VISION - Robot Micromouse

S. Barroso[†1], M. Pascuarelli[†2], P. Lopez[‡3]

[†]*Diseño de sistemas con microcontroladores*
Facultad de Ingeniería, Universidad Nacional del Comahue
Buenos Aires 1400, 8300 Neuquén
[1]santib01@hotmail.com
[2]pascuarellimarco@gmail.com
[3]lopezpauu@gmail.com

***Resumen*—En el presente informe se explicará el proceso de diseño, fabricación, ensamblaje, programación y testeo de *VISION*, un micromouse resuelve laberintos basado en el microcontrolador STM32F103C8T6. En particular se desarrollaran los algoritmos empleados para su funcionamiento y se proporcionara una idea básica de los métodos y sistemas de control implementados en el micromouse. De manera extra y a modo de investigación, se desarrollaran las ideas básicas sobre el control del sensor óptico (cámara de mouse inalámbrico) incorporado para, mediante el filtro de *Kalman*, realizar una mejor estimación de la posición.**

***Palabras clave*—Micromouse, resuelve laberintos, laberinto, STM32F103C8T6, robot, lenguaje #C, electrónica, Kalman Filter, estimación de trayectoria, modelado, seguimiento de trayectoria.**

I. Introducción a *VISION*

Un MicroMouse es un robot de búsqueda de laberintos que reune de forma compacta una gran cantidad de tecnologías, como un microprocesador, un mecanismo robusto, sensores que reconocen el laberinto, una técnica de control de motores, una búsqueda de laberintos y un algoritmo de derivación de la ruta más corta. El laberinto está formado por una cuadrícula de celdas de 200mm X 200mm, con paredes de 50 mm de altura. Los MicroMouse son robots completamente autónomos que deben encontrar su camino desde una posición inicial predeterminada hasta el área central del laberinto sin ayuda. El MicroMouse necesita realizar un seguimiento de dónde está, descubrir paredes mientras explora, trazar el mapa del laberinto y detectar cuándo ha alcanzado la meta. Una vez alcanzado el objetivo, el MicroMouse normalmente realizará búsquedas adicionales en el laberinto hasta que haya encontrado una ruta óptima desde el principio hasta el final. Una vez que se ha encontrado la ruta óptima, el MicroMouse ejecutará esa ruta en el menor tiempo posible. Todo esto se puede resumir en 3 etapas y/o procesos:

- Search Run
- Optimización
- Speed Run

Cada uno de estos procesos implica una serie de complejos algoritmos que se vinculan entre si para lograr la mejor performance a la hora de competir.

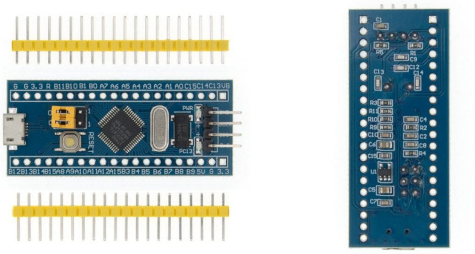
VISION, además de cumplir con estas 3 etapas, cuenta con una cámara para realizar mediciones de velocidad mediante la extracción de puntos característicos y, con la implementación de el filtro de *Kalman*, fusiona los datos provenientes de dicho sensor, con los provenientes de los encoders de los motores para lograr así una mejor aproximación de la posición relativa del robot en todo momento.

Dada la necesidad de un microcontrolador capaz de manejar tanta potencia computacional, optamos por utilizar una STM32 que funciona a 72MHz. El objetivo principal de *VISION* es nunca dejar de desarrollarlo ni mejorarlo. Hablando tanto de software como de hardware y, teniendo en cuenta las limitaciones que el contexto actual nos presenta, hemos encontrado un límite de performance que nos gustaría superar. Planteando nuevas soluciones a los problemas encontrados en el desarrollo de *VISION* y sugiriendo mejoras en todos los aspectos que al mismo competen.

II. Partes de *VISION*

II-A. Microcontrolador

El microcontrolador implementado es un STM32F103C8T6. Es una placa desarrollo llamada Blue Pill, es de bajo costo e incorpora un núcleo RISC de 32 bits ARM® Cortex®-M3 de alto rendimiento, esta tarjeta es adecuada para iniciar proyectos sobre el microcontrolador STM32. Viene con los terminales para poder ser implementado con facilidad tanto en circuitos de prueba como en complejos proyectos.



(a) FRONT (b) BACK

Figura 1: Vistas de la placa Blue Pill.

II-A1. Especificaciones y características

- Tipo: Tarjeta STM32F103C Blue Pill.
- Modelo: STM32F103C.
- Dimensiones: 23mm x 53mm.
- Color PCB: Azul.
- Alimentación de voltaje:
  - Cualquier pin + 3.3V.
  - Cualquier pin + 5V.
  - Conector MicroUSB: +5V.

    - Voltaje Lógico IO: +3.3V.
- Procesador de núcleo ARM® Cortex®-M3.
- Software de programación:
    - STM32CubeProgrammer®.
    - STM32CubeIDE®.
    - KEIL DMK®.
    - SW4STM32®.
    - ARDUINO IDE®.
    - Mbed®.
- Firmware precargado: Bootloader para programar con STLINK.
- Tamaño de núcleo 32-bits.
- Entradas/salidas Digitales: 37.
- Canales PWM:12.
- Entradas ADC: 10 canales A/D de 12Bits.
- Conectividad CAN, I²C, IrDA, LIN, SPI, UART/USART, USB.
- Periféricos DMA, control de motor por PWM, PDR, POR, PVD, PWM, sensor de temp y WDT.
- Frecuencia de trabajo de 72MHz.
- Memoria flash: 64 KB.
- Memoria SRAM: 20KB.
- Temperatura de operación -40°C 85°C.

