2}{RJ}\dot{\theta} - \frac{mgr}{J}\sin\theta - (\frac{c}{J}sign(\dot{\theta}) + \frac{b}{J}\dot{\theta}) + \frac{k_T}{RJ}V_s \tag{2.17}$$

Donde. k_T es la constante de torque del motor. R es la resistencia óhmica, J es la inercia de la carga excéntrica, y el término $(\frac{c}{j}sign(\dot{\theta}) + \frac{b}{J}\dot{\theta}$ es la fricción de Coulomb y la fricción viscosa.

3 Metodología

En líneas generales el trabajo se divide principalmente en 3 ramas, las cuales son: construcción tanto del Setup experimental como de las plataformas robóticas, software lo cual incluye simulación y control del movimiento de los robots, caracterización del comportamiento de los robots.

3.1 Construcción

Esta sección abarca todo lo relacionado con la construcción, se describen los requisitos que deben cumplirse y como se lleva a cabo.

3.1.1 Plataformas robóticas

El diseño de los robots está basado en el Cheavibot, el cual es un tipo de robot vibracional que se mueve debido al fenómeno de stick-slip. El robot Cheavibot se compone de: 2 motores vibracionales, 2 transistores, 2 fotorresistencias, un porta batería CR2032, una batería CR2032, una pcb para colocar los componentes electrónicos antes mencionados y finalmente una estructura que sirve de soporte fabricada en plástico PLA.

El diseño del robot debe cumplir los siguientes requisitos:

- Contar con 2 motores que funcionen de manera independiente, cada motor debe permitir girar en una dirección distinta, esto quiere decir que mientras el motor 1 hace girar al robot de forma horaria, el motor 2 hace girar al robot de forma anti horaria.
- Se debe contar con un circuito que permita encender y apagar los motores de manera independiente, este encendido/apagado se debe efectuar según la intensidad de luz a la cual es expuesto el robot.
- Además de controlar el encendido/apagado de los motores, se debe controlar la velocidad de giro de los motores.
- La estructura del robot debe tener geometría circular y ser lo más liviana posible.
- El robot debe tener 3 soportes que conectan la estructura del robot con el piso, estos soportes deben estar ubicados de manera que el centro del triángulo coincida con el centro del robot, la configuración de los soportes debe ser un triángulo equilátero.
- El montaje del robot debe ser lo más sencillo posible, también se debe permitir desmontar el robot.

La estructura del robot se diseña en software CAD, en una primera instancia el diseño comprendía fabricar el robot como una única pieza, sin embargo debido a que esto dificulta el montaje se decidió fabricar el robot en más de una pieza. Luego del diseño de las piezas en CAD, se procede a guardar las piezas en formato .STL (STereoLithography), que las piezas estén en este formato es un requisito para poder usar la impresora 3D, ya que el formato STL es uno de los más comunes en este tipo de máquinas. El siguiente paso es la configuración de la impresora, esta configuración se realiza mediante el software cura 3.0.4, el primer paso es la ubicación de las piezas en la bandeja de la impresora, las piezas se mueven en los ejes x e y, además las piezas se pueden rotar en los ejes x, y, z, la ubicación de las piezas se muestra en la Figura 3.1.

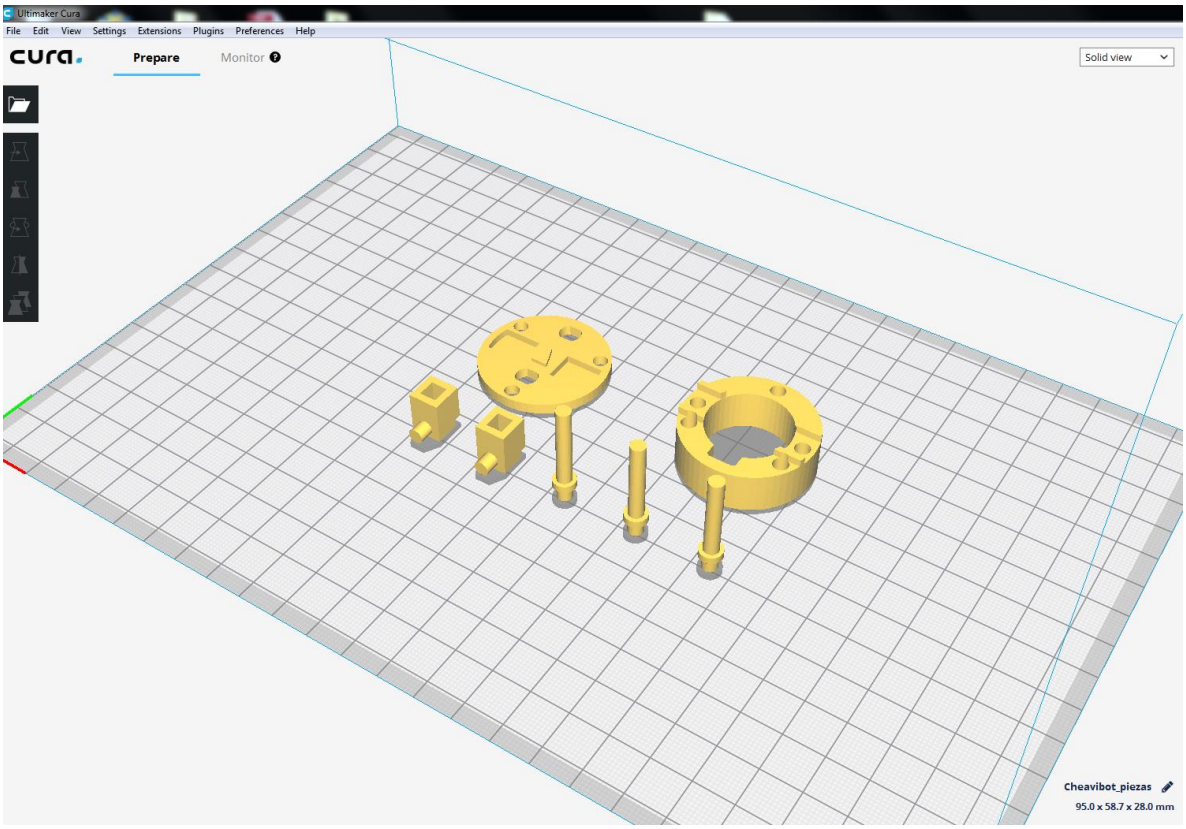


Figura 3.1 Posicionamiento de las piezas del Cheavibot en la bandeja de impresión, realizadas con el software cura 3.0.4.

Luego de ubicar las piezas en el software, se procede a realizar a ajustar los parámetros de la impresión, estos ajustes se pueden observar en las Figura 3.2 y Figura 3.3. Es necesario configurar el tipo de material con que se imprime, el cual puede ser PLA o ABS, para el Cheavibot se utiliza PLA. Quality indica el alto de la capa, dicho de otra forma, cuanto tiene que desplazarse el eje z entre capa y capa, este tiene un valor de 0.3mm. Shell indica la configuración de la cascara, se configura la cantidad de capas y el tamaño de la línea, que respectivamente son 3 y 0.8mm para el Cheavibot. Infill indica los parámetros para el llenado, se configura la densidad del llenado y el patrón a seguir, para el Cheavibot se

escoge una densidad de 20% y un patrón de llenado en grilla. Los ajustes anteriores se observan en la Figura 3.2. Material indica a que temperatura se debe calentar el extrusor para el material dado, el diámetro del filamento y cuanto porcentaje del filamento debe fluir por el extrusor, para el Cheavibot estos parámetros son 200°C, 1.75mm y 100%. Speed indica la velocidad a la que se debe mover el extrusor, hay 2 velocidades que se pueden modificar, la primera es la velocidad a la que se mueve el extrusor cuando imprime, mientras la segunda es la velocidad a la que se mueve el extrusor cuando viaja de un lugar a otro sin imprimir, ambas velocidades se configuran en 40mm/s. Finalmente el último parámetro tiene que ver con la adhesión a la bandeja de impresión, se imprime una estructura conocida como raft. La configuración anterior se puede observar en la Figura 3.2

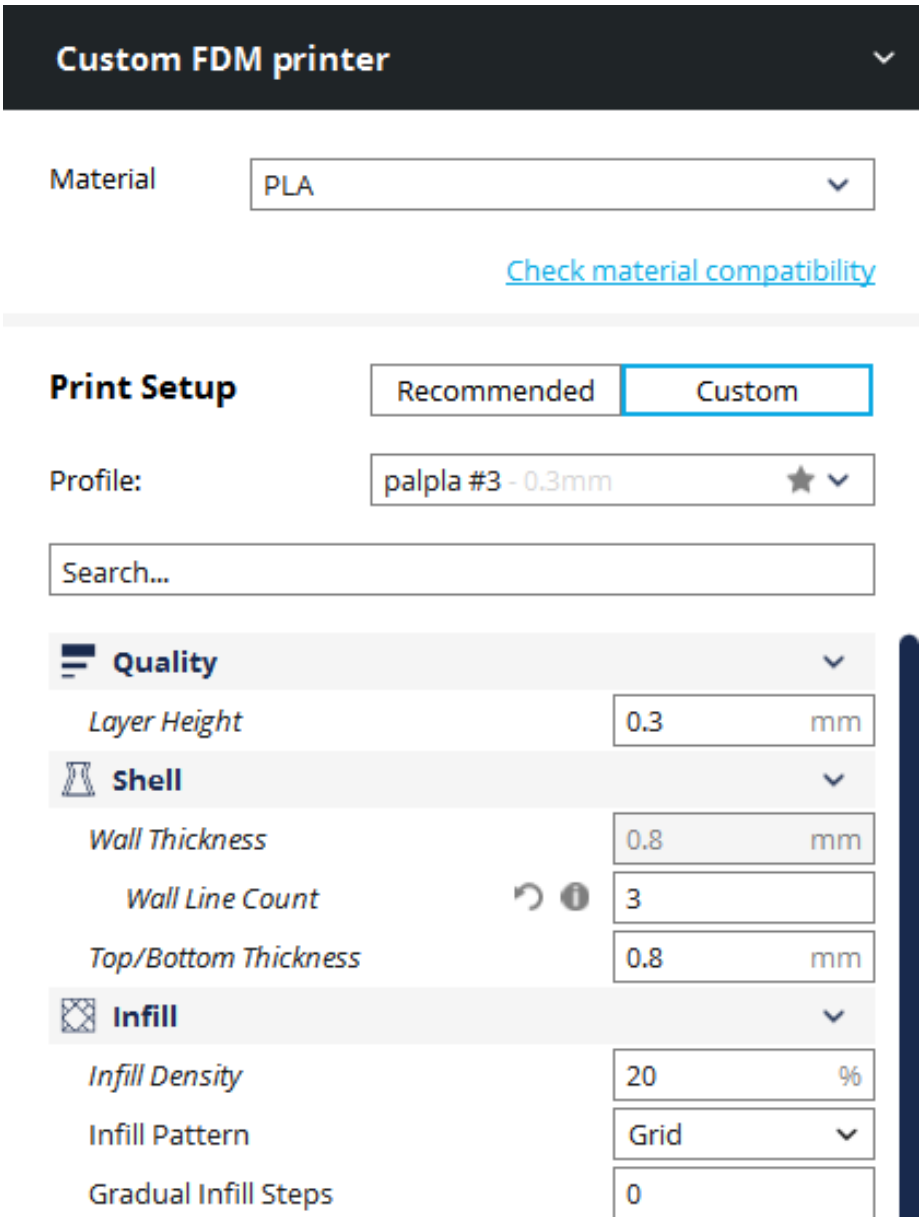


Figura 3.2 configuración de altura de capa, tamaño de cascará y llenado.

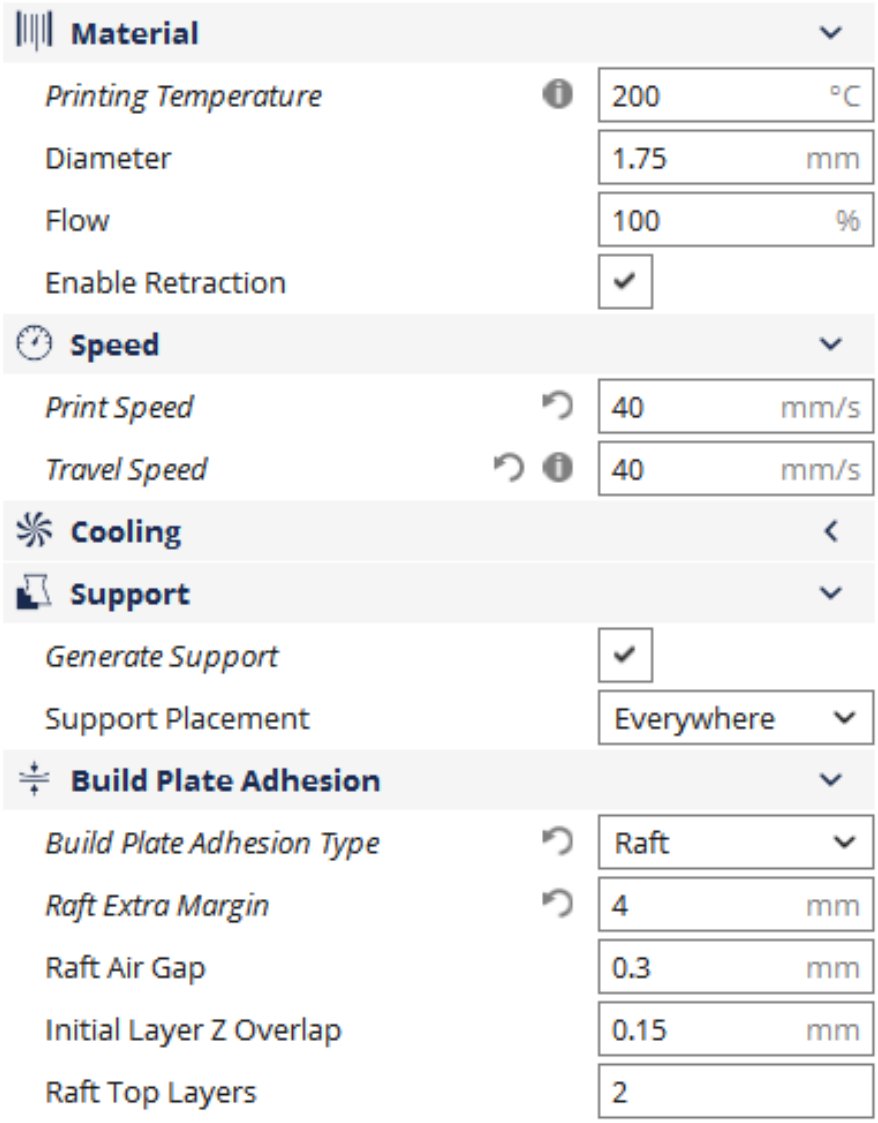


Figura 3.3 configuración de material, velocidad extrusor, soporte y raft.

Una vez que las piezas se imprimen se procede al ajuste de las piezas, para este ajuste se liman las piezas de forma tal que se logre un ajuste de presión en las piezas que van montadas entre sí.

3.1.2 Setup experimental

Para el Setup experimental son necesarios 3 elementos:

1. Cámara oscura
2. Cámaras optitrack
3. Proyector DLP

A continuación se describe porque son necesarios estos 3 elementos y como se instalan para formar el Setup experimental.

3.1.2.1 Cámara oscura

Debido a que los Cheavibots se controlan mediante la intensidad de luz incidente, es necesario proporcionar las condiciones lumínicas adecuadas para el correcto funcionamiento de los robots. El circuito de los robots solo está en estado apagado cuando hay poca luz, por esta razón no se puede trabajar con condiciones de luz ambiental, porque los robots estarían siempre encendidos.

La cámara oscura se construye con una estructura de acero, la cual puede variar la altura, sobre esta estructura se coloca lona negra a modo de cortina, la lona se coloca en las 4 caras verticales y en el techo, de modo de bloquear toda la luz proveniente del exterior. Todos los instrumentos para la medición y los robots se colocan dentro de esta cámara oscura.

3.1.2.2 Optitrack

Para conocer la posición de los robots se utiliza el sistema de posicionamiento optitrack, se utilizan las cámaras Flex 3 mostradas en la Figura 3.4, se colocan 4 cámaras en un soporte habilitado al interior de la cámara oscura. Una vez fijadas las cámaras se conectan al OptiHub que es un dispositivo que provee corriente a las cámaras para su funcionamiento además de permitir sincronizar dispositivos externos, con esto se puede tener varios OptiHub configurados como esclavos, cada OptiHub puede entregar corriente a 6 cámaras USB, una vista superior del OptiHub se muestra en la Figura 3.5. Desde el OptiHub se conecta mediante USB al computador.

Figura 3.4 cámaras optitrack modelo Flex 3

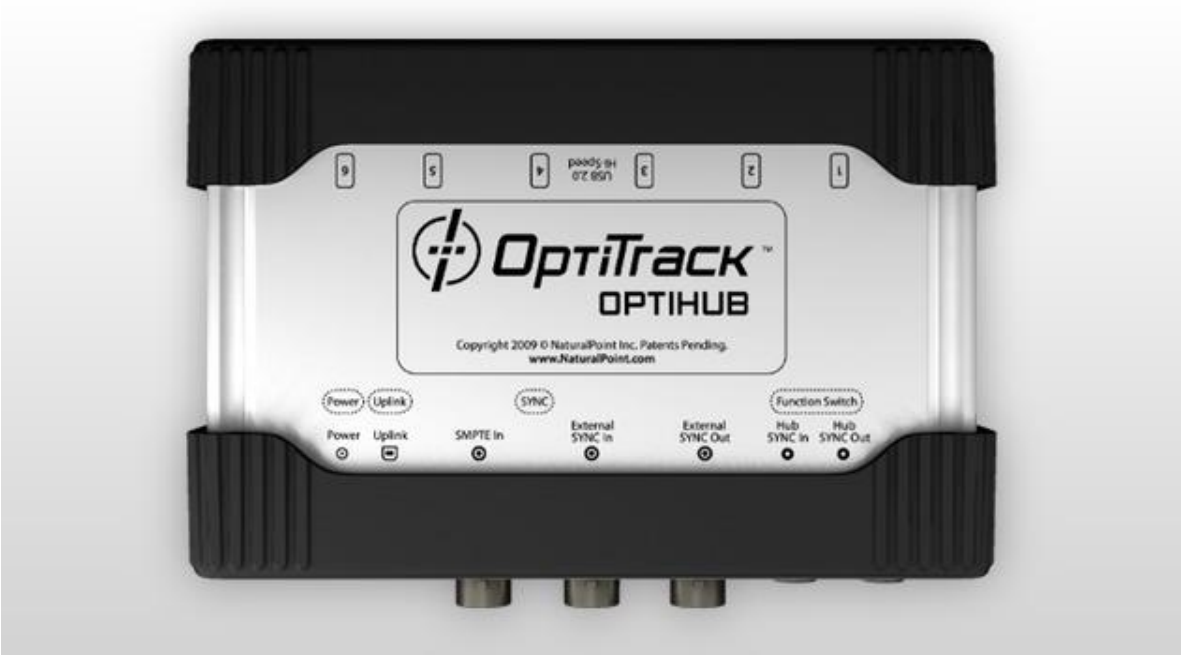


Figura 3.5 OptiHub vista superior

Una vez instalado todo el hardware de las cámaras y preparada el área de captura se procede a calibrar el software de captura de movimiento –Motive 1.7.0 –, para esto se utiliza una barra con 3 marcadores infrarrojos colocados en una misma línea, luego esta barra se mueve en el volumen de captura con movimientos de ocho, intentando abarcar

todos los espacios del volumen de captura, una vez el software adquiere suficientes puntos se procede a calcular la posición. Para el presente trabajo se tomaron un mínimo de 5000 puntos por cámara. El siguiente paso es fijar el centro de coordenadas, para esto se usa una escuadra que tiene adosados 3 marcadores, uno de estos marcadores marca el origen, mientras los otros marcadores indican la dirección de los ejes. Los pasos de la calibración se muestran en la Figura 3.6 y en la Figura 3.7.

