ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Robot Pololu 3Pi con varios sensores a distancia para evitar colisiones"

TESINA DE SEMINARIO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN EN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentado por:

César Andrés Espinoza Estrada

Jhonny Javier Barzola Iza

GUAYAQUIL - ECUADOR
2011

AGRADECIMIENTO

A Dios y a la Santísima Virgen María por haberme permitido seguir en este mundo y poder ver realizada esta anhelada meta.

A mis padres por haberme brindado su apoyo incondicional durante toda mi vida; en especial en mi carrera académica, por darme la oportunidad de terminar mi formación profesional y por creer en mí en todo momento. Agradecimiento especial a Msc. Carlos Valdivieso, Director de Seminario de Graduación por su orientación e invaluable ayuda.

César Andrés Espinoza Estrada

DEDICATORIA

A ti Dios por haberme bendecido con una familia maravillosa, la cual ha estado pendiente de mi bienestar todos los mis días de mi vida.

Con mucho cariño dedico este trabajo principalmente a mis padres Julio Espinoza y Dolores Estrada por haberme dado la vida, su amor y afecto, por haberme guiado por el camino del saber. A mis abuelitos Rafael Estrada y Celeste Espinosa por sus palabras sabias que sirvieron para mi formación humana. A mis hermanos Roxana Espinoza y Julio Espinoza por haberme dado ánimos y palabras de aliento. A mi cuñado Christian Romero, a mis tíos, primos y amigos por estar conmigo durante todo el camino y regalarme un sano consejo en el momento que lo necesité.

Quiero agradecer especialmente a Denisse Cárdenas por su comprensión y apoyo incondicional, por estar en los momentos difíciles de mi vida y darme fuerzas para

seguir adelante. Te Amo. Sólo Dios sabe cuán grande es lo que siento por ti.

A todos y cada uno de ustedes les quiero dedicar éste trabajo con mucho cariño como muestra de mi agradecimiento por todo lo que significan para mí.

César Andrés Espinoza Estrada

AGRADECIMIENTO

A Dios, a quien mantengo firmemente en mis pensamientos y en mi corazón. Al ser más importante que tengo y le dedico cada momento. Al cual obsequio palabras de amor con prudencia, respeto y temor. El que me ha permitido una vida llena de salud, gracia y al que debo todo lo que tengo por su gran misericordia.

A mi familia de todo corazón, que Dios los bendiga en cada momento y los mantenga en su gracia.

A los Ingenieros, maestros, compañeros y amigos.

Jhonny Barzola

DEDICATORIA

A Dios por sobre todas las cosas, por quién es y será cada instante de mi ser.

A mis padres, quienes han sido mi apoyo constante regalándome a diario soporte y amor.

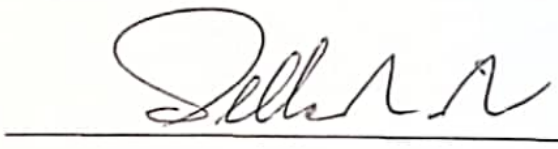
A todos mis amigos y compañeros quienes siempre confiaron en mí y me ayudaron en momentos difíciles.

Jhonny Barzola

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Enrique Valdivieso A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN



Ing. Hugo Villavicencio V.

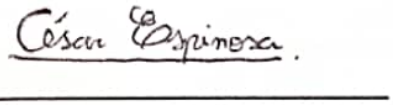
DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL).

Jhonny Javier Barzola Iza



César Andrés Espinoza Estrada

RESUMEN

La finalidad del proyecto es desarrollar un robot Pololu 3Pi con varios sensores ópticos implementados en este con el fin de evitar colisiones en el medio al momento que el robot esté circulando sobre alguna pista.

Al presionar el botón de encendido del robot Pololu 3Pi, éste mostrará mediante su pantalla LCD mensajes de bienvenida, tiempo durante el cual el robot se encuentra también preparándose para el recorrido. Una vez hecho esto, el robot mostrará nuevos mensajes indicando la distancia medida por cada sensor óptico así como la velocidad establecida para lograr esquivar los obstáculos y con esto evitar una colisión cuando esté recorriendo la pista. Cuando se encuentre con algún obstáculo en su camino que no ha logrado evadir o que fue ubicado directamente muy cerca, se detendrá completamente y realizará un giro hasta encontrar una salida por donde seguir su recorrido.

Una de las principales aplicaciones de este proyecto es la medición de distancia entre dos objetos estáticos o en movimiento, así como la resolución de laberintos.

Las herramientas a utilizar para este proyecto son:

El robot Pololu 3pi cuyo microcontrolador marca atmel atmega328p programaremos con el USR AVR programmer. También usaremos los sensores ópticos Sharp para conocer la distancia y para realizar el código utilizaremos el programa AVR Studio 4 con la previa instalación de la librería del Pololu 3Pi. Que nos servirá de ayuda para aplicar funciones más directas.

Se realizarán pruebas para establecer las velocidades y distancias a las cuales el robot Pololu 3Pi deberá realizar acciones de evasión y búsqueda de salida.

INDICE GENERAL

Pág.

CAPITULO 3

CAPITULO 4

INDICE DE FIGURAS

Pág.