II-B. Motores

Este motor-reductor es un motor de CC de 6 V de alta potencia miniatura con escobillas de carbón de larga duración y una caja de cambios metálica de 9,96:1. Tiene una sección transversal de 10 × 12 mm, y el eje de salida de la caja de cambios en forma de D tiene 9 mm de largo y 3 mm de diámetro. Esta versión también tiene un eje de motor extendido de 4,5 × 1 mm para la posible incorporación de encoders de la marca Pololu®.

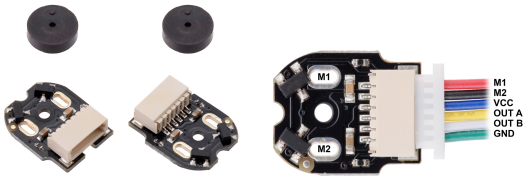
Figura 2: Moto reductor HPCB Pololu® 10:1 con eje extendido.

II-B1. Especificaciones y características

- Voltaje nominal: 6V.
- Tipo de motor: escobillas de carbón de alta potencia (HPCB).
- Corriente de bloqueo: 1.5A
- Corriente sin carga: 0.10A
- Velocidad sin carga (RPM): 3000
- Par de bloqueo extrapolad (kg.cm) :0.17
- Potencia máxima (W): 1.3
- Doble eje (caja de cambios y motor): 10: 1 HPCB 6V de doble eje.

II-C. Encoders

Los encoders utilizados son de cuadratura, magnético y de efecto Hall. Proporcionar 12 conteos por revolución del eje del motor. Los sensores funcionan de 2,7V a 18V y proporcionan salidas digitales que se pueden conectar directamente a un microcontrolador u otro circuito digital. Estos codificadores tienen un conector tipo JST macho de 6 pines de entrada lateral. Este módulo es compatible con todos los motor-reductores de micro-engranajes metálicos de doble eje de la marca Pololu®, incluidas las versiones HPCB.



(a) FRONT (b) BACK

Figura 3: Par de encoders magnéticos Pololu®.

II-C1. Especificaciones y características

- Dimensiones: 14,2mm × 11,6mm.
- Peso: 1,5g
- Voltaje mínimo de funcionamiento: 2,7V.
- Voltaje de funcionamiento máximo: 18V.
- Conector: entrada lateral, 6 pines macho, tipo JST SH.

II-D. Sensores Infrarrojos SHARP

Un sensor SHARP es un sensor óptico capaz de medir la distancia entre él y un objeto, para esto el sensor con la ayuda de un emisor infrarrojo y un receptor miden la distancia usando triangulación.

Figura 4: Sensor infrarrojo tipo SHARP.

El método de triangulación consiste en medir uno de los ángulos que forma el triángulo emisor-objeto-receptor, el Receptor es un PSD (Position Sensitive Detector) que detecta el punto de incidencia el cual depende del ángulo y a su vez de la distancia del objeto.

La geometría del sensor y de su óptica es el que limita el rango del sensor. El termino SHARP (Agudo) es porque tiene un rango de visión muy reducido, esto porque la luz que emite es puntual, lo que permite usar el sensor para escanear o mapear áreas, pero teniendo en cuenta que objetos pequeños serán difíciles de detectar.

También podemos usar varios sensores SAHRP para ampliar el rango de visión estos se pueden poner en diferente dirección e incluso en la misma dirección siempre y cuando las líneas de visión no queden muy cercanas.

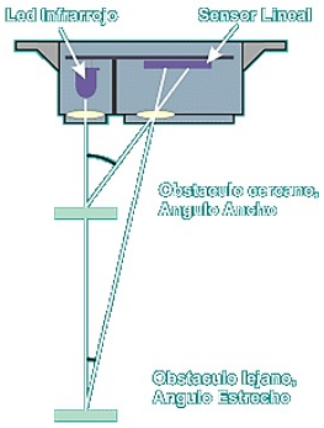


Figura 5: Representación gráfica de triangulación SHARP.

Una ventaja adicional es que no son sensibles a la luz ambiental o el Sol, enemigo de los sensores infrarrojos, un SHARP usa una luz infrarroja intermitente con una frecuencia determinada, que en el receptor es filtrada y elimina cualquier otra fuente de luz diferente a la frecuencia emitida.

Cabe destacar que este tipo de sensores no son óptimos para este proyecto debido a su gran zona ciega. Los mismos tienen una medición correcta partir de los 100mm, motivo por el cual se han tenido que recurrir a técnicas tanto de software como de hardware para subsanar este problema y poder utilizarlos.

II-E. Sensor óptico ADNS2620

El ADNS-2620 sensor óptico de mouse de computadora. Se utiliza para implementar un seguimiento no mecánico para mouses de computadora. Este sensor óptico se basa en la tecnología de navegación óptica, que mide los cambios de posición adquiriendo secuencialmente imágenes de superficie (fotogramas) y determina matemáticamente la dirección y magnitud del movimiento. El sensor está alojado en un doble escalonado de 8 pines tipo (DIP). Está diseñado para su uso con lente HDNS-2100, HLMP-ED80-xx000. No existen piezas móviles, por lo que no se requiere una alineación óptica de precisión, facilitando así el montaje. El formato de salida es un puerto serie de dos cables. Los cambios en la dirección X e Y están disponibles en los registros a los que se accede a través de el puerto serie. La resolución es de 400 recuentos por pulgada (cpp) con tasas de movimiento de hasta 12 pulgadas por segundo (ips).