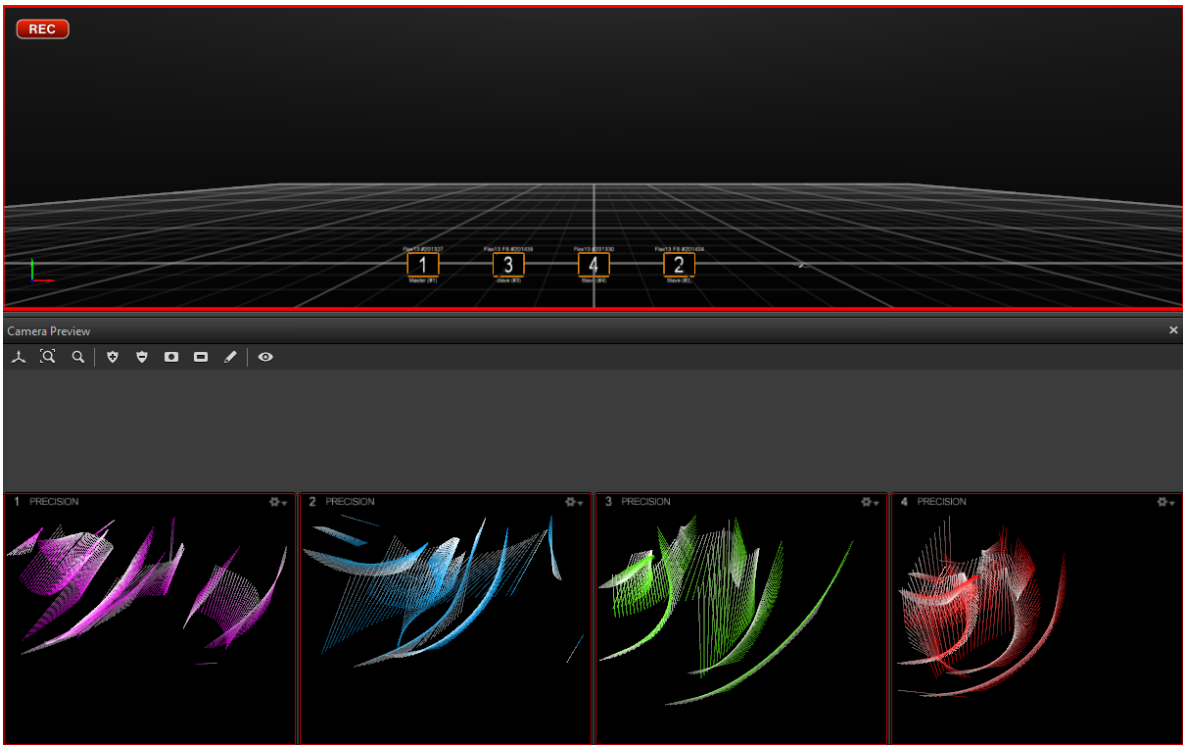


Figura 3.6 Recolección de puntos para la calibración de las cámaras en el software Motive

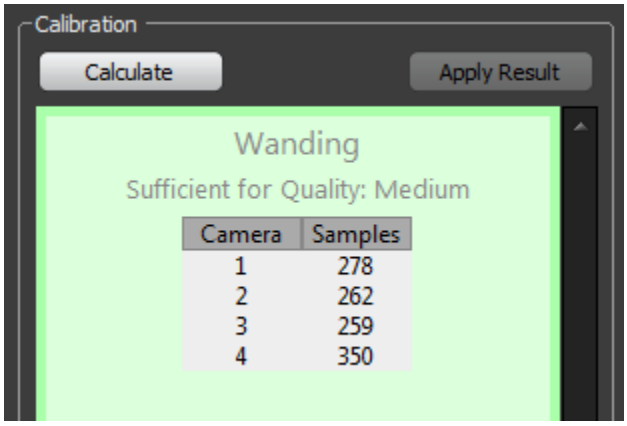


Figura 3.7 Cantidad de puntos de la Figura 3.6

3.1.2.3 Proyector DLP

Debido a que los robots son controlados con señales lumínicas es necesario un dispositivo que pueda mandar estas señales. El dispositivo empleado en el presente trabajo, corresponde a un proyector DLP de la marca ASUS P2M, este proyector se muestra en la Figura 3.8. El proyector se coloca a una distancia de aproximadamente 1.5 metros sobre la superficie de prueba, el proyector se instala dentro de la cámara oscura.

Figura 3.8 proyecto dlp usado para mandar las señales lumínicas.

3.2 Simulación

De acuerdo a los objetivos del presente trabajo, se realizan simulaciones del sistema las cuales son comparadas con la bibliografía y en una etapa posterior con las observaciones experimentales. En este capítulo primero se describe a grandes rasgos cuales son las simulaciones efectuadas, luego se describen los elementos comunes, para finalmente describir en detalle cada simulación.

Hay que notar que el algoritmo de la simulación es la base para el algoritmo de control, por lo tanto la simulación de 50 elementos debe tomar un tiempo razonable, este tiempo está en el rango de 10-24fps (esto equivale a 42 - 100ms).

De ahora en adelante, cuando se refiere al robot i, se trata del robot actual, a este robot se aplican los cambios y algoritmos.

3.2.1 Simulaciones a realizar

Son tres las simulaciones que se realizan con el fin de observar el comportamiento de enjambre, estas son:

1. Self Assembly, estructura de racimo
2. Self Assembly, figuras ingresadas por el usuario
3. Movimiento colectivo

Self Assembly, estructura de racimo

Como se observa en [22], un grupo de robots exhibe un comportamiento de Self Assembly en forma de racimo al introducir la regla de que cada robot móvil debe buscar al robot fijo más cercano y conectarse al robot fijo.

Self Assembly, figuras ingresadas por el usuario

Como se observa en [5], el usuario ingresa figuras como mapa de bits con ciertas restricciones al programa [23]. Los robots parten desde un estado empaquetado y se desplazan hasta formar la figura deseada

Movimiento colectivo

Como se observa en [24] un grupo de individuos formando parte de un enjambre presenta cierto movimiento coordinado, este movimiento se simula mediante la introducción de ciertas reglas [25], las cuales son:

1. Repulsión
2. Alineación
3. cohesión

3.2.2 Elementos comunes

Las 3 simulaciones antes presentadas presentan elementos en común, los cuales son: las simplificaciones, la interacción local, la geometría de los robots y la generación de los elementos.

3.2.2.1 Simplificaciones

Se supone que los elementos de la simulación son ideales. Esto quiere decir que no hay ruido o errores en las variables del sistema, por ejemplo: cada robot conoce con exactitud cuál es su posición, velocidad y ángulo con respecto al sistema global de coordenadas; para una velocidad de movimiento dada, se tiene que el robot no cambiara esta velocidad. Otra simplificación importante es: los robots pueden acceder a todas las variables que necesitan para realizar el movimiento, esto quiere decir que se supone que existe un sensor que puede medir dicha variable a bordo del robot.

3.2.2.2 Interacción local

Una característica de los enjambres robóticos es la interacción local, esto quiere decir que los robots solo pueden interactuar con los robots que estén cerca. Para simular este comportamiento se tiene que resolver el problema de encontrar los vecinos cercanos con radio fijo al robot consultado. Como se vio en 2.11 este corresponde a un problema de todos los pares cercanos en un conjunto dinámico de puntos.

Para la resolución de este problema se descartan los k-d tree, debido a que el tiempo de construcción de esta estructura de datos es demasiado para el caso de 50 elementos, el tiempo es incluso más grande que el mínimo necesario para funcionar como algoritmo de control. Hay que notar que para una cantidad muy elevada de elementos resolver el problema de vecinos cercanos con radio fijo usando k-d tree, necesita un tiempo menor en comparación a los otros métodos.

El método escogido para responder al problema de encontrar a los vecinos cercanos con radio fijo, es el método de fuerza bruta, se escoge este método porque es el más sencillo de implementar, además para la cantidad de elementos en donde el tiempo es crítico (50 elementos), el tiempo de este método solo ocupa aproximadamente un 10% del tiempo disponible para realizar todos los cálculos. Hay que notar que el problema a resolver en este trabajo implica redundancia, ya que es necesario contar 2 veces cada par, una vez para el par (x, y) y otra para el par (y, x).

La implementación de este método consiste en recorrer todos los elementos presentes en el sistema denotados por n, luego cada elemento xi se compara con los restantes elementos x0, x1,..xi-1, xi+1,…xn-1, xn, la comparación a realizar es la distancia entre elementos, si la distancia es menor al radio de la búsqueda, se guarda el par como vecinos.

En seudocódigo la implementación del método de fuerza bruta es la siguiente:

Algoritmo 3.1 Encontrar vecinos cercanos con un radio fijo

Requiere: calcular distancias entre elementos i, j.

Entrada: posiciones de los elementos i, j.

Salida: lista con pares de vecinos cercanos i, j y distancia.

```
1: for i in range (0, n):
2:     for j in range (0, n):
3:         distancia= sqrt((x_i - x_j)^2 + (y_i - y_j)^2)
4:         if distancia < r:
5:             guardar par (i, j)
6:         else:
7:             el par (i, j) no se guarda
```

La primera línea recorre los elementos del sistema de 0 a n, el índice de este elemento es i. La segunda línea recorre los elementos del sistema de 0 a n, el índice de este elemento es j, j es distinto a i. La tercera línea calcula la distancia euclidiana del elemento j al elemento i. De la cuarta a la séptima línea están las instrucciones lógicas, cuando la distancia entre el elemento j y el i es menor o igual al radio de búsqueda, implica que el par (i, j) son vecinos, en este caso se guarda el par dentro de una lista. En caso contrario, la distancia entre el par (i, j) es mayor al radio de búsqueda, este par no es vecino y no es guardado. Adicionalmente con el par (i, j) que se sabe son vecinos, se almacena la distancia entre ambos elementos, así la información que se guarda es: los elementos i, j y la distancia entre ambos (i, j, distancia (i, j)).

El siguiente paso es reordenar la lista que contiene todos los pares de vecinos cercanos con la información de distancia, en una forma adecuada para desarrollar los algoritmos posteriores. Se requieren 2 conjuntos de datos, el primero es: por cada elemento es necesario conocer todos los vecinos cercanos, el segundo es: por cada elemento es necesario conocer la distancia a los vecinos cercanos. Para realizar esto se usó el método de fuerza bruta, este método tiene $O(n^2*k)$, siendo k el número de vecinos a un elemento, k se puede considerar como la densidad de vecinos a un elemento. El método para contar todos los pares de vecinos cercanos con radio fijo tiene $O(n^2)$. Este método se escoge porque es el más sencillo de implementar.

En seudocódigo la implementación para ordenar los pares de vecinos cercanos es:

Algoritmo 3.2 Ordenar lista de vecinos cercanos

Requiere: lista de pares de vecinos cercanos con información de la distancia entre los elementos, se obtiene del **Algoritmo 3.2**.

Entrada: pares de vecinos cercanos.

Salida: lista 1contiene los vecinos cercanos de un elemento i, lista 2 contiene las distancias de los vecinos cercanos de i.

```
1: for i in range (0, n):
2:     for j in range (0, pares vecinos):
3:         if j es vecino de i:
4:             guardar índice j, guardar distancia entre i, j
5:     guardar índice i + conjunto de índices j que son vecinos
6:     guardar índice i + conjunto de distancias entre i, j
```

La primera línea recorre todos los elementos del sistema de 0 a n, el índice de este elemento es i. La segunda línea recorre los pares de vecinos, el total de pares es n*k, el índice para recorrer los pares de vecinos es j. En la tercera línea se comprueba que el elemento j es vecino del elemento i. En la línea 4, si (i, j) son vecinos, se almacena en una lista el índice j, en otra lista se almacena la distancia entre (i, j), en caso de no ser vecinos, se pasa al siguiente elemento j. La línea 5 vincula el índice i con todos los elementos j que son vecinos y se crea una lista que contiene a los n elementos del sistema, con sus correspondientes vecinos. La línea 6 es análoga a la línea 5, el cambio que se hace, es que en lugar de guardar los índices de los elementos que son vecinos a i, se guardan las distancias de estos elementos a i.

3.2.2.3 Geometría de los robots.

Hay 2 modelos que se usan para simular los robots, el modelo del Cheavibot que implica utilizar marcadores para obtener la posición en el optitrack, luego con estos marcadores se reconstruyen las otras características del robot. El segundo modelo consiste en una simplificación del Cheavibot, en donde solo se considera que el robot es un circulo de radio r.

Geometría de los Cheavibots

Hay 2 configuraciones geométricas en los Cheavibots que son importantes, la primera corresponde a la geometría de la base y actuadores y la segunda corresponde a la distribución de los marcadores.

Geometría de la base

La base cuenta con 3 apoyos cilíndricos, estos apoyos se ubican según un triángulo equilátero, el cual está circunscrito en un círculo el cual tiene un radio de (radio exterior + radio interior)/2, se toma como referencia una de los apoyos, luego se traza una línea que va desde el centro del soporte hasta el apoyo de referencia, esta línea divide la base en 2 partes, las cuales están en simetría. Los actuadores están alineados con el diámetro en la misma línea de acción y se colocan de manera opuesta entre sí, además la línea de acción de los actuadores es perpendicular a la línea de simetría que se obtiene con el apoyo de referencia. Las relaciones geométricas pueden observarse en la Figura 3.9.

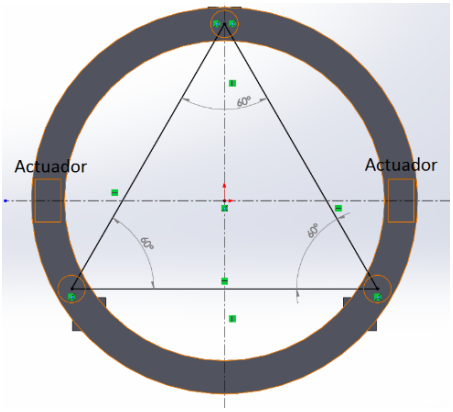


Figura 3.9 Relaciones geométricas de la base del Cheavibot

Geometría de los marcadores

Los marcadores se colocan en el círculo de la base, en una configuración de triangulo isósceles, se ha encontrado dos configuraciones que satisfacen los requisitos de los marcadores, una en que los ángulos iguales son 65° y otra en que los ángulos iguales son 55°. La posición de estos marcadores es la que se obtiene de las cámaras optitrack y corresponden a los datos de entrada se muestran en la Figura 3.10

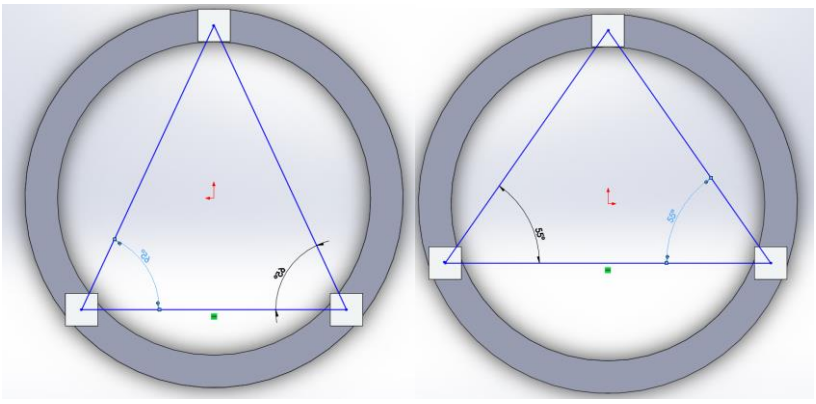


Figura 3.10 Configuración marcadores 65° y 55°

Reconstrucción geometría a partir de los marcadores

Los marcadores forman un triángulo isósceles circunscrito a un círculo de radio $(R_{ext} + R_{int})/2$, donde R_{ext} corresponde al radio externo de la base, mientras R_{int} corresponde al radio interior de la base según la figura. Conociendo las coordenadas de los vértices que forman el triángulo isósceles se puede calcular el centro del círculo circunscrito. Las fórmulas para obtener el circuncentro y el radio del círculo circunscrito están dadas por (3.1), (3.2) y (3.3).

$$U_x = \frac{1}{D}\left[\left(A_x^2 + A_y^2\right)\left(B_y - C_y\right) + \left(B_x^2 + B_y^2\right)\left(C_y - A_y\right) + \left(C_x^2 + C_y^2\right)\left(A_y - B_y\right)\right] \quad (3.1)$$

$$U_y = \frac{1}{D}\left[\left(A_x^2 + A_y^2\right)\left(C_x - B_x\right) + \left(B_x^2 + B_y^2\right)\left(A_x - C_x\right) + \left(C_x^2 + C_y^2\right)\left(B_x - A_x\right)\right] \quad (3.2)$$

$$D = 2\left[A_x\left(B_y - C_y\right) + B_x\left(C_y - A_y\right) + C_x\left(A_y - B_y\right)\right] \quad (3.3)$$

Donde las coordenadas del centro son U= (U_x, Uy), los vértices de los triángulos son A, B, C, los cuales tienen coordenadas [(A_x, Ay), (B_x, By), (C_x, Cy)].

Sin pérdida de generalidad lo anterior se puede simplificar al trasladar el vértice A al origen del plano cartesiano. Las traslaciones quedan de la siguiente manera (3.4), (3.5) y (3.6):

$$A' = A - A = \left(A'_x\,, A'_y\right) = (0,0) \quad (3.4)$$

$$B' = B - A = (B_x - A_x, B_y - A_y) \quad (3.5)$$

$$C' = C - A = (C_x - A_x, C_y - A_y) \quad (3.6)$$

Luego las coordenadas del circuncentro son U´= (U´$_x$, U´$_y$), estas coordenadas se aprecian en (3.7), (3.8) y (3.9):

$$U'_x = \frac{1}{D'}\left[C'_y\left(B'^2_x + B'^2_y\right) - B'_y\left(C'^2_x + C'^2_y\right)\right] \quad (3.7)$$

$$U'_y = \frac{1}{D'}\left[B'_x\left(C'^2_x + C'^2_y\right) - C'_x\left(B'^2_x + B'^2_y\right)\right] \quad (3.8)$$

$$D' = 2\left(B'_x C'_y - B'_y C'_x\right) \quad (3.9)$$

El valor del circunradio puede calcularse como

(3.10):

$$r = \|U'\| = \sqrt{U'^2_x + U'^2_y} \quad (3.10)$$

Las coordenadas del circuncentro pueden calcularse como (3.11):

$$U = U' + A \tag{3.11}$$

El siguiente paso es determinar los ángulos que corresponden a cada vértice del triángulo, para esto se ocupa la ecuación (3.12).

$$\theta = \cos^{-1}\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}\right) = \cos^{-1}\left(\frac{u_1v_1 + u_2v_2}{\sqrt{u_1^2+u_2^2}\sqrt{v_1^2+v_2^2}}\right) \tag{3.12}$$

Hay que notar que la fórmula anterior entrega el ángulo que hay entre dos vectores, en un rango de 0° a 180°, si el valor del ángulo entre 2 vectores es mayor a 180°, la fórmula anterior entrega el ángulo en el sentido contrario, por ejemplo si el valor del ángulo es 200° en sentido horario, la fórmula entrega un valor de 160° en sentido anti-horario, para solucionar el problema anterior y que los ángulos se presenten solo en sentido horario es necesario ocupar el determinante de los vectores (3.13).

$$Det(\vec{u},\vec{v}) = u_1v_2 - u_2v_1 \tag{3.13}$$

Luego si:

$Det(\vec{u},\vec{v}) < 0$ *el angulo esta en sentido horario*

$Det(\vec{u},\vec{v}) > 0$ *el angulo esta en sentido anti − horario*

Si el ángulo está en sentido anti-horario se transforma a sentido horario según (3.14):

$$\alpha' = 360° - \alpha \tag{3.14}$$

Donde α' es el ángulo en sentido horario y α es el ángulo en sentido anti-horario.

Una vez se conocen los ángulos asociados a cada vértice, se comparan los valores de estos ángulos, como se trata de un triángulo isósceles, 2 ángulos son iguales y 1 diferente. En el caso de la configuración de los marcadores en 65°, el ángulo diferente es el menor de entre los 3 ángulos y en el caso de la configuración de los marcadores en 55°, el ángulo diferente es el mayor entre los 3 ángulos. El ángulo diferente indica el vértice que pasa por el eje de simetría del Cheavibot, el cual sirve para orientar al robot en el espacio.

De esta forma se transforma el problema de rastrear los vértices de un triángulo, en un problema en donde se indica el centro del robot más la orientación que indica el avance frontal del mismo.

Posición de las fotorresistencias (LDR).

Debido a que los LDR son fundamentales al momento de controlar los motores, es necesario conocer su ubicación. Para esto el primer paso consiste en ubicar físicamente los LDR en un lugar determinado del bot, este lugar es la misma línea de acción donde se encuentran los motores, situados a una distancia d del centro del bot, esta distancia corresponde a $\frac{5}{7}R$, donde R es el radio del bot. Luego para poder determinar donde están ubicados los LDR se necesita conocer el centro del bot, el vector de orientación que indica el avance y la distancia desde el centro.

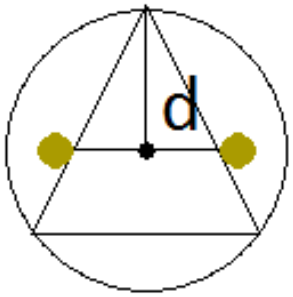


Figura 3.11 Posición de las fotorresistencias en el robot. Se muestra el eje de simetría y la posición de las LDR mediante círculos

3.2.2.4 Generación de elementos

Los elementos están posicionados en el área de trabajo de manera pseudoaleatoria, una restricción importante a tener en cuenta es: en la generación de los elementos, no debe existir superposición entre ningún par de elementos. Para solucionar este problema se parte de una grilla regular, la cual puede ser cuadrada o hexagonal, luego cada elemento se coloca en el vértice de esta grilla. Luego cada punto se desplaza con un valor aleatorio en las coordenadas x e y. Este valor aleatorio se asigna con una distribución uniforme entre un valor h y 0, el valor h debe ser tal que evite la superposición de los elementos. El valor máximo de h que evita la superposición se calcula como (3.15):

$$valor\ máximo = {}^{L}/_{2} - R \tag{3.15}$$

Donde L es la separación entre cada vértice de la grilla y R es el radio del elemento. L es igual en x e y para la grilla cuadrada, para la grilla hexagonal hay un L para x y un L distinto para y. cuando h se acerca más al valor h, los elementos parecen estar distribuidos de manera más aleatoria, mientras que si h se acerca más a cero, los elementos se distribuyen más según el orden original de la grilla.

Dado que al realizar una grilla regular con los parámetros L y R fijos conduce a grillas con un total de elementos que es fijo, se denominara al número de elementos de la grilla como n,

dado un número de elementos deseados denotado por m. Si m>n, el número de elementos de la grilla no alcanza a cubrir el número de elementos deseados, en este caso hay que aumentar el número de elementos de la grilla, esto solo se puede lograr al modificar la distancia L que separa a cada vértice de la grilla. Por el contrario si m<n, hay más elementos en la grilla que elementos deseados, en este caso se procede a eliminar elementos de n, hasta que el número de elementos es igual a m.

3.3 Detalle de la simulación

A continuación se detallan las reglas y algoritmos empleados en cada simulación, es importante notar que la simulación de movimiento colectivo se realiza con un tipo de objeto distinto al empleado en las simulaciones de Self Assembly.

3.3.1 Self Assembly, estructura de racimo

En la presente simulación, el objetivo de los robots es ir hacia donde están los objetos inmóviles y "agarrarse al objeto inmóvil", al suceder este "agarre" el robot se vuelve un objeto inmóvil y otros robots pueden "agarrarse" al robot que se ha vuelto un objeto inmóvil. Debido a este comportamiento de los robots se forma una estructura tipo racimo, en donde cada nuevo robot que se une a la estructura hace crecer las ramas de esta estructura. Se menciona "agarrar", porque en [22] los robots tienen un sistema de pinza que sirve para agarrarse a otros robots, en este caso la simulación emula este comportamiento solo deteniendo al robot que se "agarra".

El primer punto es describir los parámetros de los elementos comunes, la generación de elementos se realiza a través de una grilla hexagonal, en donde se eliminan aleatoriamente elementos hasta llegar al número de elementos deseados en el sistema. La geometría de los robots es del tipo Cheavibot. El área de interacción se considera como un área circular con un radio de 4 veces el diámetro del robot, este es el radio que se usa para resolver el problema de los vecinos cercanos con radio fijo.

El segundo punto es describir los elementos propios de esta simulación como: los tipos de estados de los robots, la asignación de estos estados y las reglas de movimiento de los robots.

Existen 4 estados de movimiento que se le pueden atribuir a cada robot, cada robot solo puede tener un estado de movimiento a la vez. Estos estados son:

1. Inmóvil

2. Móvil con vecino inmóvil
3. Móvil sin vecino inmóvil
4. Móvil con riesgo de colisión

Un robot en estado inmóvil no se mueve y permanece en estado inmóvil hasta el final de la simulación, un robot inmóvil sirve como referencia para que otros robots se puedan agarrar a este. Este estado se puede asignar al comienzo de la simulación, dejando inmóvil al robot, o cuando un robot se agarra a un elemento inmóvil.

Un robot móvil con vecino inmóvil, es un robot que se mueve y dentro del área de interacción ha detectado un robot inmóvil, este robot se moverá en dirección del robot inmóvil más cercano. Este estado se asigna cuando un robot es capaz de detectar un robot inmóvil dentro del área de interacción.

Un robot móvil sin vecino inmóvil, es un robot que se mueve y dentro del área de interacción no detecta algún robot inmóvil, este tipo de robot seguirá moviéndose en una trayectoria determinada, hasta encontrar un robot inmóvil dentro del área de interacción y cambiar de estado a "móvil con vecino inmóvil".

Un robot con riesgo de colisión, es un robot móvil, el cual puede tener vecinos inmóviles o no. Este estado se asigna cuando la distancia entre el robot i a cualquier robot j es menor a cierta distancia rc. La distancia rc, representa el radio de colisión y es una distancia que indica que hay alta probabilidad de que el robot i colisiones, este estado es importante, ya que sirve para activar el algoritmo de colisión.

La asignación de los estados en seudocódigo es:

Algoritmo 3.3 Asignación estados Self-Assembly estructura de racimo

Requiere: Calcular distancias entre vecinos i, j y comparar estados entre vecinos

Entrada: posiciones de los elementos i, j, estados elementos j

Salida: estado del robot i

```
1: Robots inmóvil = [xi, xj, xk,….]
2: while True:
3:     for i in range (0, n):
4:         if robot i == estado inmóvil:
5:             pass
6:         else:
7:             vecinos robot i
```

8: if robot i es vecino con robot inmóvil:

9: estado robot i = móvil con vecino inmóvil

10: if robot i no es vecino con robo inmóvil:

11: estado robot i = móvil con vecino inmóvil

12 if distancia robot i a robot j < rc:

13 estado robot i = móvil con riesgo de colisión

14 if distancia robot i a robot j (estado inmóvil) < ra:

15 estado robot i = inmóvil

16 condiciones para ejecutar el comando break

En la primera línea se definen los robots con el estado inmóvil. Hay que notar que es necesario al principio de la simulación indicar por lo menos a 1 robot con el estado inmóvil, de manera de servir como semilla para el crecimiento de la estructura. La línea 2 indica que este proceso se repite durante toda la duración de la simulación. En la línea 3 se recorren todos los elementos del sistema. La línea 4 y 5 dicen que si un robot tiene el estado inmóvil, permanece tal cual. De la línea 6 a la 13 se asigna el estado al robot dependiendo de los estados de los vecinos. Las líneas 14 y 15 agregan más elementos a la estructura, se compara la distancia entre el robot i y el robot más inmóvil más cercano, si la distancia es menor al radio de agarre, el robot i cambia a estado inmóvil y se agrega a la estructura de racimo. La línea 16 aplica la(s) condición(es) para ejecutar el comando break y terminar la simulación.

Reglas de movimiento.

Cuando se tiene un robot con el estado móvil y con vecino inmóvil. Se calcula la distancia entre el robot i y el robot inmóvil más cercano, en la simulación lo que se hace es buscar los elementos que son vecinos inmóviles y luego mediante la lista que tiene los datos de distancia, buscar la mínima distancia. El segundo paso es: calcular el ángulo entre el vector que va desde el centro del robot i hasta el centro del robot inmóvil más cercano con el ángulo de la dirección del robot i.

Describir las operaciones o reglas de cómo se define el movimiento.

Cuando se tiene un robot con el estado móvil y sin vecino inmóvil. Este sigue una trayectoria libre, siempre gira en sentido horario.

3.3.2 Self-Assembly, formación de figuras

Generación de figuras

Las figuras que se le entregan al programa deben ser hechas por el usuario, para simplificar los cálculos para detectar si un robot está dentro o fuera de la figura se escoge representar

cualquier figura mediante círculos, se usa un método de fuerza bruta para detectar si el robot está dentro de la figura. Se verifica cada círculo que compone la figura, si el robot está dentro de algún circulo significa que está dentro de la figura, por el contrario si no está dentro de ningún círculo, significa que esta fuera de la figura.

Suponiendo que la figura dada por el usuario tiene n círculos, para cada círculo la posición del centro es xi, yi y el radio es ri. Dados estos datos el algoritmo para detectar si un robot está dentro de la figura o no en seudocódigo es:

Algoritmo 3.4 Comprobar que un robot este dentro de la figura

Requiere: figura ingresada por el usuario en base a círculos

Entrada: datos del centro y radio de los círculos de la figura, posición del elemento a considerar

Salida: True o False, dependiendo si el robot está dentro de la figura o no.

1: figura ingresada por el usuario, consistente de n círculos

2: posición robot = x_r, y_r

3: radio robot = r_r

4: for i in range (0, n):

5: posición centro círculo i = x_i, y_i

6: radio círculo i = r_i

7: distancia robot – círculo i = $\sqrt{(x_i - x_r)^2 + (y_i - y_r)^2}$

8: distancia a comparar = r_i

9: if distancia robot – círculo i < distancia a comparar:

10: está dentro de la figura = True

11: else:

12: esta dentro de la figura = False

En el código anterior basta que para algún i se cumpla que ‘’está dentro de la figura’’ sea verdadero, para que el robot se considere dentro de la figura.

Tipos de estado

Existen 2 tipos de estado que definen a los robots, estos estados definen como interactúan los robots entre sí y que reglas de movimiento deben seguir. En un sentido constructivo estos estados representan información que los robots deben mandar a los vecinos y ser capaces de leer de los vecinos.

Estado tipo seed

El primero de estos estados es el tipo seed, el cual indica si un robot es una seed o no, que un robot sea una seed significa que este robot marca la referencia del sistema de coordenadas para los otros robots, solo 4 robots son definidos como tipo seed estos se colocan juntos como se muestra en la

Figura 3.12, formando una especie de cruz, el lado más largo de esta cruz marca el eje Y, mientras el lado más corto marca el eje X. Existen 3 opciones para el estado seed: la primera es que el robot es una seed, esto se representa gráficamente como un círculo de color verde al centro del robot, el robot no es una seed, esto se representa con un círculo de color rojo al centro del robot y la tercera opción es ninguna de las anteriores, esto se representa con un círculo de color azul, este estado indica que al robot aún no se le ha asignado un estado de seed y es necesario inicializarlo.

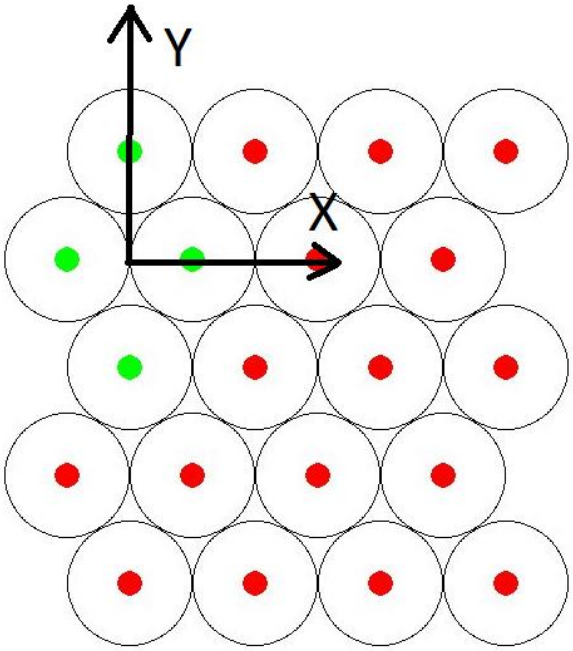


Figura 3.12 Empaquetamiento de los robots, los verdes representan el tipo seed, mientras los rojos representan a los robots que no son seed y están en espera de la orden para moverse.

Estado de funcionamiento

El estado de funcionamiento tiene relación con cómo se mueve el robot y por lo tanto que reglas debe seguir, en la simulación afecta que algoritmo se debe aplicar según cada estado,

mientras que constructivamente tiene relación con los datos que debe enviar/recibir. Existen 6 estados que indican el funcionamiento de los robots estos son: fijo, detenido, funcionando, moviéndose, figura, ninguno. Cada uno de estos estados se representa gráficamente con un anillo de color con un radio de 6/7 el radio del robot, en la Figura 3.13 se observan estos estados.

El estado fijo indica que el robot no se mueve, ya sea porque es una seed o porque se ubicó dentro de la figura, se representa gráficamente como un círculo de color verde. El estado detenido indica que el robot no se mueve, pero a diferencia del estado fijo el estado detenido implica una pausa, en la cual el robot espera instrucciones, los robots con el estado detenido están en el pack de robots, se representa gráficamente como un círculo de color rojo. El estado funcionando indica que el robot se mueve, este estado se activa cuando el robot se aleja del pack de robots, se representa con un circulo de color amarillo. El estado moviéndose indica que el robot se está moviendo por el perímetro del empaquetamiento de robots, este estado se representa gráficamente como un círculo de color cian. El estado figura indica que el robot se está moviendo dentro de la figura y se representa gráficamente como un círculo de color azul. Adicionalmente existe un sexto estado que es ninguno, este estado representa que el robot no tiene ningún estado de los anteriores y por lo tanto es necesario inicializarlo.

El primer de los estados de funcionamiento está ligado al estado seed. Si el robot es una seed el estado es fijo, por el contrario, si el estado de seed es falso el estado de funcionamiento es detenido. Luego cuando el robot entra en funcionamiento adquiere el estado funcionando, esto implica que primero debe alejarse del pack cierta distancia. Luego de alejarse el robot empieza a moverse por el perímetro del pack de robots, aquí adquiere el estado de moviéndose. Cuando el robot entra en la figura sigue moviéndose por el perímetro, pero adquiere el estado figura. Finalmente cuando el robot llega a su posición final dentro de la figura adquiere el estado fijo, por lo tanto no se mueve más de ese lugar.

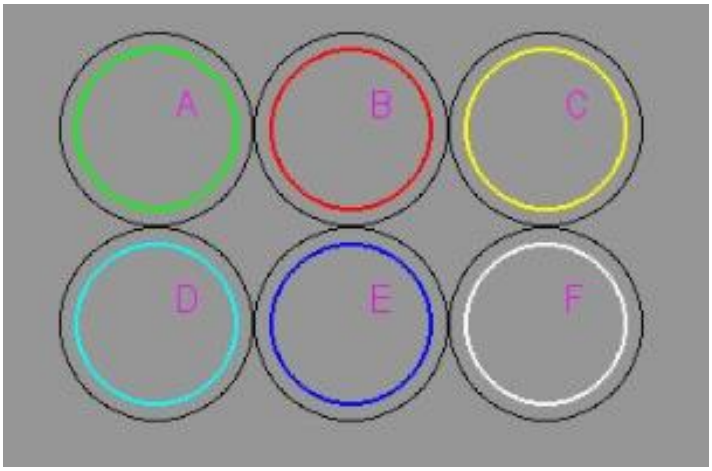


Figura 3.13 Representación gráfica de los estados de funcionamiento, se observa el anillo de color

correspondiente a cada estado. A) fijo, B) detenido, C) funcionando, D) moviéndose, E) figura, F) ninguno.

Gradiente de posición

Para poder tener la posición de cada robot en el espacio, se recurre a la ayuda de los vecinos. Para esto primero se parte con una referencia, la cual está constituida por los robots que son seed, el siguiente paso es asignarle a cada robot valor que indica que tan alejado se encuentra del origen marcado por la seed, a este valor se le denomina el gradiente de posición.

Para calcular el gradiente de posición primero se parte por indicar cuál es el robot que tiene gradiente cero, dicho robot solo se puede escoger de entre los robots que tienen el estado seed como verdadero. El siguiente paso es asignar un valor de gradiente a cada robot, para esto se consideran solo los vecinos con los cuales el robot está en contacto -problema de vecinos cercanos con radio fijo, radio = diámetro del robot-, el siguiente paso es comparar el valor del gradiente de los vecinos, se considera el mínimo valor y luego se suma 1, este valor corresponde al valor del gradiente del robot a considerar. Este proceso se repite hasta que todos los robots tienen asignado un valor del gradiente de posición. En la Figura 3.14 se observa los valores que toma el gradiente para cada robot, partiendo desde cero en el robot que es seed del extremo superior izquierdo, los vecinos a este robot tienen un valor de 1 y así sucesivamente. En seudocódigo este problema se resuelve como:

Algoritmo 3.5 Asignación gradiente de los robots

Requiere: lista de vecinos de cada elemento según **Algoritmo 3.2**.

Entrada: gradiente de los elementos semilla.

Salida: valor del gradiente del elemento i

```
1: valor gradiente robot k = 0, solo si valor seed robot k = verdadero
2: for i in range (0, n): # n = número de robots
3:     for j in range (0, nvecinos): # nvecinos = número de vecinos al robot i
4:         obtener valor gradiente robot j
5:     mg=min (valor gradiente vecinos)
6:     if valor gradiente robot i =0:
7:         pass
8:     else:
9:         valor gradiente robot i = mg+1
```

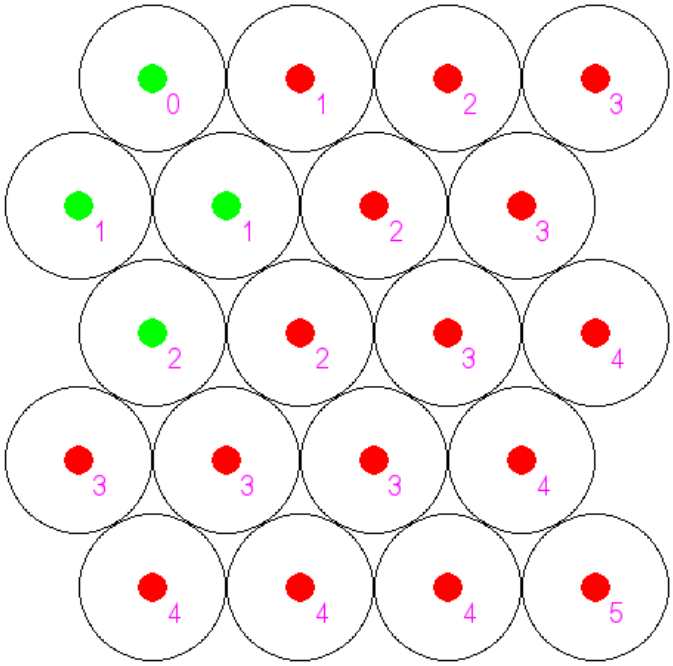


Figura 3.14 Representación gráfica del gradiente de posición. Los números representan la distancia al origen, el origen se representa con el número cero.

Hay que notar que al inicio de la simulación se deben inicializar todos los robots con un valor del gradiente por defecto, en el caso de las seed se escoge uno que tenga el valor cero, para todos los robots con excepción de aquel que tiene el valor cero, el gradiente toma un valor alto, este valor es igual al número de elementos del sistema. Al inicializar de esta manera el gradiente de todos los robots se asegura que cuando un robot tiene de vecino a un robot con el gradiente ya definido, este valor será el mínimo y el algoritmo funciona. El algoritmo anterior se repite hasta que a todos los robots se les ha asignado un valor del gradiente de posición.

Distanciamiento del empaquetamiento

Al inicio los robots se encuentran ordenados muy juntos entre sí, por esta razón el primer paso es que el robot se aleje de este empaquetamiento. Para determinar que robots se deben alejar, se debe conocer el valor del gradiente de posición, los robots que tienen el gradiente de posición más elevado, son los primeros en alejarse.

Movimiento por el perímetro

Los robots una vez se alejan del empaquetamiento de robots se mueven a través del perímetro, se mueven en sentido horario. Este movimiento por el perímetro está basado en el movimiento planetario, en donde un robot orbita alrededor de otro, a diferencia de los planetas los robots siguen orbitas circulares, con el robot vecino como centro de la órbita.

Para efectuar el movimiento planetario se calculan el vector normal y tangente, luego el robot se mueve según la dirección tangente, debido a que solo con este enfoque se tiene que el robot no puede mantener la órbita ya que tiende a ‘’caer’’ hacia el centro o alejarse en forma espiral según la velocidad del robot. Para solucionar el problema anterior, cuando el robot se desvía cierta distancia de la órbita en lugar de seguir moviéndose en dirección tangente, el robot realiza un giro en dirección a la órbita, de esta manera el robot se acerca a la órbita y puede mantener una distancia estable. Luego para efectuar el movimiento por el perímetro, el robot busca cual es el vecino más cercano y efectúa un movimiento planetario en torno a este vecino, cuando el vecino más cercano cambia, el robot efectúa el movimiento planetario con el nuevo vecino.

Movimiento dentro de la figura

Una vez el robot entra a la figura, sigue moviéndose por el perímetro, sin embargo ahora verifica 2 condiciones para detenerse. La primera condición indica que si el robot sale de la figura se debe detener, la segunda condición tiene relación con el valor del gradiente, si un robot en movimiento tiene un valor igual al gradiente de un robot detenido, el robot en movimiento se aproxima al robot detenido y se detiene cuando la distancia entre ambos robots es menor a la distancia indicada.

3.3.3 Movimiento colectivo

En el desarrollo de la simulación de movimiento colectivo hay que realizar 3 acciones, las cuales son: inicializar los robots, definir las áreas de interacción y aplicar las reglas de movimiento colectivo.

Inicialización de los Cheavibots

Al iniciar la simulación es necesario inicializar los robots con ciertos parámetros, los parámetros a inicializar son: la magnitud de la velocidad, la dirección de la velocidad, la posición inicial y el estado. Estos parámetros se inicializan con valores al azar, estos valores están en una distribución uniforme entre un valor mínimo y un valor máximo, el estado puede ser seguidor o líder, por defecto los robots tienen asignado el estado seguidor. El tamaño del robot (radio) es un parámetro fijo.

Nota: cuando a un robot se le asigna el estado líder, este robot deja de obedecer las reglas de movimiento colectivo y se desplaza según un itinerario propio.

Áreas de interacción

Como se ha mencionado anteriormente existen 3 reglas para controlar el movimiento de un grupo de elementos moviéndose al unísono, estas reglas son:

1. Repulsión
2. Alineamiento
3. Cohesión

Para aplicar estas reglas de movimiento se utilizan áreas de interacción específicas para cada regla, esto quiere decir que cada regla tiene asociada un área en donde solo esta regla se aplica al movimiento. Las áreas correspondientes a cada regla en base al radio del robot son:

- Repulsión: distancia $\leq 3r$
- Alineamiento: $3r <$ distancia $\leq 6r$
- Cohesión: $6r <$ distancia $\leq 8r$

Para determinar que regla utilizar se compara la distancia de los vecinos cercanos, la distancia con el robot más cercano es la que determina que regla utilizar, según lo anterior si hay algún robot en el área de repulsión se aplica la regla de repulsión. De no haber robots en el área de repulsión pero si en el área de alineamiento, se aplica la regla de alineamiento. Del mismo modo la regla de cohesión solo se aplica cuando los robots están en el área de cohesión y no están presentes en alguna otra área. El área de interacción de estas reglas corresponde con el área donde están los vecinos cercanos que es 8r. Cuando no hay ningún robot en el área de interacción este se sigue moviendo con la velocidad que tiene, ósea no modifica su velocidad.

Aplicación de las reglas

Para aplicar las reglas se siguen los siguientes algoritmos.

Regla de repulsión

Se promedia la posición de todos los vecinos para calcular el centro de posición de los vecinos, luego la posición de este centro se refleja con respecto a la normal entre este punto y el centro del robot a considerar, este nuevo punto es el objetivo de movimiento de la regla de repulsión, esto quiere decir que el robot se mueve para alcanzar este punto. Para calcular este punto primero se traslada el centro del robot a considerar al origen y a partir del origen se rota las coordenadas del promedio de los centros de los vecinos (x, y) con respecto a la normal, esto queda como (-x,-y), luego se vuelve a trasladar a la posición original.

Regla de alineamiento

Para la magnitud de velocidad se toma el promedio de la magnitud de velocidad de los vecinos, luego se compara este promedio con la velocidad del robot, si la velocidad del robot es menor esta se aumenta en un paso, si es mayor se disminuye en un paso y si es igual la velocidad se mantiene. Se utiliza pasos de velocidad, para simular organismos biológicos, los cuales cambian de velocidad de manera continua. Del mismo modo con la dirección se calcula el promedio angular de los vecinos, esto se hace con el arco seno, luego

se comparan los vectores que apuntan según la dirección promedio y la dirección del robot a considerar, dependiendo del determinante se puede saber cuál es la dirección de giro que el robot tiene que tener para disminuir el ángulo entre estos 2 vectores. Si el determinante es menor a cero gira anti horario, si el determinante es mayor a cero gira horario.