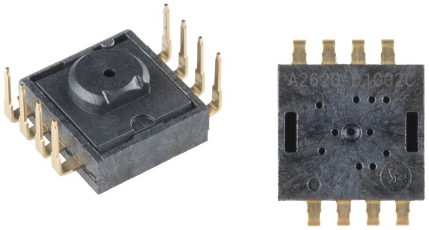


Figura 6: Sensors optico de mouse ADNS 2620

II-E1. Especificaciones y características

- Tecnología de navegación óptica precisa.
- Factor de Forma Pequeña (Espacio de 10 mm x 12,5 mm)
- Sin partes mecánicas móviles.
- Sensor de movimiento 2D completo.
- Interfaz común para controlador de propósito general.
- Navegación de superficie suave.
- Velocidad de cuadro programable hasta 3000 cuadros por seg (fps).
- Movimiento preciso hasta 12 ips.
- Resolución de 400 cpp.
- Alta fiabilidad.
- Detector de movimiento de alta velocidad.
- Soldable.
- Fuente de alimentación única de 5.0V.
- Cumple con las especificaciones del modo de suspensión USB.
- Modo de ahorro de energía durante tiempos sin movimiento.
- Registros de puerto serie.
  - Programación.
  - Transferencia de datos.
- Encapsulado en línea dual escalonado de 8 pines (DIP).

II-F. Driver de motores TB6612

El TB6612FNG es un controlador (driver) de motores que nos permite manejar dos motores de corriente continua, variando tanto la velocidad como el sentido de giro.

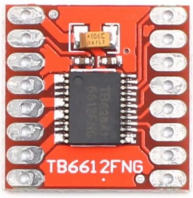


Figura 7: Driver TB6612.

Internamente está formado por dos puentes-H, junto con la electrónica necesaria para simplificar su uso. Los puentes-H están formados por transistores MOSFET que permiten que el TB6612FNG tenga mejor eficiencia y menores dimensiones que otros puentes H del mercado (por ejemplo el L298). El TB6612FNG también permite suministrar 1.2A por canal de forma continua, y 3.2A de pico.

II-F1. Especificaciones y características

- Tensión máxima de salida: 13.6V
- Tensión máxima para lógica de control: 5.5V
- Corriente máxima de salida pico (teórica): 3.2A
- Corriente máxima de salida continua (teórica): 1.2A
- Cantidad de motores controlables: 2 CC o 1 PAP.

II-G. Modulo Bluetooth HC-05

El HC-05 es un módulo Bluetooth pequeño y fácil de configurar. Se configura mediante comandos AT y tiene la

posibilidad de hacerlo funcionar tanto en modo maestro como esclavo.

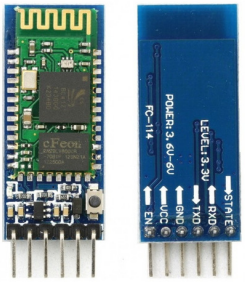


Figura 8: Modulo Bluetooth HC-05.

El módulo Bluetooth HC-05 puede alimentarse con una tensión de entre 3.3 y 6V (normalmente 5V), pero los pines TX y RX utilizan niveles de 3,3V. También dispone de un pulsador para entrar en modo comandos.

II-G1. Especificaciones y características

- Protocolo Bluetooth: v1.1/2.0.
- Frecuencia: banda ISM de 2,4GHz.
- Modulación: GFSK.
- Potencia de transmisión : menos de 4dBm, Clase 2.
- Sensibilidad: Menos de -84dBm en el 0,1.
- Ratio asíncronos: 2.1Mbps(Max)/160 kbps .
- Síncrono : 1Mbps/1Mbps.
- Perfiles de la ayuda : puerto serie Bluetooth (maestro y esclavo).
- Fuente de alimentación: +3.3VDC/50mA (soporta de 3.3 a 6V).
- Temperatura de trabajo: -5°C a 45°C.

II-H. Ruedas Pololu® 40×7mm

Estas ruedas de plástico rojo tienen neumáticos de silicona y miden 40 mm (1,57 pulg) de diámetro, y se ajustan a presión en los ejes D de 3 mm de los motores Pololu® antes mencionados.

Figura 9: Caption

II-H1. Especificaciones y características

- Dimensiones: 40 x 7 mm.
- Peso: 4.252 grs.
- Diámetro del eje: 3 mm2.
- Color: rojo.

II-I. Batería LiPo 2S

La batería de polímero de iones de litio, de ion de litio polímero o más comúnmente batería de polímero de litio (abreviadamente LiPo) son pilas recargables, compuestas generalmente de varias celdas idénticas en paralelo para aumentar la capacidad de la corriente de descarga.

En este proyecto se decidió usar este tipo de baterías debido a los cambios bruscos de velocidad que puedan llegar a surgir en las rectas del laberinto. La capacidad de descarga de este tipo de baterías es suficiente para la demanda del sistema que compone a *VISION*.

Figura 10: Bateria LiPo 2S.

II-I1. Especificaciones y características

- Capacidad: 300 mAh.
- Voltaje: 2S1P / 2 celdas / 7,4 V.
- Descarga: 35C constante / 75C pico.
- Peso: 31g (incluido cable, enchufe y caja).
- Dimensiones: 33 × 20,5 × 9,8 mm.
- Enchufe de descarga: JST-PH.

II-J. Chasis: PCB de fibra de vidrio

Para el chasis de *VISION* se utilizó el PCB en el que se montaron sus componentes. Se determino que la mejor manera de eliminar y controlar los pesos y el centro de masa, es utilizar el PCB de fibra de vidrio (por su alta resistencia mecánica) como chasis.