Regla de cohesión

Análogo a la regla de repulsión se calcula el centro de todos los vecinos, luego este punto se utiliza como objetivo de movimiento.

4 Resultados

4.1 Fabricación

4.1.1 Plataforma robótica

Los elementos estructurales del robot se fabrican mediante impresión 3D, como se describe en la sección 3.1.1, el robot se compone de 4 tipos de piezas, las cuales son: las patas, la base, el soporte del motor y la tapa. Estas piezas se diseñan en un software CAD y luego se exportan en formato STL a la impresora 3D. Las piezas que componen el robot se ensamblan como se muestra en la Figura 4.1, mientras que en la Figura 4.2 se muestra una vista explosionada de las piezas antes mencionadas, adicionalmente se agrega el porta batería a la vista. El ajuste de las piezas es un ajuste de presión.

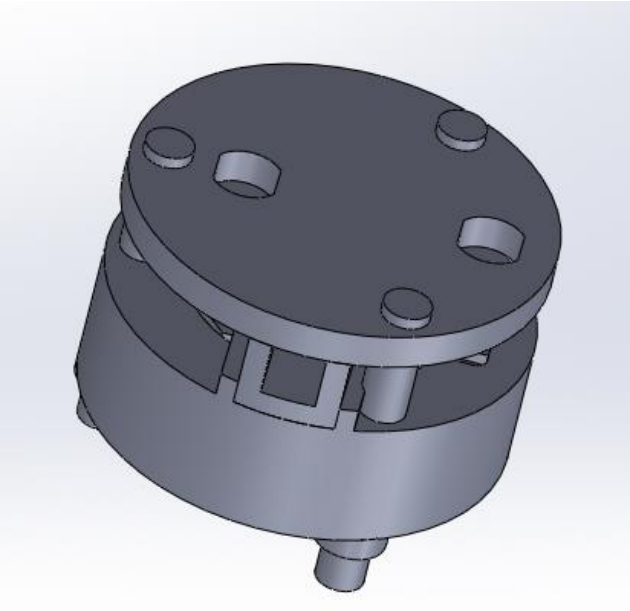


Figura 4.1 Robot ensamblado

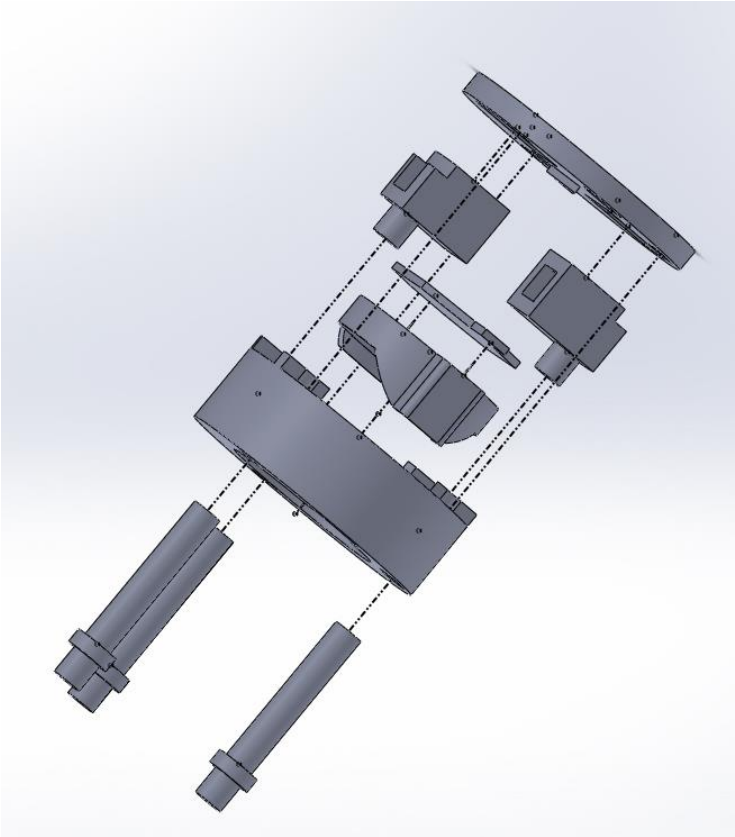


Figura 4.2 Vista explosionada del robot

A continuación se muestran las medidas más importantes de las piezas. Iniciando por las patas, la cantidad de patas por robot es de 3, estas son piezas cilíndricas, en donde el diámetro es de 4 mm, las patas tienen un aro a 4mm del piso para mantener la base sobre el piso, adicionalmente las patas tienen 0,5 mm sobre la tapa, de modo que sobresale parte de la pieza. Las medidas de las patas se observan en la Figura 4.3

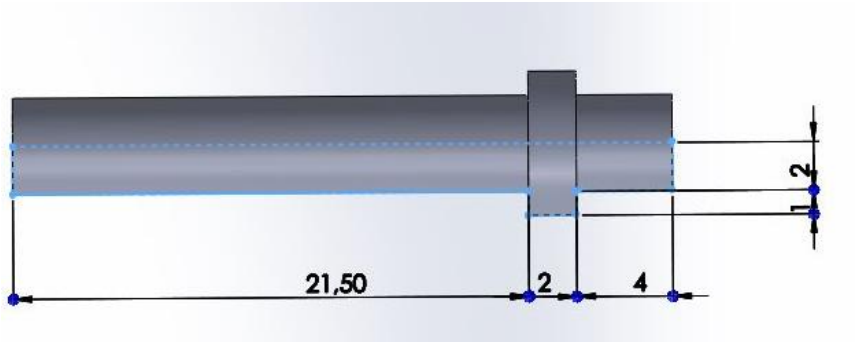


Figura 4.3 dimensiones de las patas

La siguiente pieza a analizar es la base, en la Figura 4.4 se pueden observar algunas dimensiones, estas dimensiones tienen relación con la ubicación de las patas. Las patas se ubican en un triángulo equilátero, este triángulo está circunscrito a un círculo de diámetro

29mm. El diámetro de la base es de 35mm esto se aprecia en la Figura 4.5, el robot además tiene un diámetro interno de 22,3 mm lo cual corresponde al diámetro del porta batería, se observa que la base tiene un sacado en el interior, este sacado también es para que encaje el porta batería.

Otro aspecto importante de la base es que la pieza que soporta el motor debe ir fijada a la base, para esto cuenta con una perforaciones según las dimensiones de la Figura 4.6 y unos apoyos que impiden que el soporte del motor gire las dimensiones se observan en la Figura 4.7.

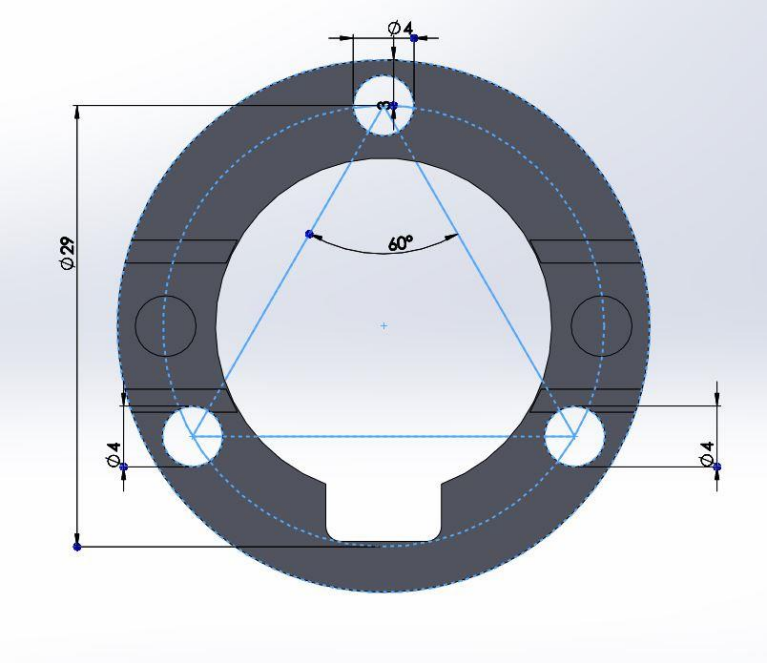


Figura 4.4 dimensiones de la base, estas dimensiones tienen relación con la posición de las patas

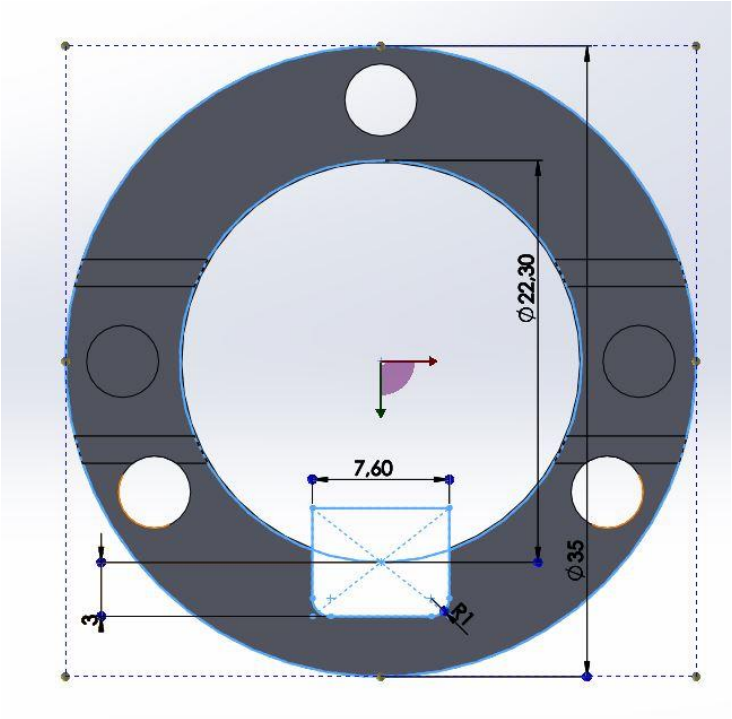


Figura 4.5 Dimensiones de la base, diámetro y relación con el porta batería

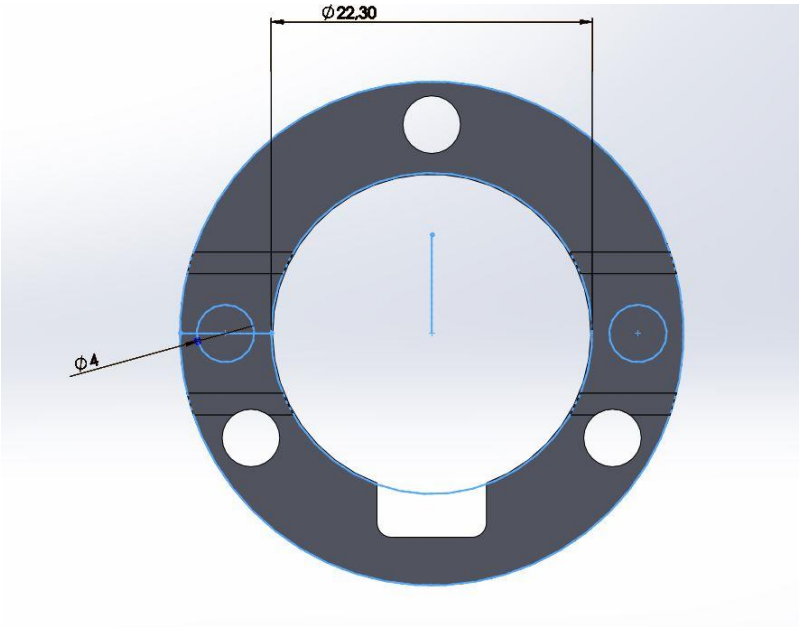


Figura 4.6 Dimensiones de la base, ubicación de las perforaciones para el soporte del motor

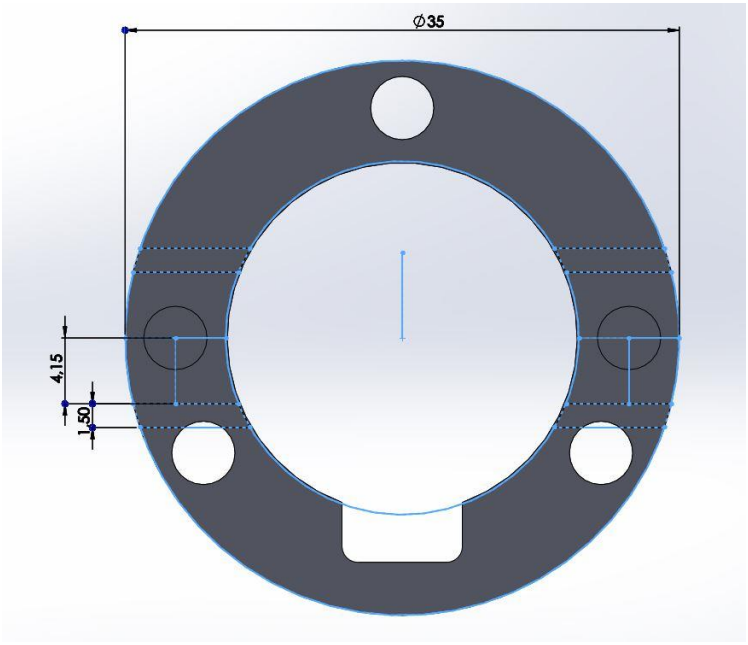


Figura 4.7 Dimensiones de la base, apoyo soporte motor

La siguiente pieza es el soporte del motor, las dimensiones se pueden observar en la Figura 4.8, las dimensiones internas corresponden al espacio donde se inserta el motor. Hay que notar que el ajuste es a presión y que debido a que no todos los motores tienen la misma medida es necesario ajustar individualmente cada pieza.

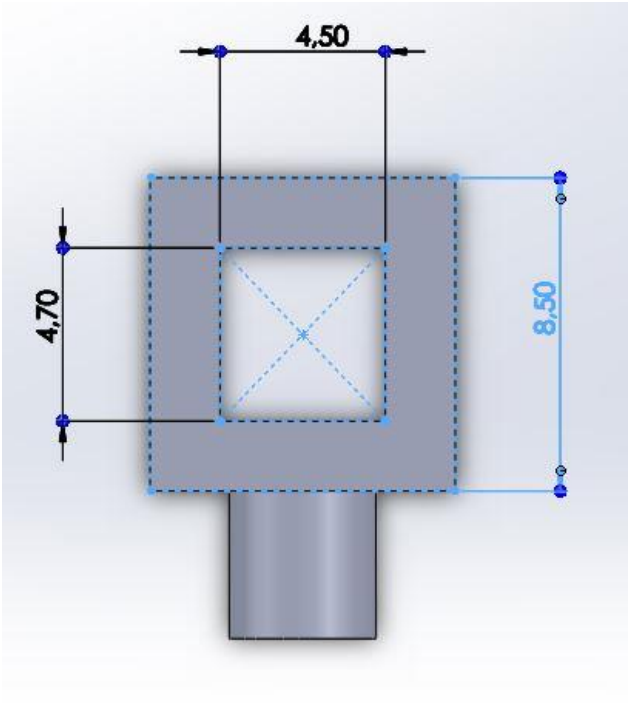


Figura 4.8 Dimensiones soporte motor, se muestra el cuadrado externo y el interno donde se coloca el motor.

La última pieza a considerar es la tapa, en la Figura 4.9 se puede observar que las medidas de las perforaciones por donde van las patas coinciden con la base, en la Figura 4.10 se muestran las perforaciones que corresponden a las LDR, se debe notar que las LDR no están ubicadas a la misma distancia del centro, adicionalmente se tiene que el tamaño del agujero es mayor que el de la LDR, esto es para dar cierta holgura al momento de ensamblar la tapa con la LDR.

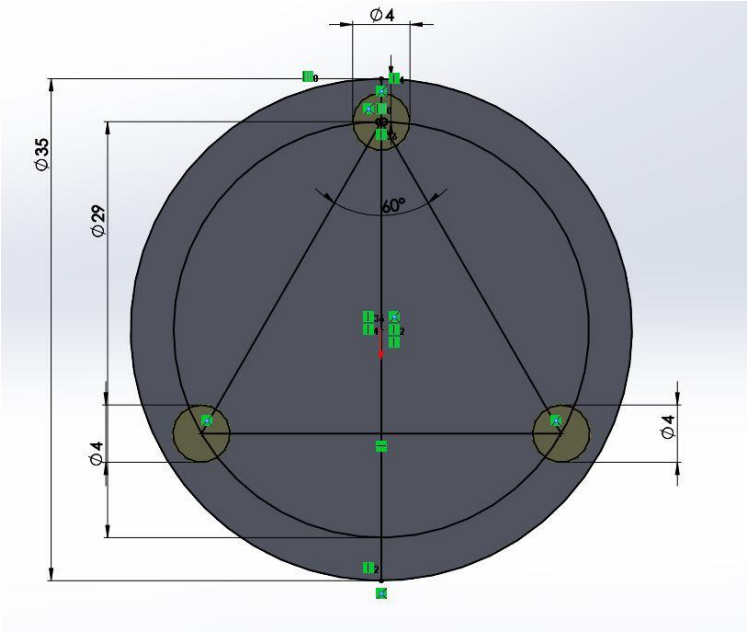


Figura 4.9 Dimensiones de la tapa, correspondencia con las dimensiones de la base

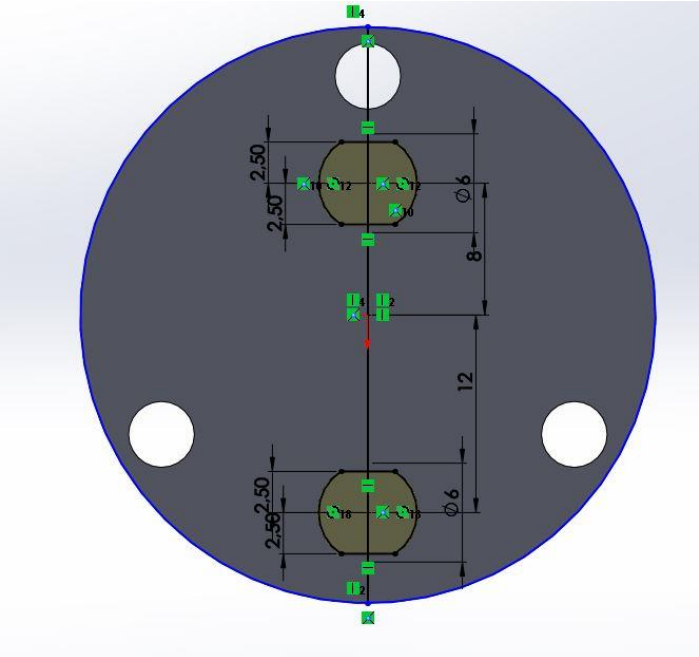


Figura 4.10 Dimensiones de la tapa, dimensiones para el ajuste de las LDR

Finalmente en la Figura 4.11 se muestran los 4 tipos de piezas en bruto, esto es después de haber sido impresas en 3D.

Figura 4.11 Piezas en bruto, se observan las piezas impresas en PLA.

4.1.2 Circuito

El circuito debe cumplir la función de encender los motores cuando hay luz y apagarlos cuando no hay luz, para esto se utiliza una resistencia sensible a la luz (LDR) y un transistor para servir como switch de ON/OFF.

Los componentes del circuito son:

- 2 fotorresistencias
- 2 transistores NPN 2222ª
- 1 batería CR2032
- 2 motores vibracionales

El esquemático del circuito se muestra en la Figura 4.12, hay que notar que el circuito base se repite tanto en el lado izquierdo como en el derecho.

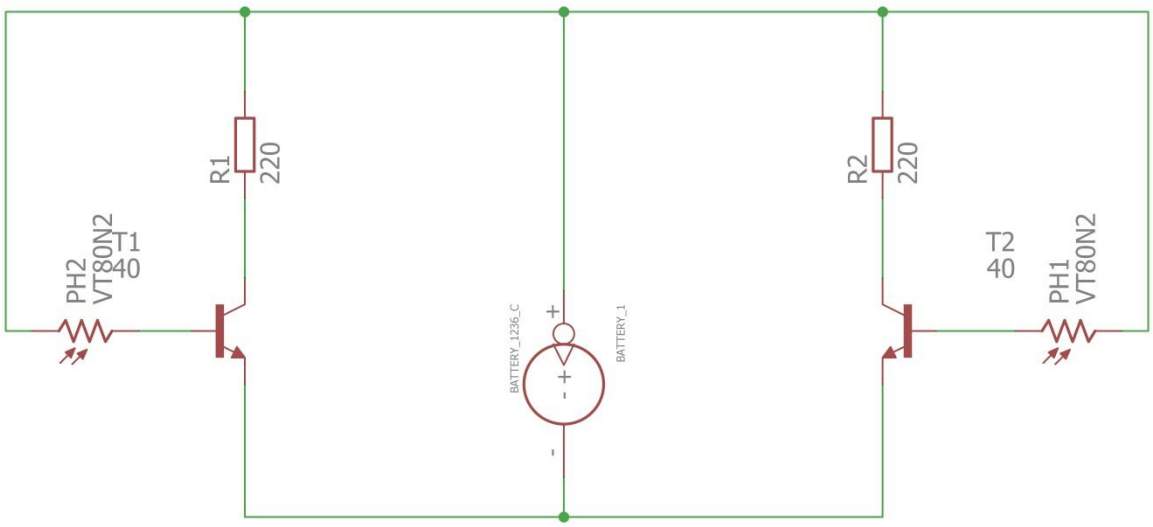


Figura 4.12 Esquemático del circuito

Para poder colocar los componentes del circuito a bordo del robot es necesario la construcción de una PCB, debido a lo compacto del robot colocar cables para unir los componentes es poco práctico debido a la falta de espacio, además el uso de una PCB permite ordenar los componentes dentro del espacio del robot. En la Figura 4.13 se observa el diseño de la PCB, en el software se colocaron los componentes y luego se unen las rutas en las capas superior e inferior. En la imagen aparecen 2 resistencias, estas resistencias reemplazan al motor. Luego este diseño se debe convertir en un archivo gerber para poder mandarlo a fabricar, la visualización de este archivo gerber se puede observar en la Figura 4.14. Luego de mandar el archivo gerber al fabricante se tiene la PCB terminada, esto puede observarse en Figura 4.15.

Una vez se tiene la PCB se procede a soldar los componentes del circuito, luego la PCB se suelda al porta batería y el porta batería se coloca en la base del robot, esto se puede observar en la Figura 4.16.

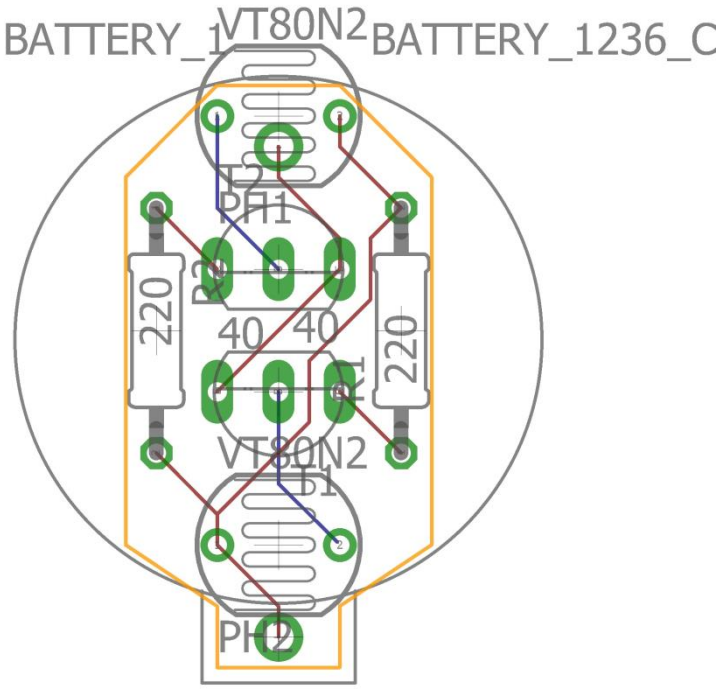


Figura 4.13 Diseño de la PCB, componentes y rutas.

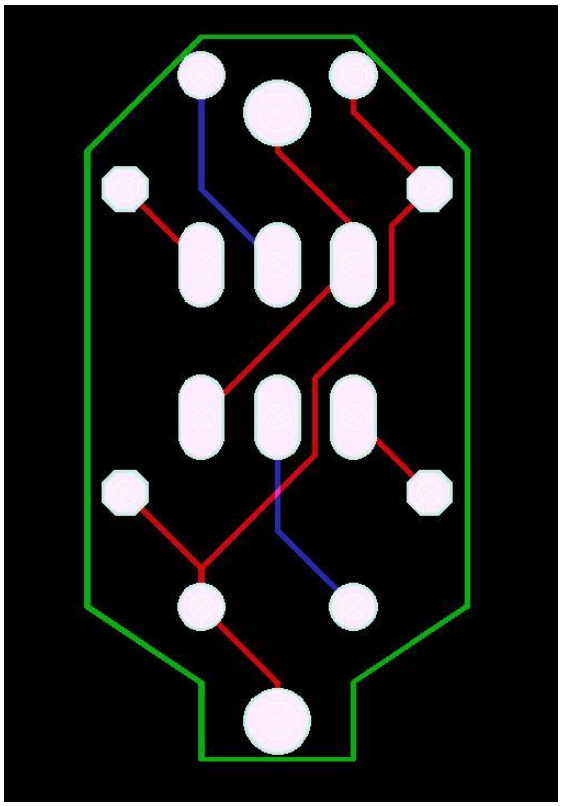


Figura 4.14 Visualización del archivo gerber.

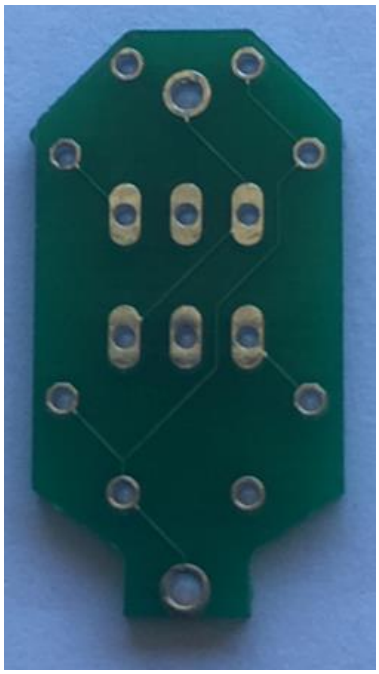


Figura 4.15 PCB ya fabricada

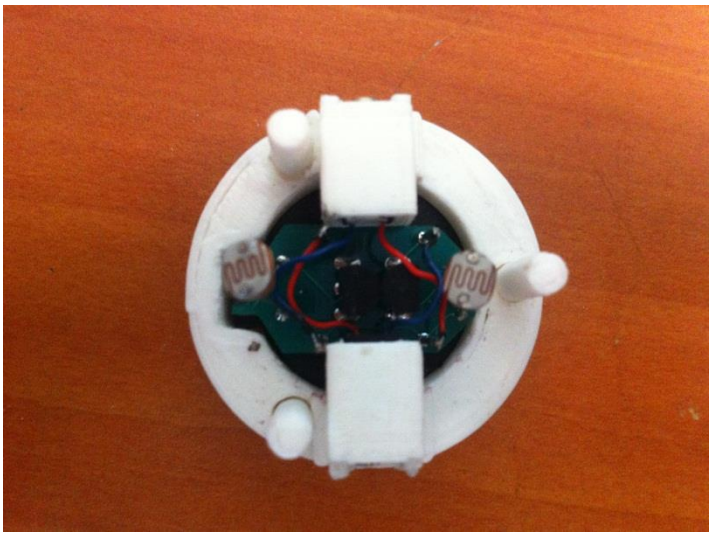


Figura 4.16 Robot con PCB más componentes soldados

Al robot finalizado se le denomina Cheavibot 2.0, esto se debe a que en trabajos anteriores se desarrolló el Cheavibot modelo en el cual se basa la presente memoria. En la Figura 4.17 se aprecia el Cheavibot, algunos cambios importantes que vale la pena mencionar son: el Cheavibot 2.0 se compone de varias piezas que se ensamblan para formar en el robot, en contraste con el Cheavibot que está impreso como una sola pieza, el Cheavibot 2.0 tiene patas más robustas, ya que un problema del Cheavibot es que las patas al ser más delgadas resisten un menor esfuerzo de flexión y corren el riesgo de romperse fácilmente debido a una mala manipulación del robot.

Figura 4.17 Cheavibot

4.1.3 Marcadores infrarrojos

Para que el optitrack pueda detectar los robots es necesario que estos tengan marcadores infrarrojos, estos marcadores son una cinta especial que refleja la luz infrarroja, se colocan en la tapa del robot. Cada marcador es un cuadrado de 4mm por lado, estos marcadores se colocan con forma de triángulo isósceles con el vértice que tiene el ángulo distinto en el eje del robot que está marcado por la perpendicular que une a los motores. En la Figura 4.18 se observan los marcadores infrarrojo, adicionalmente se pueden observar las LDR.

Se tiene que 3 marcadores es el mínimo que permite definir un cuerpo rígido en el software motive, este software es el que recibe la información de las cámaras optitrack, la forma de triángulo es para permitir calcular el centro del robot, ya que el centro del triángulo coincide con el centro del robot. Los marcadores se colocan en el interior del robot para dar separación suficiente entre marcadores en caso de un choque con otro robot, en el caso de estar en el círculo exterior y que 2 marcadores se acerquen demasiado (menos de la distancia de resolución de las cámaras –aproximadamente 3mm) el software motive interpreta que los marcadores son solo 1 y provoca que el programa fallé (el control en Python).

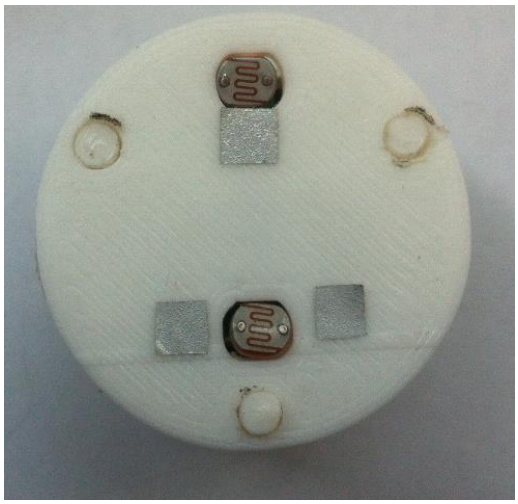


Figura 4.18 Vista superior del robot, se observa el arreglo de marcadores infrarrojo

4.2 Reglas de movimiento

De la sección 3.3 se tiene que hay 3 tipos de comportamientos a estudiar: movimiento colectivo, Self Assembly estructura de racimo y Self Assembly formación de patrones. En estos 3 comportamientos un elemento común es que cada robot solo puede ver lo que sucede en cierto radio, llamado el radio de interacción, más allá de este radio el robot es incapaz de percibir estímulos.

A continuación se presentas las reglas para cada tipo de movimiento.

4.2.1 Movimiento colectivo

El radio de interacción se define en 8r, en donde r es el radio del robot hay 3 reglas para este tipo de movimiento: repulsión, alineación, cohesión. Cada regla se aplica según los vecinos que existen dentro del área de interacción.

Para la regla de repulsión si dentro de un radio de 3r centrado en el robot se encuentra otro robot. El robot en cuestión calcula un punto central considerando a todos los robots en un círculo de diámetro 6r y se mueve hacia el lado contrario de ese punto.

Para la regla de alineación se tiene que si el robot no tiene ningún vecino en el área de repulsión y el siguiente vecino está en un radio menor a 6r se aplica esta regla. La regla de alineación calcula todas las velocidades de los vecinos y luego las promedia, el robot en

cuestión cambia la velocidad actual para acomodarse a la velocidad promedio, el robot se acomoda en pasos discretos de rapidez y dirección.

Para la regla de cohesión se tiene que dar el caso de que no haya ningún vecino en un radio de 6r, pero si uno en un radio de 8r, en este caso se calcula la posición promedio de todos los vecinos y el robot se mueve hacia ese punto.

En el caso de que no haya ningún vecino en un radio de 8r, el robot seguirá con la misma velocidad que tiene, hasta encontrar algún vecino.

4.3 Filosofía de control

El modelo de control que es usado en la presente memoria, es el de un sistema retroalimentado, el esquema de este sistema se puede observar en la Figura 4.19, el modelo es análogo al de un termostato, debido a que cada robot tiene una posición objetivo a la cual se debe mover y en cada paso se compara esta posición objetivo con la posición actual y luego se corrige, en el caso de un termostato se define una temperatura objetivo y esta se compara con la temperatura actual, a partir de la comparación se sabe si hay que aumentar o disminuir la temperatura para alcanzar el objetivo.

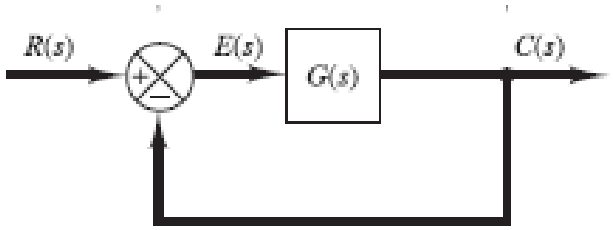


Figura 4.19 Esquema del modelo de control usado

Para que sea un sistema retroalimentado se debe tener una comparación, esta comparación ocurre al principio de cada ciclo, el sistema funciona a 24 fps (24hz de frecuencia), el elemento encargado de realizar la comparación es la cámara optitrack, la cual detecta la posición de los robots.

El flujo de datos del sistema se puede observar en la Figura 4.20, se tiene que cada robot cuenta con 3 marcadores infrarrojos ubicados en la tapa, estos marcadores son detectados por el optitrack y este a su vez envía la información de la posición de los robots a un computador central. En el computador es donde ocurre el proceso de comparación, en donde se ve la posición actual de los robots versus la posición objetivo de estos, luego que se tiene la comparación y la subsecuente acción de corrección se envía una señal a los

motores por medio del proyector LDP, el proyector debe iluminar la LDR correspondiente para efectuar el desplazamiento del robot, luego de que el robot se mueve el ciclo se repite y se vuelve a comparar la nueva posición del robot versus la posición objetivo. De esta forma se tiene un sistema de lazo cerrado en donde solo se tiene la función de transferencia y la comparación, como se ve en el esquema de la Figura 4.19.

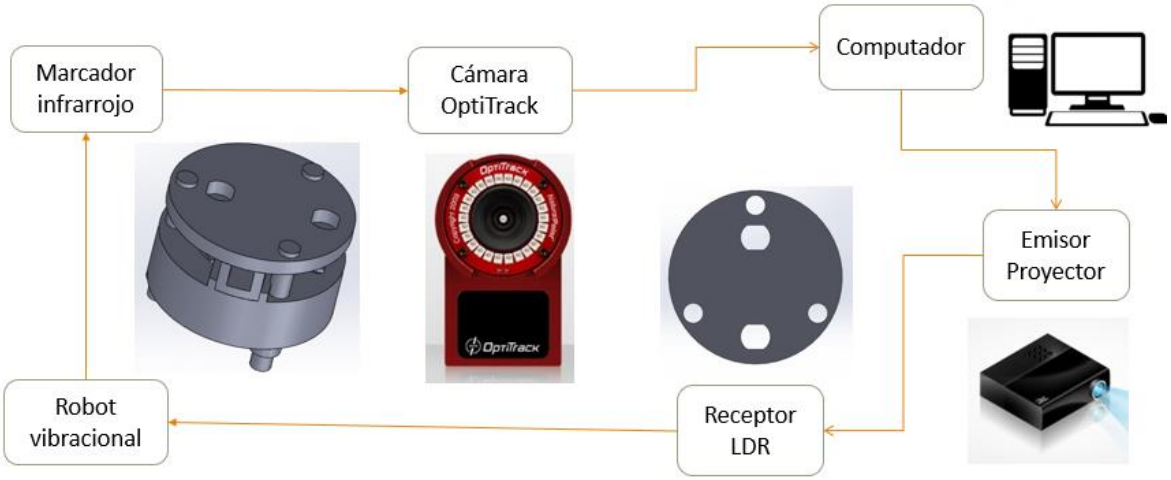


Figura 4.20 diagrama del flujo de datos del sistema

4.4 Setup experimental

El Setup experimental se compone de 2 partes, la fabricación y ubicación de las partes que lo componen y la calibración de los componentes. En la sección 3.1.2 se muestra como se calibran los componentes como el optitrack y también datos sobre la cámara oscura y el proyector DLP en la presente sección se muestra el resultado de la construcción de la cámara oscura y la ubicación de las cámaras optitrack y el proyector DLP. En una segunda parte se muestra la calibración del proyector.

4.4.1 Construcción Setup experimental

En la Figura 4.21 se muestra la cámara oscura por fuera, dentro hay un ambiente con poca luz lo cual permite que los robots se encuentren en estado apagado y solo se encienden cuando incide luz desde el proyector LDP en alguna LDR. En la Figura 4.22 se observa las cámaras optitrack y el proyector DLP estos son el receptor y el emisor respectivamente, el optitrack recibe la posición de los robot mientras el proyector envía luz a las LDR

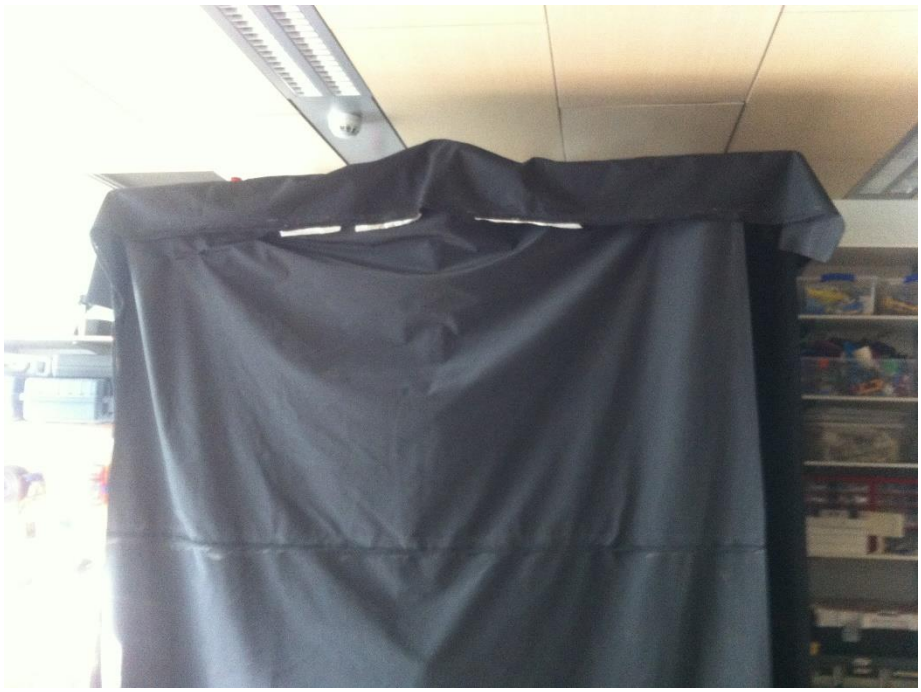


Figura 4.21 Cámara oscura, vista frontal

Figura 4.22 Interior de la cámara oscura, centro proyector DLP, costados cámaras optitrack

4.4.2 Calibración de la imagen proyectada

Debido a que el proyector no es completamente lineal, esto quiere decir que las medidas varían según donde se proyecta y a que la alineación del proyector con el área a proyectar no es perfecta se produce un corrimiento entre la posición real del robot y la posición proyectada.

Este desplazamiento es distinto para cada posición y además no sigue un comportamiento lineal, por esta razón es necesario hacer una calibración de la imagen. Un diagrama de cómo se visualiza este corrimiento se observa en la Figura 4.23 los robots están representados por su diámetro exterior, el robot real es un cuerpo físico, mientras el robot proyectado es un circulo de luz que se observa sobre el robot real. Cuando ambos círculos coinciden (se admite una pequeña desviación de 2mm) se puede enviar los pulsos de luz al robot para controlarlo, en caso de que exista un desplazamiento, al enviar los pulsos de luz, la luz no apunta a la LDR y es imposible controlar el robot.

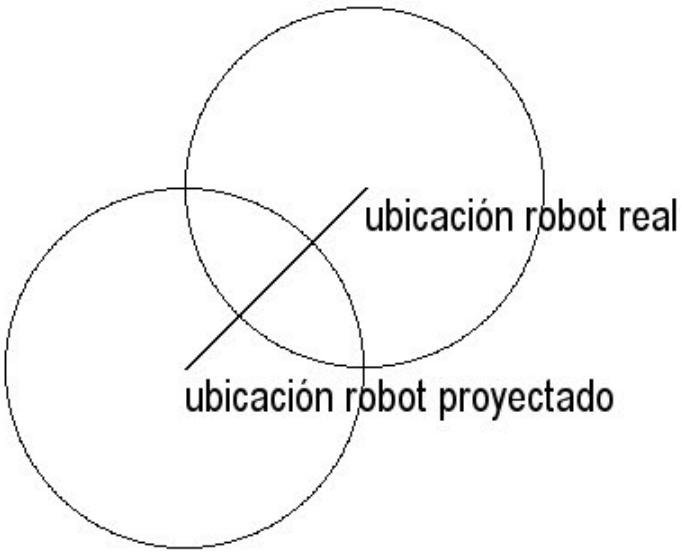


Figura 4.23 desplazamiento robot real vs robot proyectado

Para realizar esta calibración se tomaron 2 set de medidas, en ambos set de medidas se mide cuanto es el desplazamiento del robot proyectado versus el robot real, se mide tanto el desplazamiento en el eje x, como en el eje y. Los 2 set de medidas difieren en la posición del proyector. En la Figura 4.24 y en la Figura 4.25 se observa el set de medidas N°1 y N°2 respectivamente, en el centro hay un refinamiento de los datos, ya que corresponde al origen según el optitrack. Para corregir este desplazamiento se divide el área de trabajo en 9 cuadrantes, a cada cuadrante se le aplica una corrección de tipo lineal, en la práctica solo se

necesitó encontrar el valor de la constante, ya que la pendiente resulto tener un valor cero. Para realizar la calibración se coloca el robot en el cuadrante a calibrar, luego se procede a ajustar el valor de la pendiente y la constante de modo que el círculo proyectado coincida con el círculo del robot real.