Figura 11: PCB de fibra de vidrio.

II-J1. Especificaciones y características

- Dimensiones: 100 x 100 x 3 mm.
- Revestimiento: cobre.
- Material: fibra de vidrio.

III. Partes extras del proyecto

III-A. Laberinto

Para la construcción del laberinto se intento minimizar los costos. La base se fabrico con MDF de 10mm y las paredes con MDF de 3mm cortado por laser. El tamaño se eligió para obtener la máxima flexibilidad y, al mismo tiempo, facilitar el transporte en un automóvil familiar típico. Además se

diseñaron, mediante la herramienta SketchUp®, piezas en 3D para luego ser impresas en 3D con material PLA. Estas piezas facilitan el encastre de las paredes del laberinto, de manera tal que se puedan cambiar las disposiciones de las mismas a gusto y en cualquier momento.

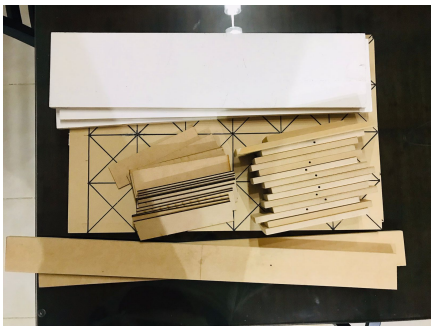


Figura 12: Construcción de la base del laberinto.

Una vez cortadas a medida exacta todas las maderas e impresas todas los acoples, el laberinto fue pintado de manera tal que las paredes sean blancas y la base negra. Esto se hace con la intención de contrastar bien las partes y disminuir el error de los sensores en la medición.

IV. Algoritmos

IV-A. Diagrama en bloques general

En la figura 13 se puede observar un diagrama en bloques del código de *VISION*. En él se pueden ver las distintas librerías que se crearon e incluyeron para el correcto funcionamiento. Más allá de que todas son necesarias y esenciales para cumplir los objetivos de *VISION*, cabe destacar las siguientes:

- GENERADOR DE PERFILES
- KALMAN FILTER
- ENCODER
- SENSORES
- MOTORES
- TIMER

En estas librerías se hizo mucho incapié a la hora de programarlas porque en ellas residen los detalles que hacen de *VISION* un micromouse competente. Sobre todo, la librería generadora de perfiles de aceleraciones, velocidades y posición, encargada de lograr un movimiento suave y fluido en las distintas partes del laberinto. Esta, en conjunto con la librería encargada de los encoders y motores son fundamentales para conseguir una performance de nivel considerable. La librería encargada de controlar el Filtro de Kalman es de mucha importancia a la hora de realizar seguimientos de trayectorias, en ella residen los modelos matemáticos que hacen posible, mediante la fusión de sensores, una correcta estimación de la posición relativa al punto de origen. Por ultimo, la librería que controla los timers, es vital para establecer intervalos de tiempo en dónde se ejecutarán procesos tan importantes como el controlador PID, la iteración del Filtro de Kalman, la actualización de sensores, la comunicación con el sensor óptico, la lectura de los encoders, etc.

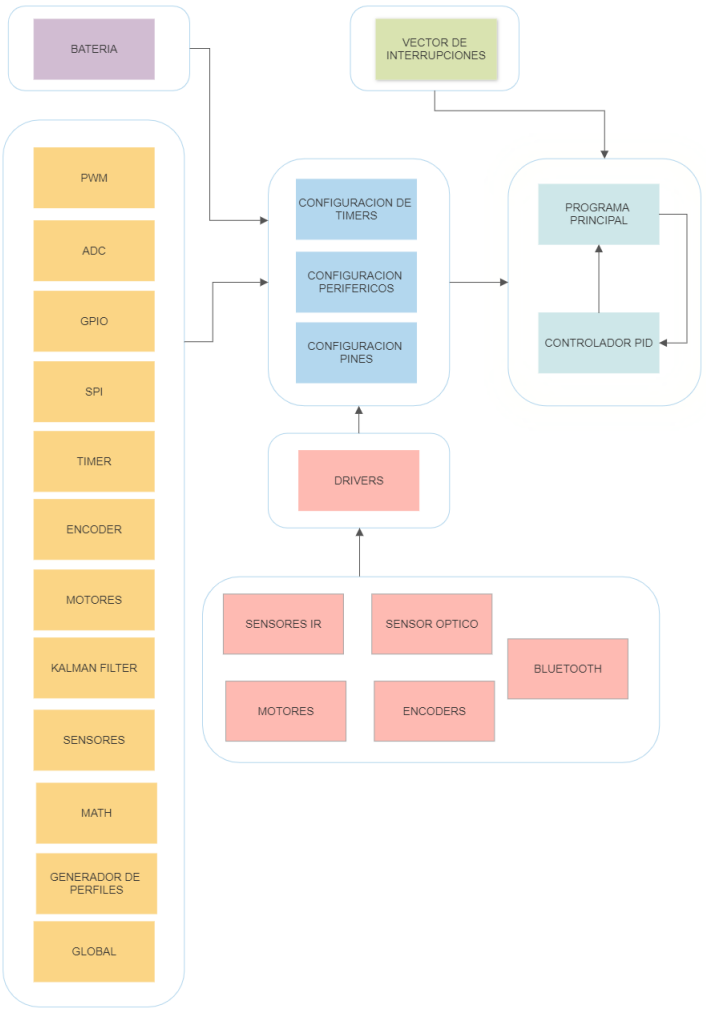


Figura 13: Diagrama en bloques del código de *VISION*

IV-B. Perfiles de aceleraciones y velocidades

Los algoritmos de perfiles de aceleraciones y velocidades son bloques de códigos que generan curvas que se aplican a los PIDS (que determinan el PWM adecuado) de los motores para producir movimientos de forma fluida. Esto quiere decir que se tiene en cuenta el comportamiento no lineal de los motores y su reacción frente a estímulos de alta frecuencia. Para ver esto gráficamente veamos la figura 14 Como se puede observar en la primer imagen de la figura