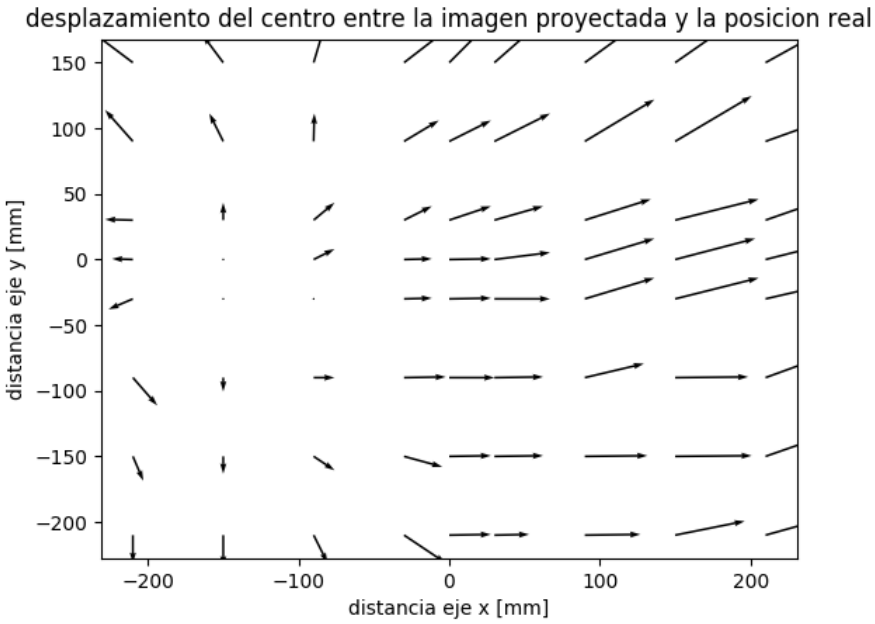


Figura 4.24 Desplazamiento entre robot real y robot proyectado, set de medidas N°1

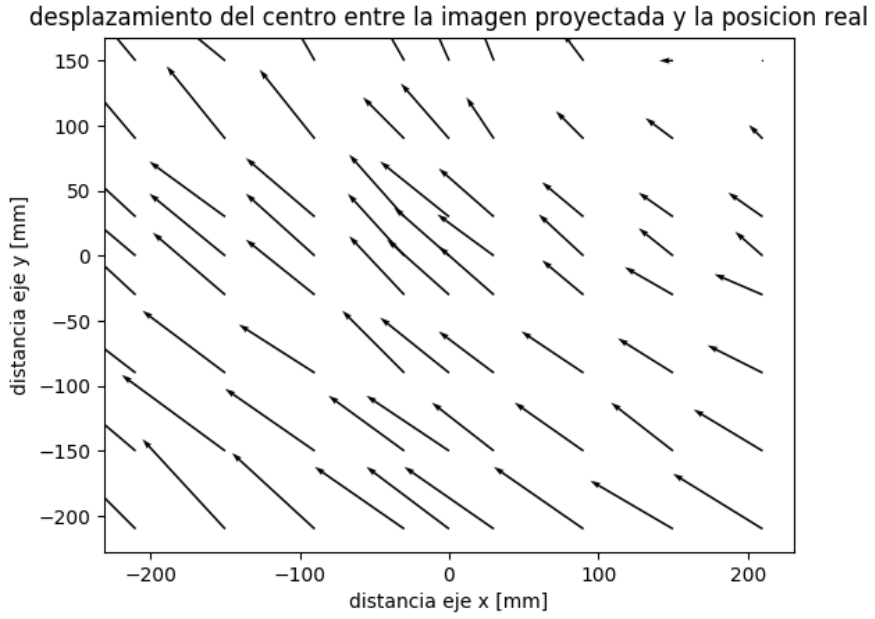


Figura 4.25 Desplazamiento entre robot real y robot proyectado, set de medidas N°2

4.5 Simulación

Se busca simular un enjambre robótico, en la simulación se considera un caso ideal, en donde el desplazamiento de cada unidad es el mismo, todas las propiedades de las unidades, tales como la masa, centro de inercia, fricción de las patas, se considera idénticas.

El primer paso en la simulación es obedecer los principios de la robótica de enjambre que se enuncian en 2.4, estos principios se simulan de la siguiente manera:

Homogeneidad: este principio se cumple debido a que cada robot es igual al resto y cumple las mismas tareas, la única diferencia que existe es una diferencia temporal que se debe a los estados que puede tomar el robot. Hay que notar que cada robot puede tomar cualquier estado y el estado depende solo de las condiciones dadas en el momento.

Interdependencia: esto principio se cumple debido a que la resolución de las tareas es solo posible cuando el sistema cuenta con varios elementos y existe cierta densidad de elementos, de modo que estos elementos puedan interactuar entre sí.

Comunicación y detección local: este principio se cumple debido a que cada robot recibe datos solo de robots situados dentro de un cierto radio de interacción. Para poder simular este principio, se debe resolver el problema de los vecinos cercanos con radio fijo.

Escalabilidad: debido a que este es un enjambre simulado, este principio no se cumple. Ya que es necesario encontrar los vecinos cercanos en un radio fijo, el algoritmo para resolver este problema es de orden n^2, para cada robot el tiempo es constante, pero debido a que se procesa en un computador central este tiempo incrementa con un orden n.

Robustez: este principio se cumple a medias, se puede simular una falla en un robot y el sistema sigue funcionando, sin embargo puede ocurrir una falla en el programa principal, lo cual implica que se cae todo el sistema.

4.5.1 Movimiento colectivo

En el movimiento colectivo se inician los robots con una posición al azar y una rapidez al azar, la posición se define por el ángulo que presenta el robot y va de 0 a 360°, la velocidad va en un rango de 0 y ¼ del radio del robot por ciclo. En la Figura 4.26 se observa gráficamente las posiciones iniciales de los robots representados por círculos, mientras la dirección es representada por la flecha que nace de cada círculo, el tamaño de la flecha representa la magnitud de la velocidad, en la Figura 4.27, se muestra la representación en un gráfico polar de la orientación inicial. En Figura 4.28 se representa la posición, orientación y rapidez en el paso N°30 y en la Figura 4.29 se representa la orientación en un gráfico polar

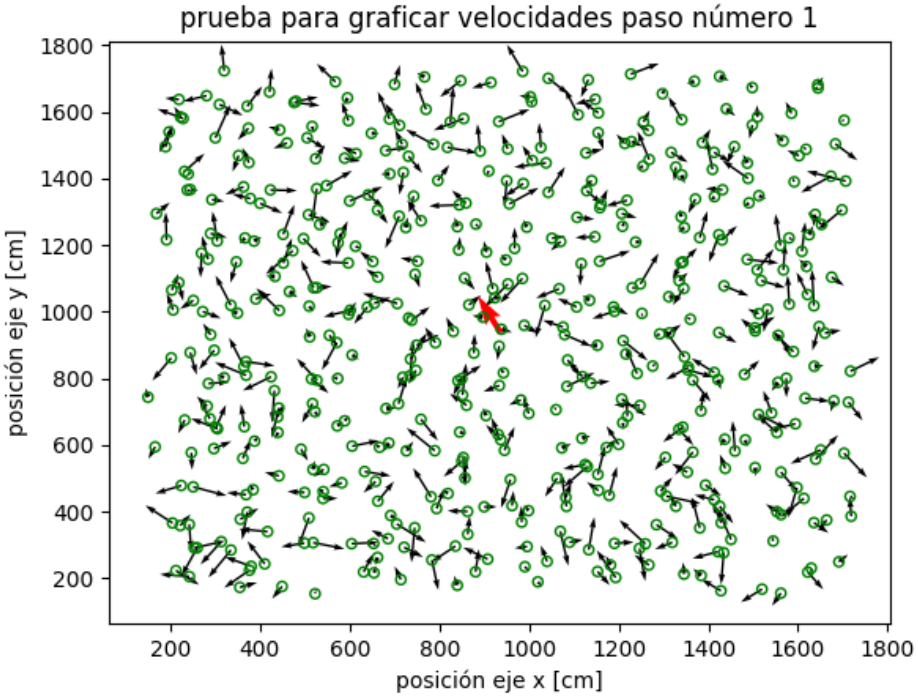


Figura 4.26 Posición y velocidades iniciales de los robots

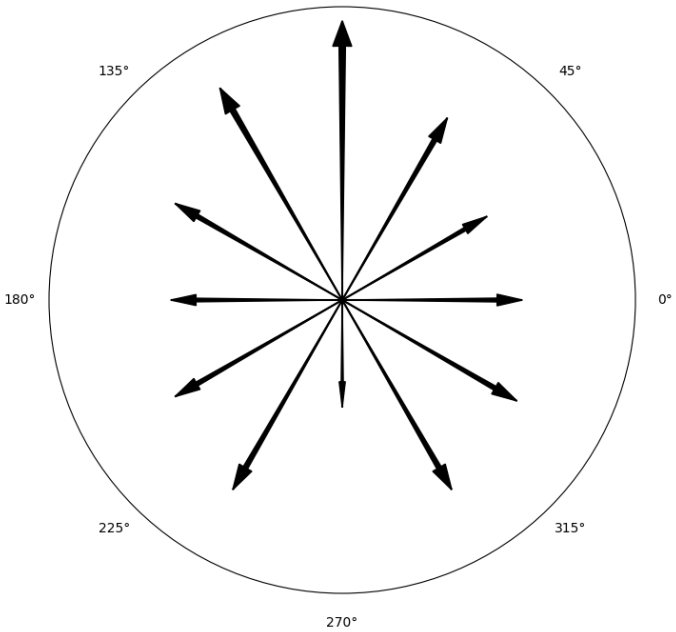


Figura 4.27 Gráfico polar de la orientación inicial de los robots

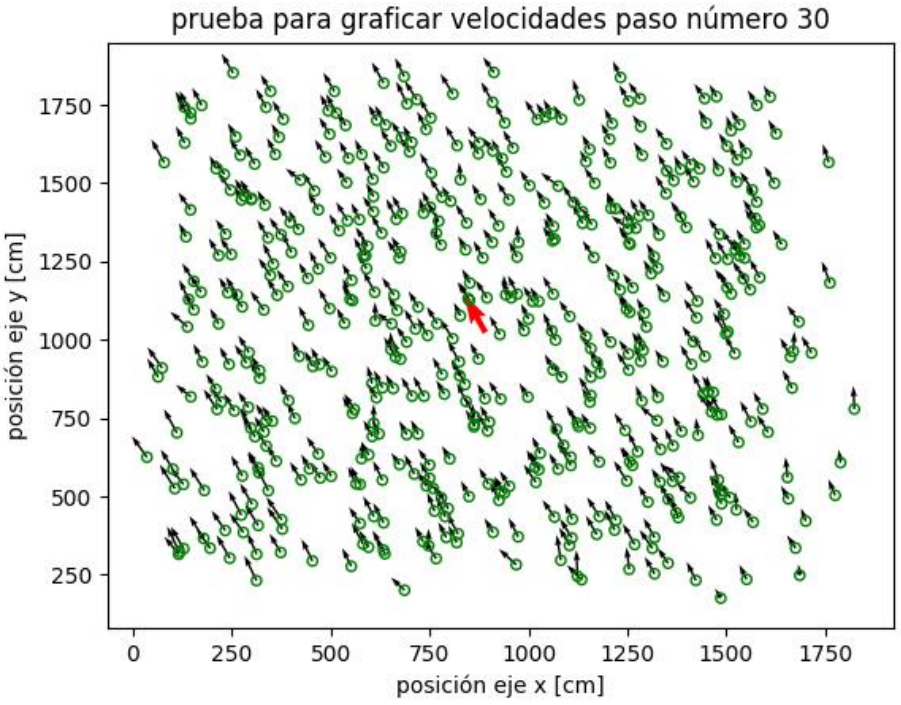


Figura 4.28 Posición y velocidades de los robots en el paso 30

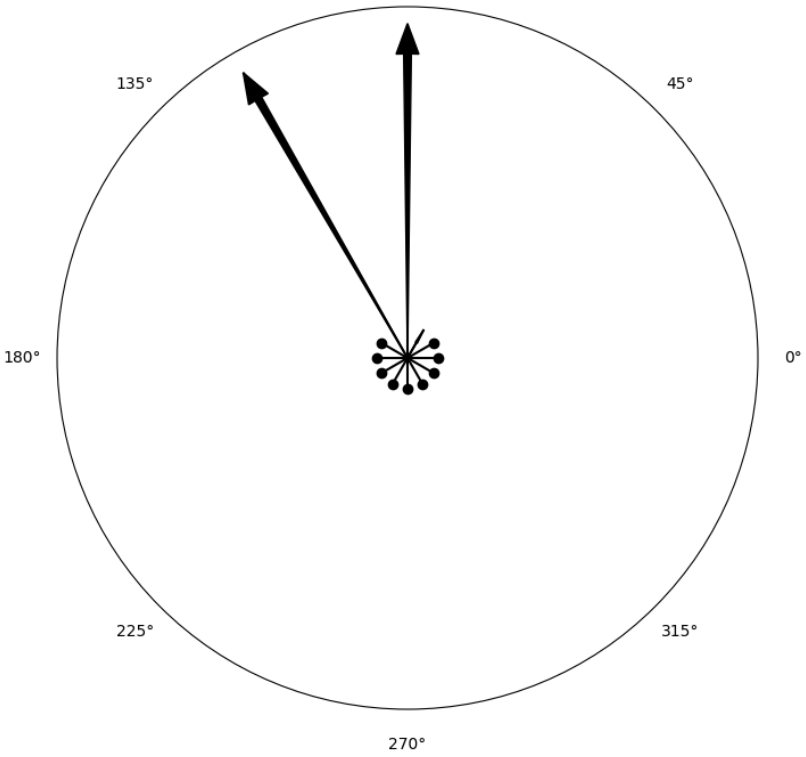


Figura 4.29 Gráfico polar de las orientaciones de los robots en el paso 30

4.5.2 Self Assembly

Para la simulación del Self Assembly del sistema se ocupan 2 experimentos: estructuras de racimo y formación de patrones.

4.5.2.1 Estructura de racimo

Este es una simulación que contiene pocas reglas, el objetivo es que cada robot tiene que moverse hacia un robot estático, al principio de la simulación se define un elemento estático que actúa de semilla. Cuando un robot en movimiento detecta a un robot estático, el robot en movimiento se mueve en dirección del robot estático, cuando la separación entre el robot en movimiento y el estático es menor que cierta cantidad (1/10 del diámetro del robot), el robot en movimiento cambia a estado estático, en la Figura 4.30 se observa el resultado final de esta simulación, en donde todos los robots terminan en estado estático.

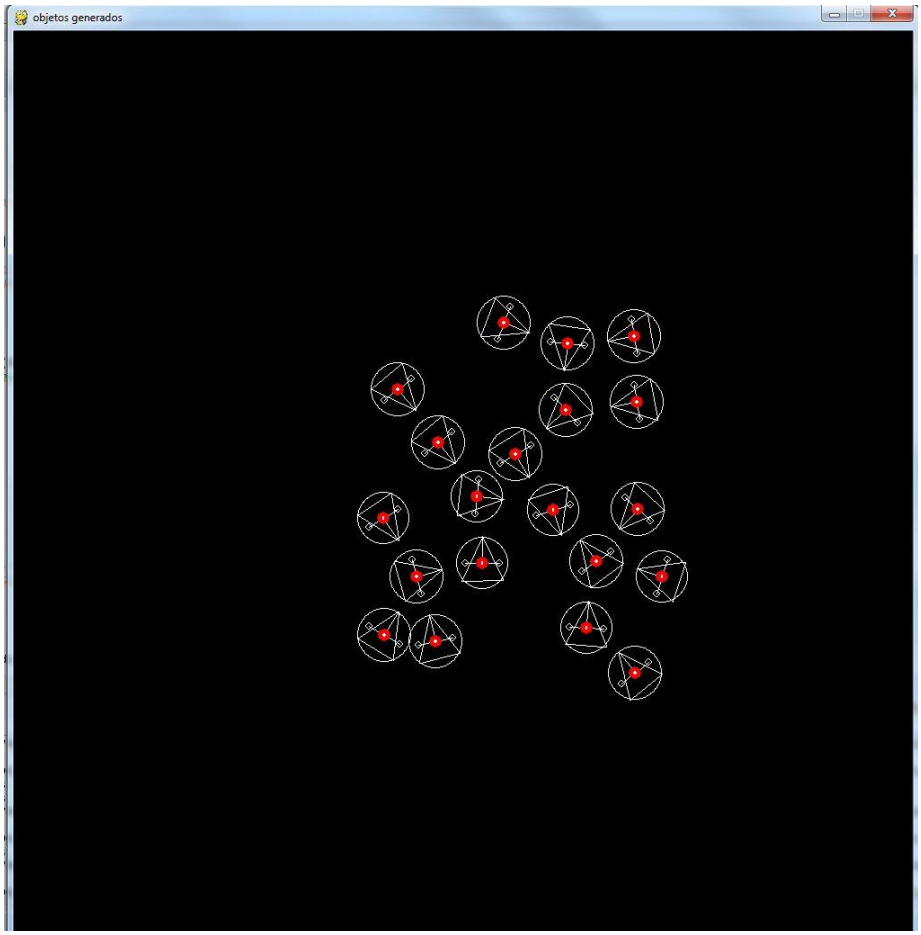


Figura 4.30 Formación de la estructura de racimo

Se le denomina estructura de racimo, porque a partir de la semilla empiezan a crecer ramas, las cuales a su vez se bifurcan en otras ramas.

4.5.2.2 Formación de patrones

Para la realización de patrones se sigue el algoritmo presentado por el kilobot en [3], para poder efectuar esta simulación se requieren varios puntos, estos son:

- Creación de figuras por el usuario
- Identificación de las seed
- Gradiente de posición
- Movimiento para salir del grupo de robots
- Movimiento por la periferia
- Posicionamiento del robot con respecto a la figura

La creación de figuras por parte del usuario se entregan de manera manual, la figura debe estar compuesta enteramente de círculos, esto se hace para simplificar los cálculos, ya que solo se necesita saber la distancia entre el centro del círculo que representa la figura y el centro del robot.

Las seed son un grupo de 4 robots que sirven de guía al resto, se colocan en forma de rombo como se muestra en la Figura 4.31, estos robots permanecen estáticos durante toda la simulación.

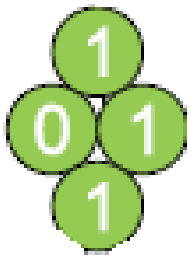


Figura 4.31 Creación de las seed

Para saber la posición individual de cada robot se utiliza como referencia a las seed, luego a cada robot se le asigna un gradiente dependiendo de qué tan lejos se encuentra de la seed, los detalles de esta asignación pueden encontrarse en la sección 3.3.2.

Para salir del pack se escoge un robot aleatorio de entre los que tienen el gradiente con mayor valor, luego se calcula el centro de los robots vecinos y el robot en cuestión se mueve al lado contrario, esto puede observarse en Figura 4.32, adicionalmente en la figura se observa las seed, la figura ingresada por el usuario, que en este caso es una estrella de 5 puntas formada por el círculo central y 2 círculos más para cada brazo.

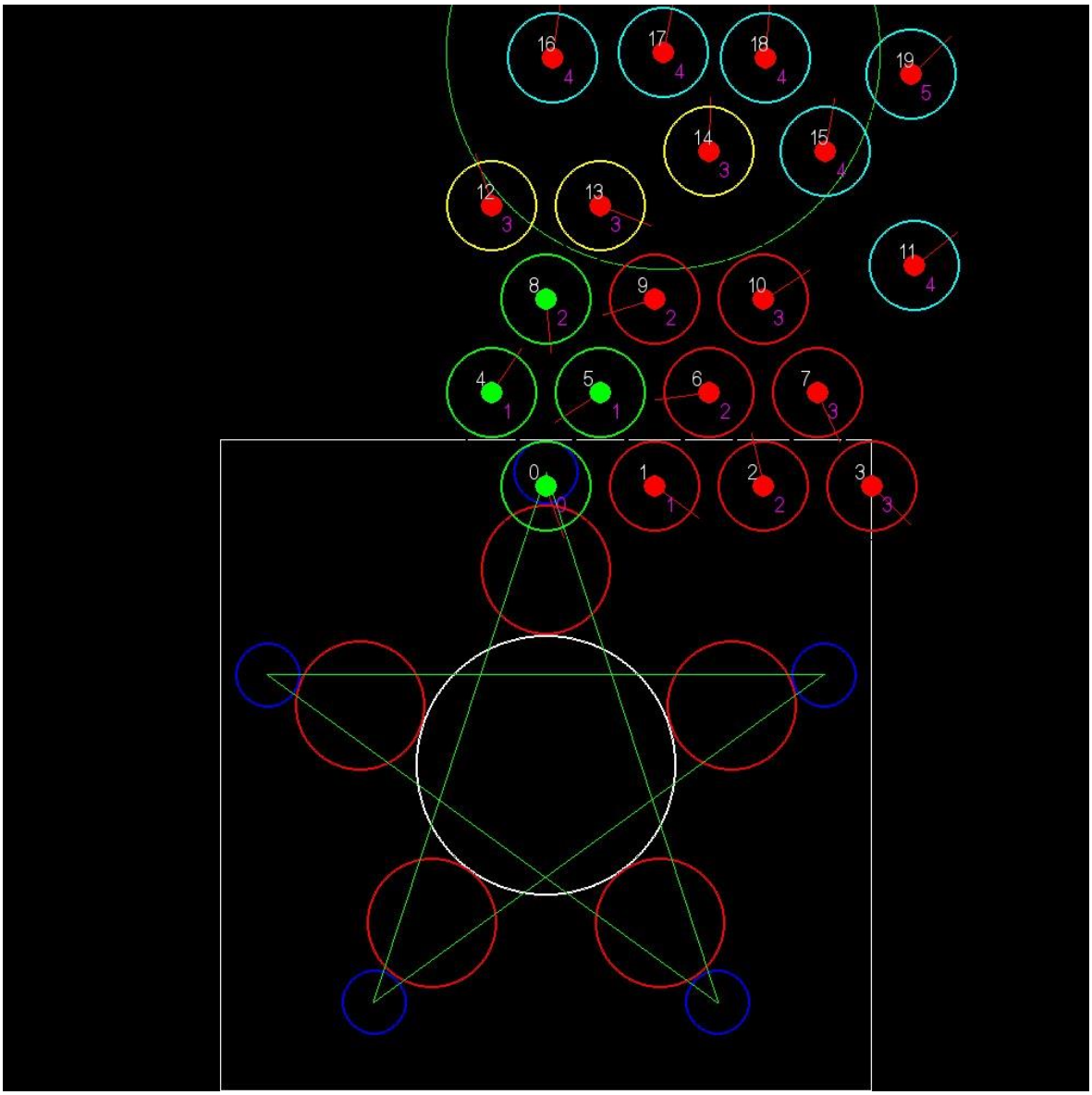


Figura 4.32 Movimiento fuera del pack de robots, adicionalmente se observa la figura ingresada por el usuario y el gradiente de cada robot.

El siguiente paso es el movimiento por la periferia, para realizar este movimiento se realiza un movimiento planetario, en el cual el robot que se desplaza fija como centro de la órbita al siguiente robot fijo, los robots se mueven de manera de minimizar el gradiente. Para realizar el movimiento planetario es necesario calcular la tangente, el robot se mueve manteniendo la tangente, si se aleja se moverá hacia el robot base y si se acerca al robot base, se moverá para acercarse a la línea de la órbita, el detalle de este movimiento se puede apreciar en la Figura 4.33.

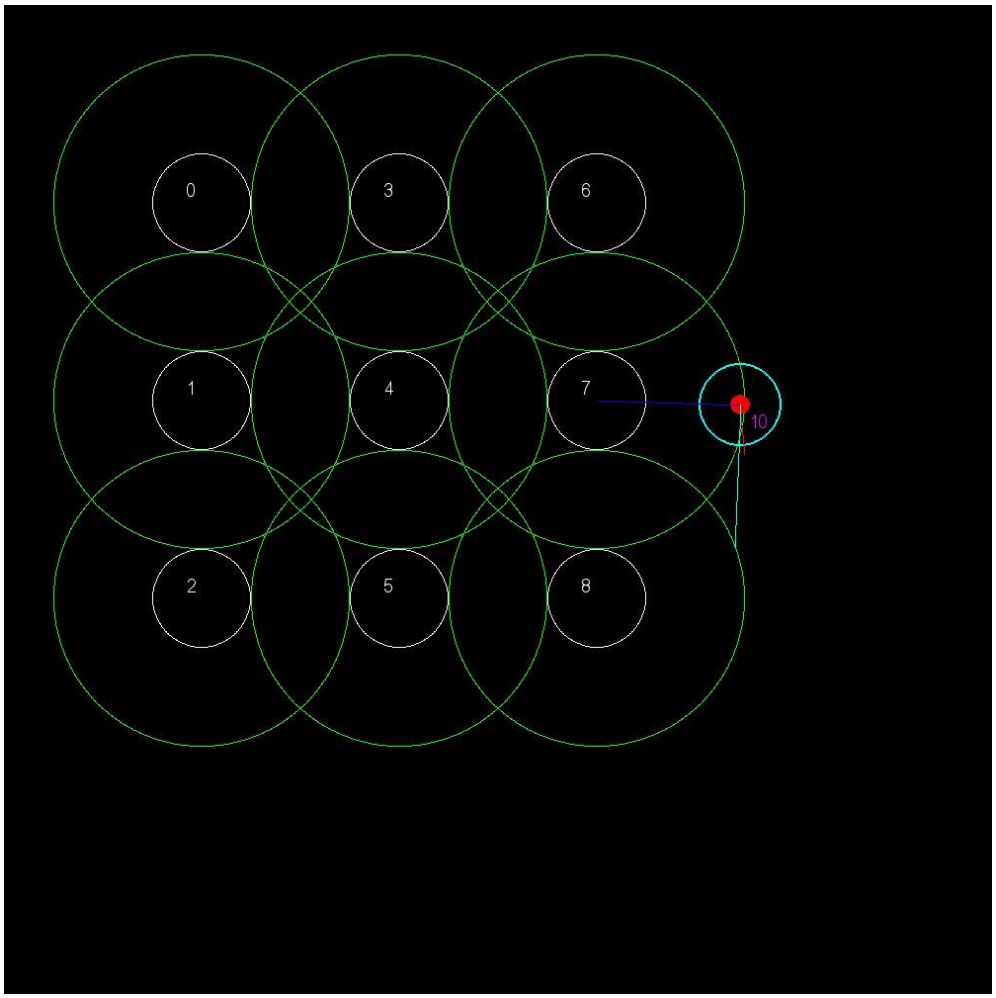


Figura 4.33 Movimiento por la periferia

El último paso en la simulación es que los robots puedan detectar cuando entran a la figura y detenerse dentro de la figura. Esto se logra comparando la distancia entre el robot y la figura ingresada por el usuario, hay que notar que el robot es capaz de obtener su posición en el espacio mediante la ayuda de los otros robots que componen el sistema. Cuando el robot entra en la figura detecta este cambio y cuando sale de la figura recibe la orden de detenerse, el anterior algoritmo solo sirve para rellenar el exterior de la figura, para llenar el interior se hace uso de los gradientes de posición, cuando un robot que se mueve dentro de la figura detecta a un robot con un gradiente mayor este se detiene, de este modo se llena el interior. Para llenar las capas superiores también se hace uso del gradiente, si un robot que aún no entra a la figura detecta a un robot con gradiente superior lo que hace es usarlo como base para el movimiento planetario y de esta forma la figura puede desplazarse una línea más arriba. Este resultado se muestra en la Figura 4.34.

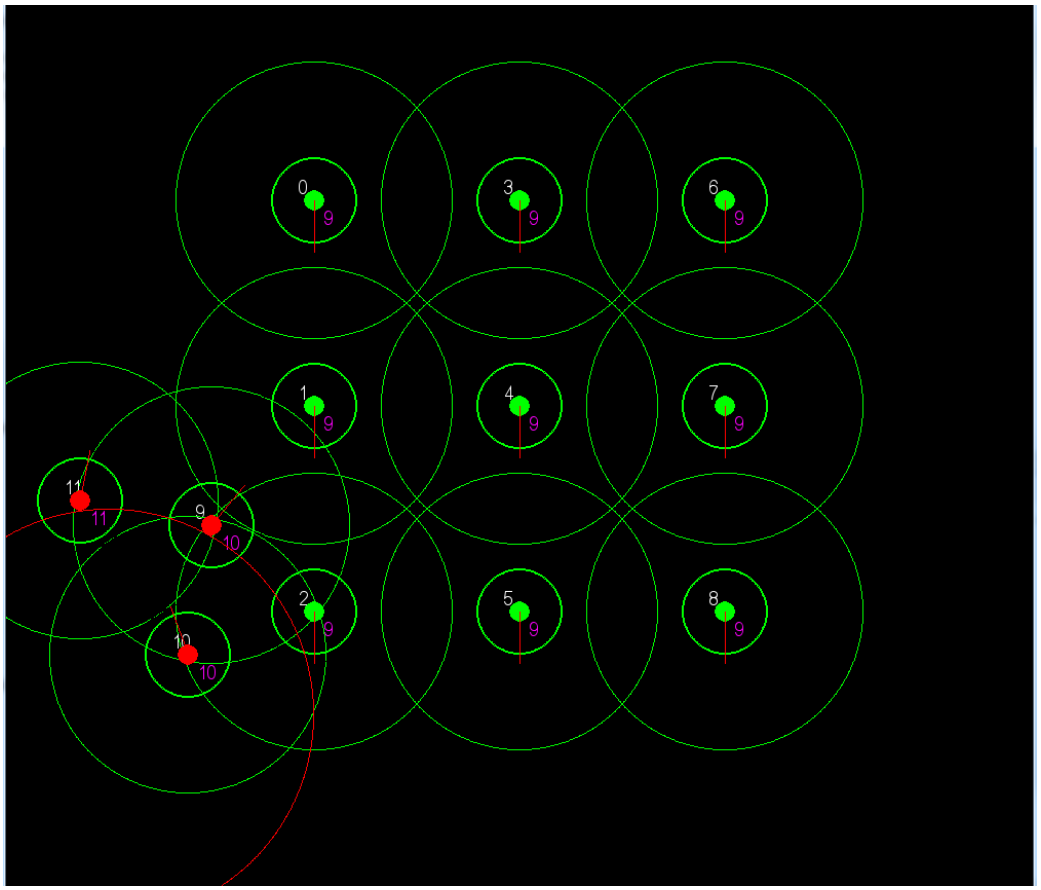


Figura 4.34 Localización de los robots con respecto a la figura

Con todos los pasos anteriores, se escribe el programa tanto para la simulación como para el control de los robots, en la Figura 4.35 se observa el resultado de una simulación con 60 unidades, estas unidades forman una estrella de 5 puntas, se observa que de las 60 unidades hay una unidad que no entra en la figura y queda siempre en movimiento (hasta que acaba la simulación o la batería en la realidad). El círculo en la imagen corresponde a otra figura de prueba.

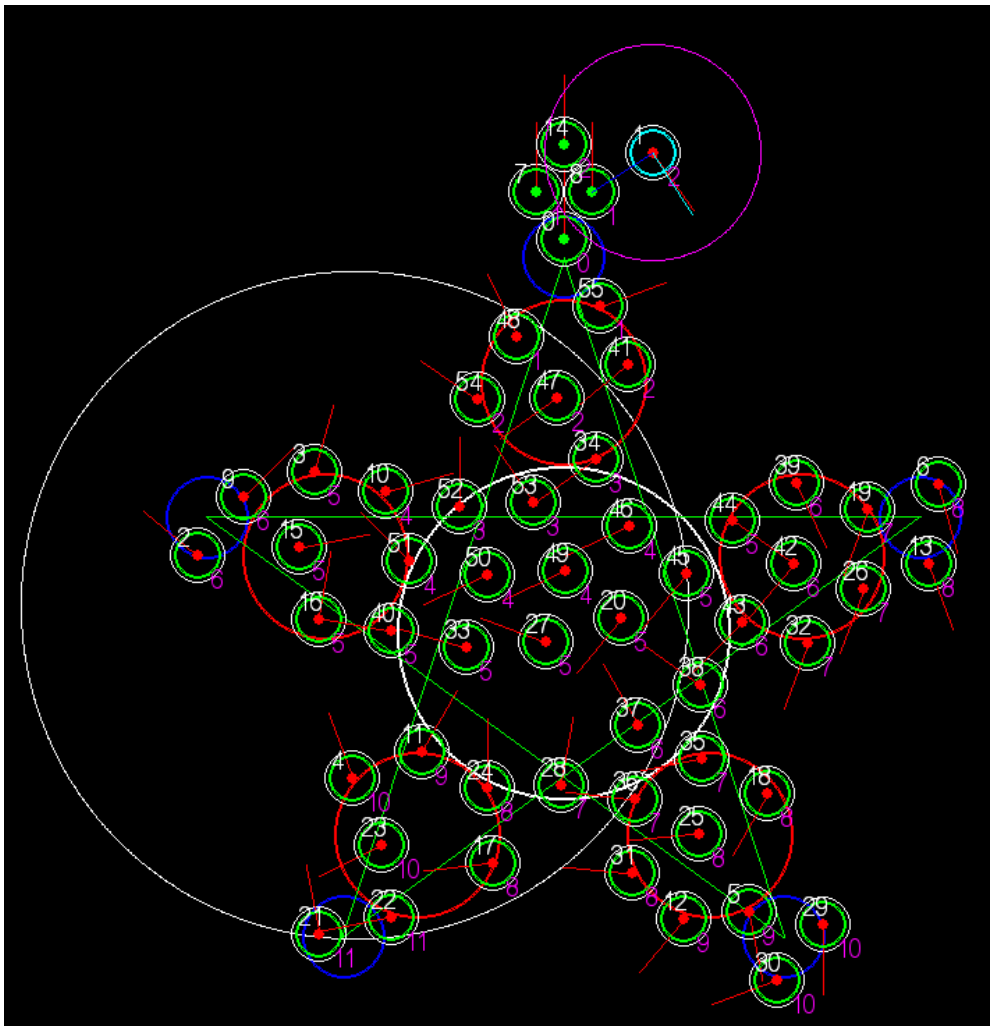


Figura 4.35 Self-Assembly, se observa la figura de una estrella completada con 60 robots

4.6 Influencia en el sistema según cantidad de unidades

En la sección anterior se analizan los distintos tipos de simulaciones con las cuales se trabaja, en la presente sección se analiza cómo afecta el número de robots con el desempeño del sistema. Para los tres tipos de simulación se analiza el desempeño del programa, tiempo entre cada paso y tiempo total de la simulación.

El tiempo total de simulación se define como el tiempo cuando se cumple el objetivo de la simulación, para cada uno de los tipos de simulación el objetivo es el siguiente:

- Movimiento colectivo: todos los elementos del sistema adoptan direcciones preferentes, esto se nota al analizar el área de interacción de cada robot individual. Al formarse clúster estos tienden a separarse con el tiempo, así en cada clúster se cumple la condición anterior.

- Self-Assembly, racimo: no hay robots con estado “móvil con vecino inmóvil”. Esto significa que todos los robots son parte de la estructura de racimo o que hay robots que están errando por el área de trabajo.

- Self-Assembly, formación patrones: los robots ya no ingresan a la figura o todos los robots han ingresado a la figura. Esto significa que la figura se completó y hay robots que sobran que continúan moviéndose o bien todos los robots entraron a la figura (figura completa con todos los robots o figura incompleta, por lo tanto faltan más robots para completar la figura)

Las simulaciones se ejecutan en un computador con un procesador Intel (R) Core (TM) i7-2600k @ 3.40GHz (8 CPUs), 16GB RAM. Se analizan los casos con 10, 25 y 50 elementos, dependiendo del caso se agrega un mayor número de elementos, para los tres tipos de simulaciones se realizan 3 sets de mediciones, cada set tiene distintos parámetros de inicialización, se realizan las simulaciones para las cantidades indicadas con visualizadores gráficos (con graf) y sin estos (sin graf).

Nota: Los visualizadores gráficos se muestran en pantalla para indicar ciertos estados de los robots, como puede ser la velocidad, la dirección de movimiento, etc. Su función es facilitar el entendimiento de la simulación al usuario.

Self Assembly: formación de patrones

Se analizan 4 casos, en los primeros 3 casos el radio del círculo a dibujar es 6 veces el radio del robot simulado en el último caso el radio del circulo a dibujar es 9.2 veces el radio del robot. Para el primer caso se analiza el comportamiento de 10 elementos en el sistema, en la práctica son 9 elementos; Para el segundo caso se analiza el comportamiento de 25 elementos en el sistema; Para el tercer caso se analiza el comportamiento del sistema con 50 elementos, en la práctica son 49 elementos. Los tres casos anteriores se rellena un circulo de radio 6 veces el radio del robot. Para el último caso se analiza el sistema con 50 unidades (49 realmente) y el circulo a rellenar tiene un radio de 9.2 veces el radio del robot.

Nota: hay una diferencia entre los elementos a analizar en el sistema con los que realmente se mide, por ejemplo se quiere analizar el comportamiento con 50 elementos, pero se analiza en la práctica con 49 elementos, esto se debe a que se busca que los robots se ordenen en un arreglo rectangular, por eso se eliminan los robots que están fuera del rectángulo que corresponde al empaquetamiento inicial.

Se toman 3 sets de medidas. En el primer set la velocidad de todos los robots es constante y su orientación es al azar según una distribución uniforme que va de 0 a 2π. Para el segundo set la velocidad de los robots es fija y apunta hacia arriba (según las figuras) en un ángulo de $3/2\,\pi$. Para el tercer set la velocidad de los robots es contante y el ángulo en el cual se orientan corresponde al ángulo de salida del robot del pack esto es $7/4\,\pi$. Para cada set de medidas se analizan los 4 casos mencionados anteriormente: 10, 25, 50 elementos y 50 elementos con un círculo a rellenar mayor. Adicionalmente para cada set de medidas se analiza como varían los tiempos con los gráficos que se dibujan en pantalla. En el primer caso (con graf) se toman los datos cuando están todas las ayudas visuales activadas, tales

como id del robot, información del gradiente, dirección de la velocidad, etc. En el segundo caso se toman los datos con las ayudas visuales mínimas para entender la simulación, como son: dibujar el contorno del robot, el estado seed y el estado de funcionamiento. Los datos tomados de estos casos se presentan en la Tabla 4.1, en la Tabla 4.2 y en la Tabla 4.3, en la tabla el 50+ representa que la simulación se lleva a cabo con 50 elementos, sin embargo el tamaño del círculo a rellenar es mayor.

Tabla 4.1 Datos para el primer set de medidas, ángulos al azar

	Elementos	Tiempo Frame [ms]	FPS	Tiempo Total [s]	Elementos en Figura
con graf	10	4	224	14	5
	25	12	81	165	19
	50	24	42	288	17
	50+	24	42.3	712	42
sin graf	10	0	2310	2.2	5
	25	2	540	20.8	19
	50	3	301	39.6	18
	50+	3	309	89	42

Tabla 4.2 Datos para el segundo set de medidas, ángulos en 270°

	Elementos	Tiempo Frame [ms]	FPS	Tiempo Total [s]	Elementos en Figura
con graf	10	5	195.3	16.7	5
	25	12	80	160.8	19
	50	24	42	300.6	18
	50+	24	42	711.7	42
sin graf	10	0	2361	2	5
	25	1	919.6	20.4	19
	50	2	571.8	37.6	18
	50+	2	479.7	87.3	41

Tabla 4.3 Datos para el tercer set de medidas, ángulos en 315°

	Elementos	Tiempo Frame [ms]	FPS	Tiempo Total [s]	Elementos en Figura
con graf	10	5	191.5	16.8	5
	25	15	68.7	191.3	19
	50	29	35	354	18
	50+	24	42.2	721.7	41
sin graf	10	0	2227	1.9	5
	25	1	985.4	19.6	19
	50	2	522.3	36.5	18
	50+	2	508.9	84.6	41

Las condiciones iniciales y finales de la simulación se muestran en las siguientes imágenes. En la Figura 4.36 aparece el caso con 10(9) elementos en el sistema, en la Figura 4.37 aparece el caso con 25 elementos en el sistema y en la Figura 4.38 aparece el caso con 50(49) elementos en el sistema, estas figuras son tomadas cuando están todas las ayudas visuales activadas. Adicionalmente en la Figura 4.39 se muestra el caso con 50(49) elementos con un círculo a rellenar de mayor diámetro.

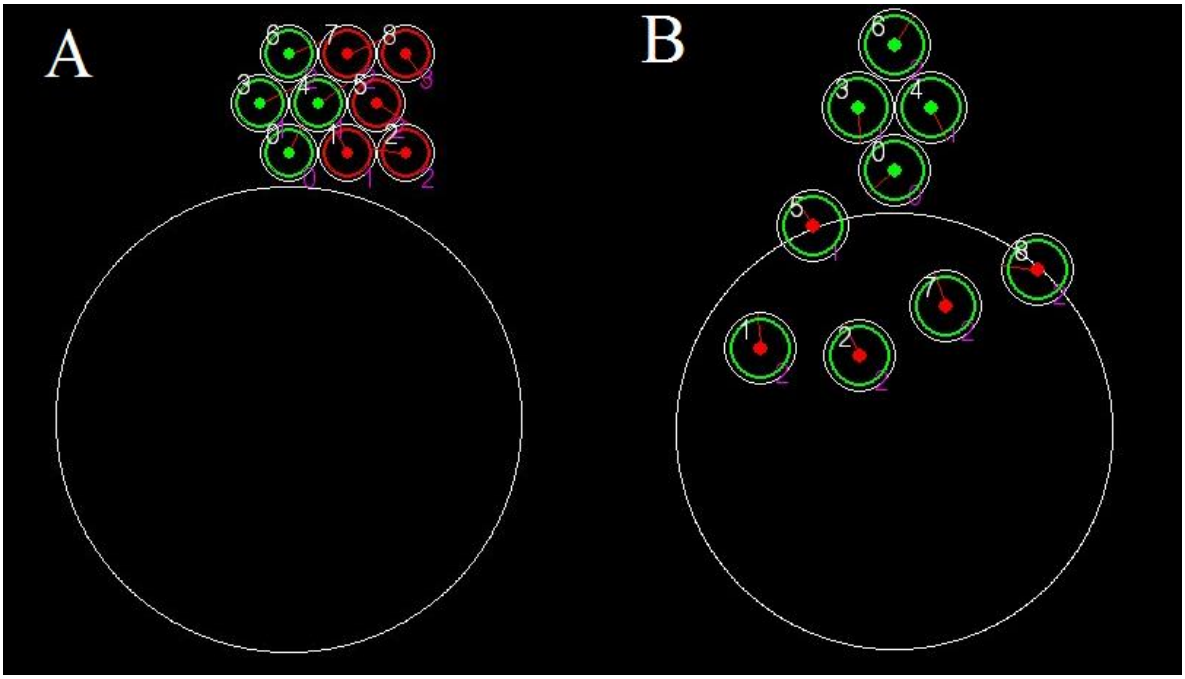


Figura 4.36 Simulación con 10 elementos formando un círculo, A) condición inicial, B) condición final.

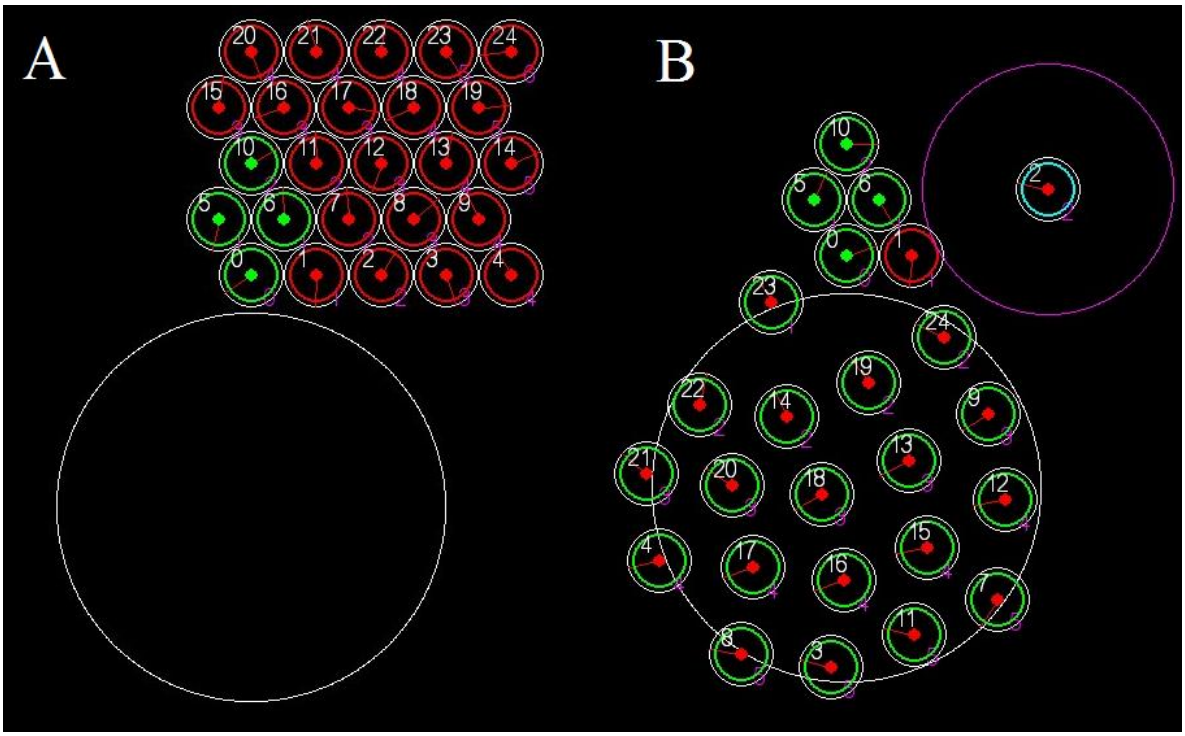


Figura 4.37 Simulación con 25 elementos formando un círculo, A) condición inicial, B) condición final.