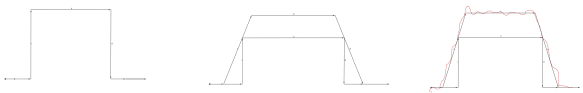


Figura 14: Comparación entre velocidad ideal, velocidad generada por perfil y velocidad real.

14, un comportamiento ideal seria que el motor cambie instantáneamente su estado de reposo o movimiento, a un nuevo estado deseado. Esto no es posible debido a que la pendiente de la recta tangente en el punto de cambio tiende a infinito. Una manera de subsanar este problema es el de realizar pequeños incrementos de velocidad (regidos por el paso de algún timer) desde el punto inicial al punto objetivo, de manera tal que el robot pueda seguir esos

pequeños cambios y no realizar acciones bruscas que puedan desestabilizarlo o, peor aun, dañar las pastillas de carbono de los motores y sobre-calentarlos innecesariamente. Algo muy importante a destacar, es que esta acción debe garantizar que las áreas encerradas por ambas curvas sean iguales, de manera tal que la distancia recorrida sea idéntica. Esto deriva de saber que la posición es la integral de la velocidad.

$$x(t) = \int_0^t v_x(t)dt \tag{1}$$

En otras palabras, para garantizar que el comportamiento ideal y real sea el mismo, el área de las curvas debe ser igual.

En la tercer imagen de la figura 14 se puede ver el comportamiento real del robot. Es claro como, mediante el controlador PID, el robot trata de seguir la curva de manera aproximada. Esto permite movimiento fluidos y un mejor rendimiento de la batería ya que no se realizan acciones correctivas que impliquen descargas significativas a la misma.

Este perfil se nombró como *perfil de rectas* y genera las condiciones necesarias para trazar curvas que hagan *fluir* al robot en las rectas. Adicionalmente se creo otro perfil con iguales características para realizar los giros. El justificativo y principio de funcionamiento es el mismo que el de las rectas, solo que en vez de ingresarle datos con unidades de distancia (milímetros), se trabaja con unidades angulares (ángulos y radianes). El mismo no se explicara pará no ser redundante.

IV-C. Controlador PID

El controlador PID esta ajustado para poder seguir la referencia proveniente de los perfiles generadores de velocidades y aceleraciones. Previo a esto se obtuvo la función de transferencia de los motores haciendo uso de la herramienta *IDENT* de MATLAB®. De esta manera se trabaja sobre un modelo que aproxima lo suficiente a los motores como para generar movimientos rápidos y precisos.

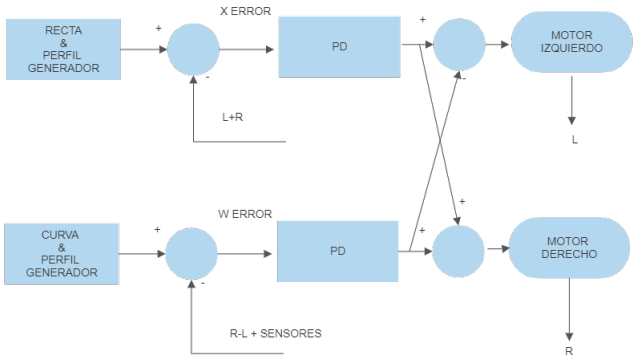


Figura 15: Diagrama en bloque del controlador PID en conjunto con los perfiles generadores de velocidades.

Como se puede observar en la figura 15, ambos perfiles (rectas y rotación) están conectados en el bloque PID. Esto se hace con el propósito de que el robot pueda moverse en linea recta manteniendo un ángulo de rotación constante y evitar desviarnos de la trayectoria. De todas maneras hay que tener en cuenta los detalles de construcción que hacen que el robot no sea perfectamente simétrico, como también la diferencia de diámetros de las ruedas propios de la construcción. Estos detalles se consideraron al momento de realizar los modelados y fueron ajustados empíricamente. Por estos resultados empíricos podemos asegurar que el comportamiento es suficientemente bueno como para completar el laberinto construido. A su vez, en la linea encargada de ajustar el ángulo de rotación de *VISION*, se cuenta con la entrada de los sensores infrarrojos que apuntan hacia los laterales del robot, con la intención de mantenerlo siempre centrado y equidistante de las paredes disponibles en el momento de realizar los movimientos. Un dato importante es que ambos errores plasmados en la figura 15 son del tipo integral. Es decir que se acumulan con el paso del tiempo. Esto permite que el sistema de control tenga noción de lo que sucedió instantes antes y pueda accionar en base a ello.

En el siguiente extracto de código se puede apreciar la lógica de control de centrado dependiendo de cuantas paredes hayan.

```
if (left_wall_presence && right_wall_presence){
error = right_error - left_error;
}else if (right_wall_presence){
  error = 2*right_error;
} else if (left_wall_presence){
    error = -2*left_error;
}
```

Como se puede interpretar, se hace uso de las paredes presentes con la siguiente lógica y jerarquia:

- Si hay dos paredes, utilizo las dos para centrar.
- Si esta solo la derecha uso la derecha.
- Si esta solo la izquierda uso la izquierda.