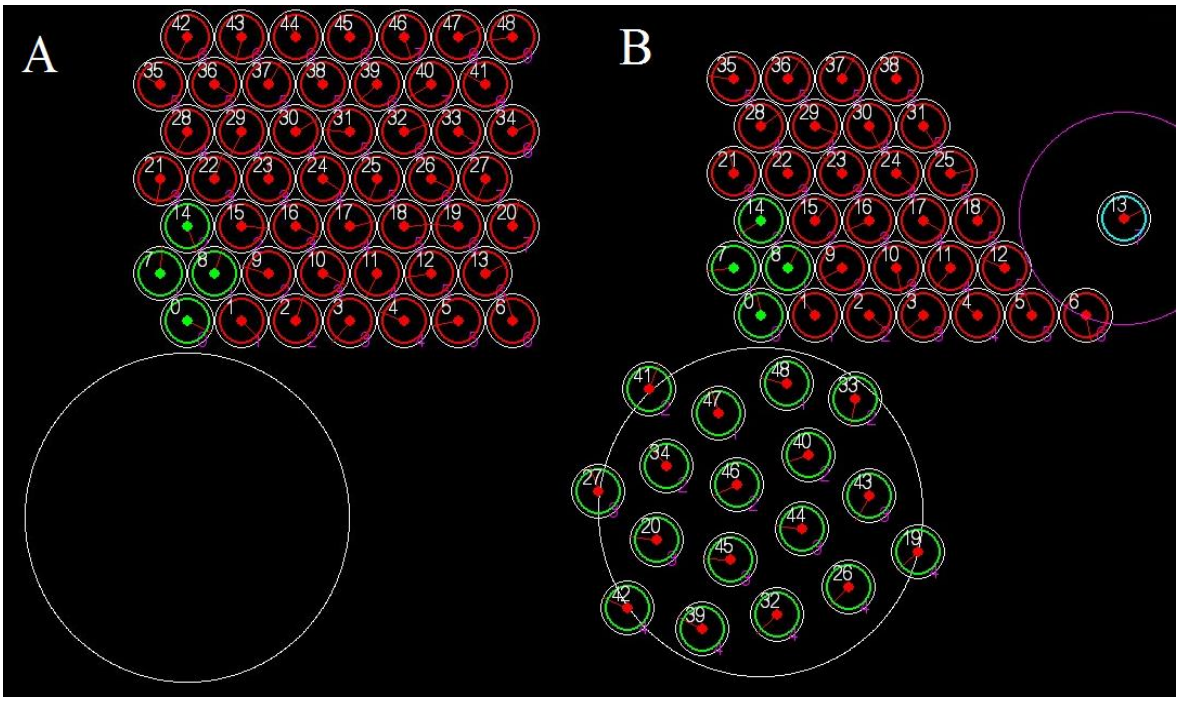


Figura 4.38 Simulación con 50 elementos formando un círculo, A) condición inicial, B) condición final

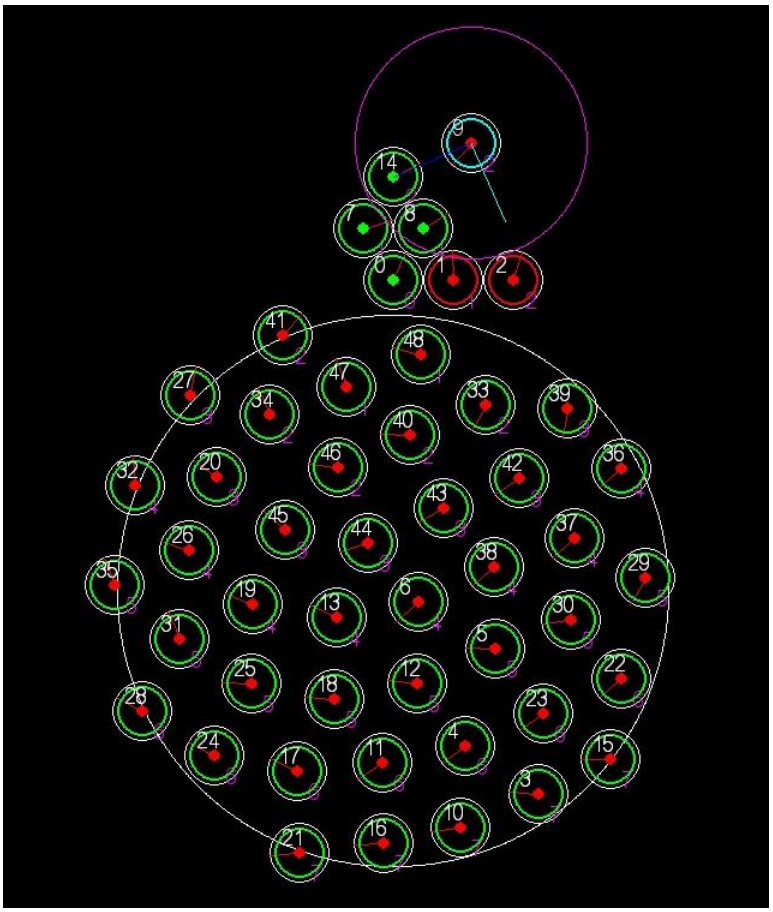


Figura 4.39 Simulación con 50 unidades, el círculo a dibujar tiene un diámetro mayor al dibujado en las anteriores simulaciones.

4.7 Baterías

4.7.1 Carga y descarga con corrientes superiores a la nominal

Como se sabe de la sección 2.13, las baterías CR2032 usadas en el desarrollo de la presenta memoria, presentan un comportamiento de rápida perdida de voltaje cuando son descargadas con una corriente que supera la corriente nominal de descarga. Según el cálculo de la vida útil de la batería, estas deberían durar 2 horas, sin embargo se tiene que solo pueden funcionar en óptimas condiciones por aproximadamente 3 minutos, luego de esto el voltaje decrece a aproximadamente 1.9 V y los motores no tienen suficiente potencia para mover el robot. Lo anterior es por el fenómeno de la descarga con una corriente mayor a la nominal.

4.7.2 Funcionamiento en Stand by.

Se realizaron mediciones al voltaje de la batería cuando esta esta puesta en el robot y el robot está en estado apagado, en la Tabla 4.4 se observan los valores del voltaje de la batería. Para este efecto se deja el robot dentro de una caja opaca, de manera de aislar al robot de la luz ambiental.

Tabla 4.4 Voltaje de la batería en Stand by en el tiempo

Tiempo	Voltaje
0 segundos	3.36 V
4 horas 40 min	3.06 V
7 horas 6 min	2.94 V
6 días	2.98 V
13 días	2.95 V
27 días	3.08V
34 días	2.94 V
41 días	3.00 V
50 días	3.07 V
57 días	3.05 V
75 días	3.07 V

4.8 Movimiento Cheavibot 2.0

Se realizaron pruebas cualitativas para determinar cómo afecta la masa del robot al movimiento del mismo, partiendo de la masa base que posee el Cheavibot 2.0 se agregaron pesos de 5gramos para estudiar el movimiento. Se encontró que con 15 gramos extra el robot prácticamente no se desplaza y el único movimiento observable es el giro sobre su propio centro, en el caso de los 5 gramos y los 10 gramos extra, se tiene que el radio en el cual se desplaza el robot mientras gira disminuye.

5 Discusiones

5.1 Cheavibot 2.0

Se conservan las características principales del Cheavibot, como la distribución de las patas en un triángulo equilátero y la ubicación de los motores, sin embargo se realizan algunos cambios en el diseño con el objetivo de que sea más fácil de fabricar en grandes escalas, también se busca que el modelo sea fácil de ensamblar y desensamblar. En cuanto a la estructura, se aumentó el diámetro de las patas de modo que estas puedan soportar mayores esfuerzos de flexión, el objetivo de esto es evitar que las patas se rompan debido a una manipulación indebida.

En cuanto al circuito se realizaron ciertas modificaciones, debido a que se encontraron 2 configuraciones que cumplen el objetivo de prender y apagar el motor con luz, para poder decidir cuál configuración escoger se midieron los voltajes y corrientes de los componentes en funcionamiento, se encontró que la configuración del circuito en el Cheavibot tiene 2/3 de voltaje y corriente en el motor en comparación con la configuración usada en el Cheavibot 2.0.

Para facilitar la colocación de los componentes del circuito en el reducido espacio que tiene el Cheavibot 2.0 se procedió a la fabricación de una PCB, además la PCB permite que el proceso de ensamblar los componentes electrónicos se pueda realizar como un proceso de fabricación en cadena, se optó porque todos los componentes sean de agujero pasante para abaratar los costos de la fabricación y debido a que los componentes que tienen una contraparte como smd son las LDR y el transistor y el mayor tiempo de soldado se ocupa en colocar el motor y los pines que conectan el motor a la PCB y estos componentes deben ser soldados como agujero pasante.

Se tiene que el Cheavibot 2.0 al girar no lo hace en torno a su propio centro, sino en torno a un punto alejado del centro que se encuentra fuera del robot a aproximadamente 50 mm del centro del robot, en primera instancia este comportamiento puede parecer indeseable para el control de los robots, sin embargo el movimiento del Cheavibot 2.0 se asemeja bastante a los seres vivos, los cuales al momento de cambiar de dirección tienen cierto radio de giro. Debido a lo anterior el estudio de enjambres biológicos con los Cheavibots 2.0 se torna interesante.

5.2 Operación pilas

Debido al fenómeno de descarga a corrientes superiores a la nominal, el robot no puede funcionar de manera continua, para poder solucionar este problema se evita que la pila pierda totalmente el voltaje, para esto se evita dar un funcionamiento continuo al robot y se le da tiempo a la batería de recuperar el voltaje. Lo anterior se logra estableciendo ciclos de carga y descarga rápidos, cada ciclo dura 10 segundos, 5 segundos de funcionamiento del robot (descarga) y 5 segundos con el robot apagado (recuperación del voltaje).

Con respecto al funcionamiento en stand by, se tiene que el voltaje en la batería se conserva de los datos de la sección 4.7.2 se tiene que el voltaje permanece en torno a los 3 V, hay una ligera desviación, pero esto puede deberse al voltímetro con el cual se adquieren los datos. Al tomar los datos se observa que el voltaje de la batería aumenta ligeramente en cada segundo, aumenta en pasos de 0.05mA y pasado un tiempo a pasos de 0.001mA, para tener uniformidad en las mediciones se espera un tiempo de 10 segundos para registrar el voltaje. Como el voltaje no muestra una tendencia a disminuir se tiene que el robot en stand by puede estar operativo por el tiempo de vida útil de la pila, lo cual es aproximadamente 5 años.

5.3 Simulación

Se logra simular el comportamiento de un enjambre robótico, sin embargo no se cumple el principio de escalabilidad, ya que por cada unidad que se agrega al sistema, aumenta el tiempo total de procesado.

Para la simulación de movimiento colectivo se tiene que todos los robots se alinean en la misma dirección y la magnitud de la rapidez también se iguala en todo el sistema, esto sucede tan solo en 30 pasos, hay que notar lo rápido que el sistema pasa del desorden al orden solo siguiendo 3 reglas simples. En vista de lo anterior se puede explicar porque los enjambres biológicos (por ejemplo cardúmenes de peces) pueden cambiar tan rápido la dirección del movimiento como un todo.

Para la simulación de Self Assembly con estructuras tipo racimo, se observa que el resultado final es similar al trabajo en el cual se basó esta simulación [13], esto implica que el sistema logra simular bastante bien el comportamiento de otro enjambre robótico.

5.4 Influencia de la cantidad de elementos en el sistema

En todas las simulaciones se observa que el tiempo entre frames aumenta a medida que aumenta el número de elementos en el sistema. Esto se debe a que el presente trabajo es un enjambre hibrido, esto significa que se simula el comportamiento de enjambre en un

computador centralizado. Ciertamente los cálculos individuales de cada robot son a tiempo constante, sin embargo estos cálculos se hacen de manera centralizada de modo que por cada robot que hay que analizar el tiempo se suma de manera lineal. Adicionalmente para detectar los vecinos cercanos con radio fijo se corre un algoritmo de fuerza bruta, el cual es de $O(n^2)$, lo cual implica que las operaciones crecen con el cuadrado de la cantidad.

5.4.1 Self Assembly: formación de figuras

Se observa de los datos de la sección 4.6 que el tiempo del Frame aumenta a medida que aumenta la cantidad de elementos en el sistema, tal cual se explica al comienzo de esta sección, también se observa que el tiempo cuando se agregan las ayudas visuales es mayor al tiempo sin las ayudas, esto se debe a que el sistema tiene que procesar menos operaciones. Un mayor tiempo de Frame implica que el techo máximo para los frames por segundo (FPS) disminuye, sin embargo en el peor caso los FPS están en 35, lo cual está sobre el esperado de la simulación, se espera que corra a un mínimo de 24 FPS.

En cuanto a la formación de figuras se observa que el sistema no tiene la capacidad de adaptarse o escalar la figura y solo la llena con los robots disponibles, quedando en algunas oportunidades medio llena como en el caso de 10 robots, en el caso de 25 robots se observa que la figura se llena, pero quedan 2 elementos fuera de la figura, en el caso de 50 robots se observa que la gran mayoría de los robots queda fuera de la figura. Se realiza una simulación adicional en la cual se incrementa el radio del círculo a rellenar de forma que la mayoría de los 50 robots puedan entrar a la figura.

Se observa que la calidad de las figuras aumenta según aumenta el tamaño de estas, por ejemplo el círculo donde se emplean 25 elementos tiene menor calidad al círculo donde se emplean 50 elementos, esto se puede interpretar haciendo una analogía con los pixeles de una imagen, en este caso cada robot representa un pixel, en el caso del circulo de 25 elementos se podría decir que este tiene una menor resolución al círculo de 50 elementos y por esto los bordes se ven más recortados.

En cuanto a los tiempos del Frame se tiene que estos solo se ven afectados por la cantidad de elementos en el sistema y la orientación inicial de cada robot no tiene incidencia en este tiempo. Con el tiempo total de simulación se tiene un caso similar, en el cual la orientación inicial no tiene incidencia en este tiempo, hay que notar que entre la activación de cada robot se tienen 1000 frames de diferencia, este tiempo de espera hace que cualquier ganancia que se pueda obtener de una dirección preferencial inicial se pierda.

6 Conclusiones

Se logró la construcción de un enjambre robótico con 50 unidades, este enjambre es controlado mediante luz blanca, se escoge luz blanca debido a que con este color se presenta la mayor intensidad de luz (por diseño del proyector), se observa que se pueden formar patrones, sin embargo hay desviación del patrón inicial. Cada robot tiene 2 grados de libertad, cada grado de libertad está controlado por un motor, los grados de libertad son giro en sentido horario y giro en sentido anti horario, el giro que se realiza no es sobre el propio centro del robot sino que es un giro en un círculo de un diámetro de aproximadamente 100mm.

Las bases para las reglas de movimiento se obtuvieron de la bibliografía, sin embargo se simplificaron, en el caso de movimiento colectivo las últimas tendencias son establecer pesos para las distintas reglas según la distancia de los demás elementos, dando lugar a una función continua de aplicación de los pesos. En el caso de la presente memoria las reglas se aplican con una función escalón en donde a cada área la corresponde una regla distinta.

La lógica de control se basa en un sistema retroalimentado simple, esto quiere decir que se tiene un comparador que corrige la dirección del robot según la regla especificada, esta corrección se da en cada paso, funciona de manera similar a un termostato.

Se tiene que se logra simular el comportamiento de un enjambre robótico, sin embargo este enjambre no cumple el principio de escalabilidad, ya que el tiempo de ejecución de cada ciclo depende del número total de elementos en el sistema. Sin embargo para la cantidad de elementos con los cuales se trabaja (50 robots físicos), la cantidad de ciclos por segundos que el sistema puede alcanzar (36), es mayor a la cantidad de ciclos del funcionamiento del sistema (24). Por lo tanto se tiene que todas las operaciones para que la simulación funcione se realizan en un menor tiempo al requerido y por lo tanto no existe el problema de una saturación de operaciones que baje el tiempo de respuesta del sistema.

Para la formación de figuras no existe un algoritmo que permita usar todos los robots, en algunos casos pueden faltar robots para rellenar la imagen o en otros sobrar algunos robots que no entran en la figura. Se observa que el tiempo de cada ciclo solo depende de la cantidad de robots del sistema y es mayor al tiempo definido en el sistema (42ms). En cuanto a la calidad de las figuras formadas se observa, que esta mejora al aumentar el tamaño de estas o dicho en otras palabras al aumentar la cantidad de robots que la componen, lo anterior es análogo a aumentar la resolución de una imagen.

7 Bibliografía

[1] Detrain C y Deneubourg J., (2006). Self-organized structures in a superorganism: Do ants "behave" like molecules?. *Physics of Life Reviews, 3*, pp.162-187.

[2] Whitesides G. y Grzybowski B., (2002). Self-assembly at all scales. *Science, 295 (5564),* pp. 2418–2421.

[3] Groß R. y Dorigo M., (2008). Self-Assembly at the Macroscopic Scale. *Proceedings of the IEEE, 96(9),* pp. 1490-1508.

[4] Vásquez M., (2015), *Análisis del comportamiento colectivo presentado por robots vibracionales. (Tesis Ing. Civil Mecánico).* Universidad de Chile.

[5] Rubenstein M., Ahler C., Hoff N., Cabrera A. y Nagpal R., (2014). Kilobot: A Low Cost Scalable Robot System for Collective Behaviors, *Robotics and Autonomous Systems*, 62, pp. 966–975

[6] Rubenstein M., Cornejo A. y Nagpal A., (2014) Programmable self-assembly in a thousand-robot swarm, *Science, 345 (6198),* pp.795-799.

[7] Yu C. y Nagpal R., (2010). A Self-adaptive Framework for Modular Robots in a Dynamic Environment: Theory and Applications, The International Journal of Robotics Research, DOI: 10.1177/0278364910384753.

[8] Iglesias M., (2015), Sistema de transporte de material con grupo de robots voladores miniatura. (Tesis Ing. Civil Eléctrico). Universidad de Chile.

[9] OptiTrack V120 trio. [en línea] < http://www.optitrack.com/products/v120-trio/ > [consulta: 28 de mayo 2018].

[10] OptiTrack General Faqs. [en línea] < http://www.optitrack.com/support/faq/general.html > [consulta: 28 de mayo 2018].

[11] Ogata K. (2010). *Ingeniería de Control Moderna*, 5° ed. Madrid: Pearson, 2010.

[12] Vicsek T., Czirok A., Ben-Jacob E., Cohen I. y Shochet O., (1995). Novel Type of Phase Transition in a System of Self-Driven Particles, *Physical Review Letters, 75* (6), pp.1226-1229.

[13] Vicsek T. y Zafeiris A., (2012). Collective motion, *Physics Reports, 517*, pp. 71–140.

[14] Bensan M., (2014). *Aplicación Móvil Georreferenciada de Búsqueda de Intereses en Común por Medio de un Sistema Diseñado para Alta Demanda*. (Tesis Ing. Civil Computación). Universidad de Chile.

[15] Bentley J., (1975). A Survey of Techniques for Fixed Radius Near Neighbor Searching, *US Energy Research and Development Administration, 94305.*

[16] Bentley J., (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM, 18*, pp. 509-517.

[17] Furset K. y Hoffman P., (2011). High pulse drain impact on CR2032 coin cell battery capacity, Nordic Semiconductor and Energizer.

[18] Vartholomeos P. y Papadopoulos E., (2006). Analysis, Design and Control of a Planar Micro-robot Driven by Two Centripetal-Force Actuators, *Proc. of the 2006 IEEE International Conference on Robotics and Automation*, pp. 649-654

[19] Papadopoulos E. y Chasparis G., (2004). Analysis and Model-Based Control of Servomechanisms with Friction, *ASME J. Dyn. Syst. Meas. Control, 126(4)*, pp. 911–915.

[20] Vartholomeos P. y Papadopoulos E., (2005). Analysis and Design of a Novel Mini-platform Employing Vibration Micro-motors, *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 3627-3632.

[21] Sciavicco L. y Siciliano B., (2001). *Modelling and Control of Robot Manipulators*. Londres: Springer-Verlag

[22] Groß R., Bonani M., Mondada F. y Dorigo M., (2006). Autonomous Self-Assembly in Swarm-Bots, *IEEE Transactions on Robotics, 22(6)*, pp. 1115-1130.

[23] Rubenstein M., Cornejo A. y Nagpal A., (2014) Supplementary Materials for Programmable self-assembly in a thousand-robot swarm, Science, 345 (6198). [en línea] < http://science.sciencemag.org/content/sci/suppl/2014/08/13/345.6198.795.DC1/Rubenstein.SM.pdf > [consulta: 14 de junio 2018].

[24] Vicsek T. y Zafeiris A., (2012). Collective Motion, *Physics Reports, 517*, pp. 71-140.

[25] Reynolds C., (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model, *Computer Graphics, 21(4)*, pp. 25-34.