En el caso de que *VISION* se encuentre frente a tres paredes (celda que obliga una vuelta de 180°), utiliza las mismas para realizar un proceso de alineación. De manera tal que luego de realizar un giro de 180°, retrocede hasta hacer contacto con la pared de la celda y paralelizar su parte trasera con la misma. Esto le permite continuar la búsqueda desde un nuevo punto relativo de partida conocido.

IV-D. Algoritmos de búsqueda

Para realizar el mapeo del laberinto o ”búsqueda de la meta”, se diseñaron dos algoritmos. Ambos son usados y conocidos por la mayoría de los competidores a nivel internacional, debido a su simpleza y fácil implementación.

IV-D1. Algoritmo de la mano derecha

Este algoritmo es muy sencillo y únicamente requiere hacer una cosa: Girar a la derecha siempre que se pueda. De esta manera, teniendo siempre una pared a la derecha y recorriendo toda su superficie, se avanza hasta la salida.

IV-D2. Algoritmo de la mano izquierda

Este algoritmo, al igual que el anterior, se basa en realizar giros siempre en la misma dirección siempre que se pueda, pero en este caso girando a la izquierda.

Cabe destacar que el uso de estos algoritmos no garantiza resultados positivos en laberintos que presenten islas, es decir, paredes sin conexión a los bordes del laberinto. Utilizarlos en laberintos de esas características llevaría a quedar dando vueltas alrededor de un circuito cerrado infinitamente.

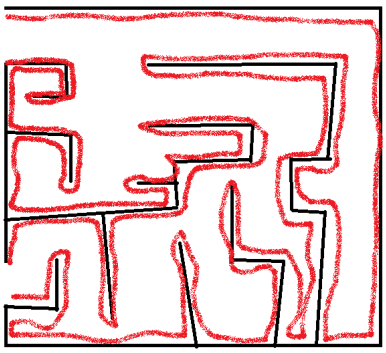


Figura 16: Ejemplo de la aplicación de los algoritmos de búsqueda.

Un ejemplo de la implementación de estos algoritmos puede verse en la figura 16

VISION utiliza estos algoritmos para realizar el mapeo correspondiente, pero debido a las limitaciones que estos presentan, se esta trabajando en la implementación de algoritmos tipo Flood Fill para realizar mapeos en laberintos mas completos.

IV-E. Algoritmo de optimización

Para esta etapa ya se consta con el laberinto mapeado y guardado en la memoria interna de *VISION*, por lo que se prosigue a procesarlo y, mediante un algoritmo basado en *Floo Fill*, encontrar el camino mas corto desde el punto de origen hasta la salida. Para el ejemplo de la figura 16, el resultado de optimizar la trayectoria seria el marcado en color verde en la figura 17.

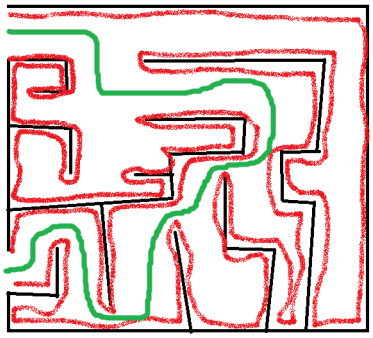


Figura 17: Ejemplo de aplicación del algoritmo de optimización.

La implementación del algoritmo *Flood Fill* para determinar el camino mas corto consiste en asignar un valor a cada casilla del laberinto, este valor es correspondiente con la distancia de esta casilla al punto de partida. Cuando todas tienen un peso asignado , se realiza el recorrido inverso desde la meta hasta la partida siguiendo siempre el camino de casillas con el numero mas bajo. De esta forma, se obtiene el camino mas corto.

IV-F. Algoritmo de Speed Run

Una vez optimizado el laberinto y determinado el camino mas corto hacia la meta, se analizan los resultado y se procede a determinar una *ruta de viaje* para que el robot, en combinación con las etapas antes explicadas, realice a una velocidad pre-seteada los movimientos de traslación y rotación necesarios para llegar hasta la meta en el menor tiempo posible.

V. Construcción de *VISION*

Para la construcción de *VISION* se tuvieron en cuenta aspectos fundamentales para el correcto desempeño dentro del laberinto tales como:

V-1. Ubicación y ángulo de los sensores infrarrojos

Esto fue esencial para que el robot pueda tener una correcta medición sobre *dónde* esta posicionado y *hacia dónde* debe ir. Es decir que los sensores fueron colocados de manera tal que se pudiera tener noción de la distancia entre las paredes y el robot dentro de una casilla, pero a su vez deben tener un cierto ángulo (que depende de las dimensiones del laberinto) para poder *predecir* lo que tendrá que hacer en la próxima casilla. Es decir, si deberá girar de manera suave hacia los lados, dar media vuelta o puede acelerar porque sigue una linea recta. De esta manera se logra anticipar movimientos y alargar los tiempos de toma de decisión del robot. Para esto se necesitó un *tuneo* del robot que involucró todas las variables a tener en cuenta: dimensiones del laberinto, radios de giro, velocidades, aceleraciones y distancias respectivas requeridas, resolución de encoders, radio de las ruedas, etc. Una idea de esto se puede ver en la figura 18.

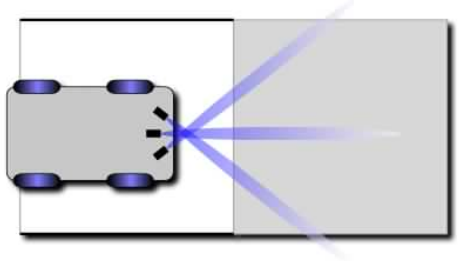


Figura 18: Ängulo de los sensores para anticipar casillas.

V-2. Dimensiones de la placa PCB (chasis)

Esto fue de primordial importancia debido a que un robot demasiado grande para un laberinto de 200 x 200 mm seria imposible de maniobrar y por el contrario, un robot demasiado chico tendría muchas limitaciones de diseño innecesarias. Las dimensiones de *VISION* alcanzan los 100 x 100 x 50 mm. En la figura 19 se puede ver una captura de pantalla del programa de diseño de PCB.

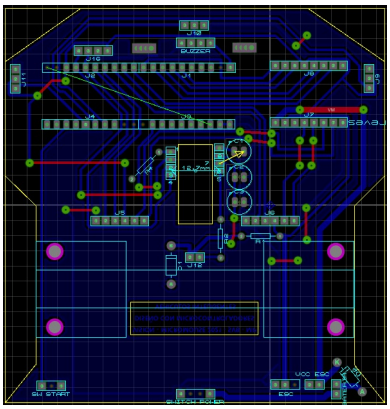


Figura 19: Captura de pantalla de PCB en etapa de diseño

V-3. Ubicación de los motores y eje de tracción

En este aspecto se tuvo en cuenta los momentos en los que el robot hará correcciones y aceleraciones. Un robot con el eje muy adelantado seria un robot muy nervioso y por ende poco estable. Por lo tanto en *VISION* se diseñó el chasis de manera que el eje deje las ruedas lo mas atrás posible, siempre conservando los 100 x 100 mm de la base.

V-4. Radio de giro según ubicación y tamaño de ruedas

El radio de giro es el que determina la maniobrabilidad, sobre todo en las esquinas. Se diseñó en base a la ubicación del eje de tracción, con la intención de hacer un robot capaz de realizar giros de 180° en casillas con 3 paredes, de una forma fluida sin depender mucho la cercanía a la mismas.

V-5. Peso de cada uno de los componentes

En este ítem nos vimos limitados por la disponibilidad de hardware. Los componentes utilizados no fueron los óptimos para realizar un diseño que priorice el peso. Los sensores, los componentes no *SMD*, las partes impresas en 3d y los cables de los sensores, son aspectos negativos en cuanto al peso de *VISION*.

V-6. Zona ciega de los sensores infrarrojos

Como ya se repitió muchas veces en este informe, los sensores tipo *SHARP* utilizados no son los adecuados para este tipo de proyectos. Los mismos presentan una zona ciega que imposibilita realizar mediciones correctas dentro de los 100mm desde el sensor. Por lo tanto se tuvo que tener especial cuidado de que el robot no entre en ninguna de estas -tres- zonas ciegas para no *"perderse"* y colisionar con alguna de las paredes o simplemente perder el rumbo. Este aspecto se subsanó adecuando el lazo de control de manera tal que el robot no supere los 10° [grados] de desviación respecto de la linea recta que atraviesa las casillas del laberinto. De esta manera *VISION* siempre se encuentra en una posición que permite un cierto *"margen de error"* y hace posible la navegación entre celdas.

V-7. Modulo Bluetooth

Este modulo nos dio la posibilidad de realizar captura de datos en tiempo real, siendo capaz de comunicar a *VISION* con cualquier computadora, emulando un puerto *COM*. Por medio de MATLAB pudimos crear una interfaz gráfica que nos permitiese visualizar los movimientos en forma tridimensional y teniendo en consideración características físicas reales tales como la velocidad punto a punto, la disposición física de los componentes y la dimensión de VISION. Para lograr esto, utilizamos la aplicación integrada de MATLAB *Driving Scenario Design* que forma parte del módulo de *Automotive* para la simulación de diversos vehículos inteligentes. Con está aplicación podemos lograr ver como se va formando el laberinto y el recorrido que hace *VISION* en cada punto en tiempo real como se muestra en la Fig.20 A su vez utilizamos la funcion de *bird eye plot*

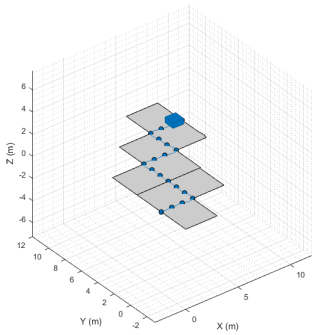


Figura 20: Visualización del recorrido que realiza *VISION*

que nos permite gráficar la información recolectada por los sensores respecto de la posición de las paredes como se puede observar en la Fig.21 Queda implementar una interfaz

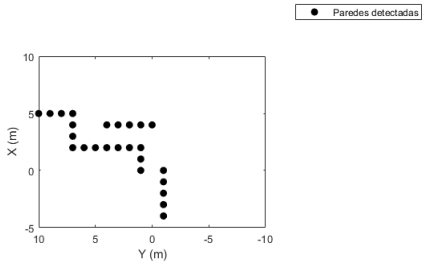


Figura 21: Visualización de las paredes detectadas del laberinto

que nos permita visualizar las correciones PID en tiempo real, además de mejorar la calidad de la disposición de datos para el usuario.

V-8. Sensor Óptico

Para no extendernos mas páginas, para este ítem se adjunta otro informe realizado por los integrantes de este proyecto que explica en detalle lo vinculado al sensor *ADNS2620* y su respectiva implementación mediante el algoritmo de estimación del filtro de *Kalman*.

VI. Mejoras a futuro

Por ultimo y para terminar este informe, se mencionarán las mejoras que se considerarán en el próximo diseño de *VISION*, que ya está en proceso de diseño en base a los conocimientos adquiridos en este proyecto y a las pertinentes investigaciones realizadas. Consideramos que tenemos mucho que aprender sobre los líderes en las competencia de robot micromouse: los Japoneses.

- Diseño de PCB mecanizado (no artesanal).
- Sensores IR emisores TSAL6100.
- Fotoreceptores LPT2021.

- Motores FaulHaber.
- Encoders FaulHaber de 4000CPR.
- STM con núcleo CORTEX-M4.
- Ruedas de caucho de silicona P53 (mas adherencia).
- 4 sensores (dos para alineación frontal y dos para alineación lateral).
- Fabricación completa con componentes SMD.
- IMU MPU6050.
- Fuente de corriente para sensores.
- Control de sensores mediante ráfaga.
- Mejora de los perfiles de aceleraciones.
- Implementación de algoritmo FloodFill.
- Disminución de la altura.
- Diseño más aerodinámico.
- Eliminación de cables (solo batería, motores y encoders)
- Reemplazar tornillos metálicos por tornillos plásticos.

Todo esto es con la intención de poder seguir desarrollando robots competentes cada vez mas complejos, haciendo énfasis en aspectos tan importantes como la física, la mecánica, el diseño y la programación del robot. En conjunto estos aspectos, creemos, determinan la performance de cualquier robot de estas características. También, la intención de este proyecto es motivar a aquellos futuros estudiantes de la materia a que apliquen sus conocimientos a la construcción de robots de competencia ya sean, seguidores de linea, robots sumo o micromouse. Creemos que los desafíos que conlleva la construcción de cualquiera de estos robots engloba áreas que no son desarrolladas en nuestra carrera y son esenciales para comprender el funcionamiento de un sistema en la vida real.

VII. Fotos de *VISION*

Se adjuntan algunas fotos de distintos ángulos de *VISION*. Las fotos de las distintas iteraciones y construcciones previas a *VISION* se adjuntaran en la carpeta de campo correspondiente al proyecto, con la intención de no alargar más este documento. Como se espera que este informe sea tratado de manera virtual, se adjunta una lista de reproducción que se irá actualizando en base a las mejoras o cambios del robot. También se adjunta una lista de reproducción de otros robots desarrollados los integrantes de este proyecto.

- VISION - Robot MicroMouse basado en STM32.
- LISA - Robot Seguidor de Linea Argentino basado en Arduino®.
- VOLTA - Robot Seguidor de Linea Argentino basado en PIC16F887.

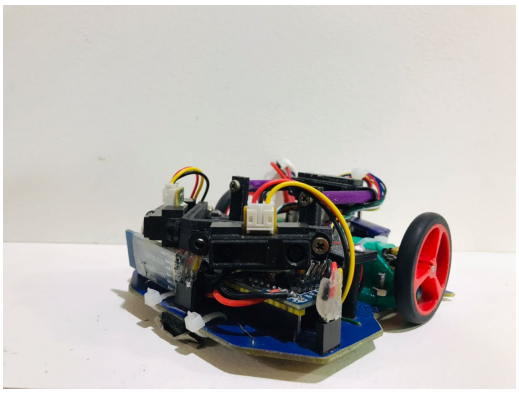


Figura 22: Vista diagonal *VISION*

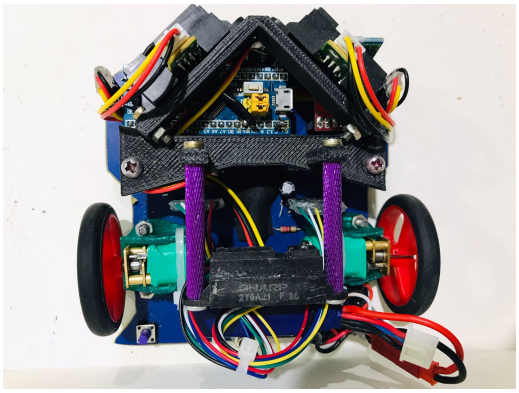


Figura 23: Vista superior *VISION*

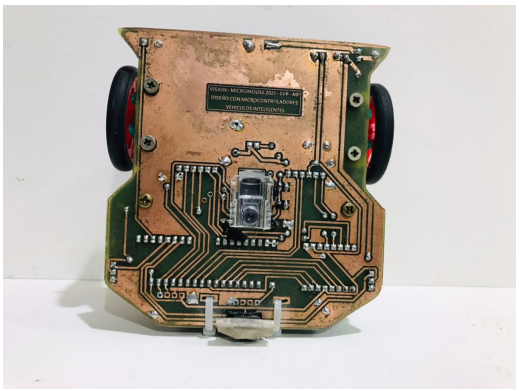


Figura 24: Vista inferior *VISION*

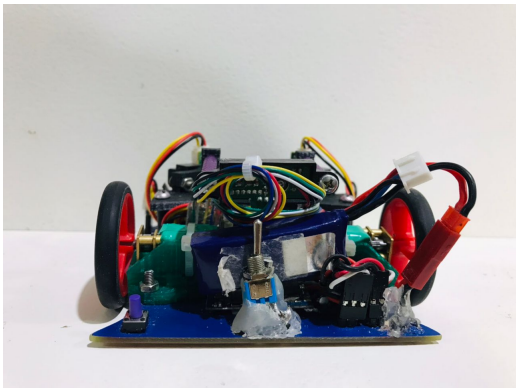


Figura 25: Vista trasera *VISION*