INDICE DE TABLAS

Pág.

CAPÍTULO 1

1. DESRIPCIÓN DEL PROBLEMA

1.1. ANTECEDENTES

Desde hace mucho tiempo tenemos la necesidad de lidiar con el inconveniente de las conocidas colisiones ya sea entre personas en alguna puerta o corredor, así como el problema del tránsito vehicular y en los últimos años a maquinarias y robots móviles. Para esto ha sido necesario desarrollar por medio de la experiencia y el análisis de la naturaleza, dispositivos que se adapten a las necesidades para que sirvan de guía. Un ejemplo de esto es el sentido de la visión en los animales. Al observar un obstáculo en su recorrido realizan algún tipo de acción ante este evento.

También es necesario tomar en cuenta el tipo de respuesta a realizar. Un ejemplo en el mundo biológico se presenta cuando un ciervo camina por un bosque porque cuando este llega a un gran árbol, debe rodearlo. ¿Por qué lado? Después de unos momentos de observación, el ciervo elige una dirección al azar y continúa.

El azar también se puede utilizar para hacer descubrimientos. Considere una hormiga que se dirige hacia una fuente conocida de comida. Cuando las hormigas avanzan, su trayecto no es absolutamente rectilíneo: se mueven hacia un lado y otro en base a algún tipo de generación azarosa en el control de sus patas. Acercándose a su destino, la hormiga descubre una

segunda fuente de alimento, que estaba a un lado del camino. Su movimiento "azaroso" le resultó útil. Nosotros también podemos emplear valores azarosos en las respuestas de nuestros sentidos biomiméticos para mejorar las rutinas de escape y búsqueda de los robots caseros.

Uno de los detectores que más comúnmente se instalan en un robot es algún tipo de parachoques. El parachoques de un robot puede ser la única capacidad sensorial. Pero resultan necesarios aún teniendo otros métodos de detección dependiendo la aplicación, pues constituyen algo así como una última línea de defensa. Cuando los demás sistemas no han detectado un obstáculo, para que el robot no reciba golpes en partes delicadas de su estructura, o se quede haciendo fuerza con sus motores y gastando sus baterías en una trampa mecánica, es bueno que posea el más primitivo pero seguro y eficaz sistema de detección.

1.2. FUNCIONAMIENTO BÁSICO

El sistema de sensores le indica al robot que hay un objeto en su recorrido, con forma y tamaño sin determinar. Ante el suceso, la lógica de control puede ordenar una maniobra que le permita salir de la situación o una señal indicadora de lo ocurrido.

Aunque muchos robots experimentales básicos se construyen con una cantidad mínima de sensores en la parte delantera, y a veces uno en la parte trasera, lo que permite una navegación primitiva, una disposición así no será suficiente para lograr que un robot se mueva por los ambientes de una casa sin quedar atrapado en algún sitio.

1.3. LÓGICA CONVENCIONAL PARA EVITAR COLISIONES

En una implementación básica, lo más típico es que la lógica funcione con las siguientes reacciones:

- Si detecta un obstáculo a la izquierda, detener la marcha y girar un poco a la derecha, luego continuar;
- Si detecta un obstáculo a la derecha, detener la marcha y girar un poco a la izquierda, luego continuar.

Esto funciona, pero lamentablemente también es una buena fórmula para garantizar que un robot quede atrapado muy pronto en un rincón de una casa, o incluso en el recorrido más controlado de una zona de pruebas.

1.4. LA CANTIDAD DE SENSORES A UTILIZAR

Es obvio que agregar un anillo continuo de sensores alrededor de la periferia del robot puede ser algo excesivo, un desperdicio de recursos. Los experimentos en ambientes caseros normales han determinado que cinco sensores en el frente y uno en la parte posterior ofrecen un funcionamiento óptimo dentro de lo que es una red sensorial algo limitada. Con el que se logra un campo de visión muy completo para satisfacer los diferentes ángulos en los que se detecte un obstáculo.

La configuración de dos sensores en la parte delantera del robot ubicados ligeramente laterales para ampliar el campo de visión es muy usada para el propósito que requerimos ya que nos ofrece la capacidad de análisis hacia la izquierda y derecha para evitar las colisiones en su recorrido y por medio de programación la posibilidad de realizar rutinas de movimientos complejos

para resolver las distintas circunstancias en la que podría llegar a encontrarse el robot.

1.5. PROYECTOS SIMILARES

1.5.1. Robot con sensores mecánicos de choques

El contacto en un parachoques le indica al robot que hay un objeto en su recorrido, con forma y tamaño sin determinar. Ante el suceso, la lógica de control puede ordenar una maniobra que le permita salir de la situación. Aunque se debe tomar en cuenta que mientras menos parachoques se encuentren conectados, menor será la habilidad del robot para enfrentar algún tipo de obstáculo.

FIGURA 1.1 Sensores mecánicos de choque

Navegar con parachoques es como moverse palpando en la oscuridad: alcanzar un destino puede llevarle al robot un tiempo excesivamente largo, si es que alguna vez llega. Se entiende que es el método más básico para implementar la detección de obstáculos en un robot.

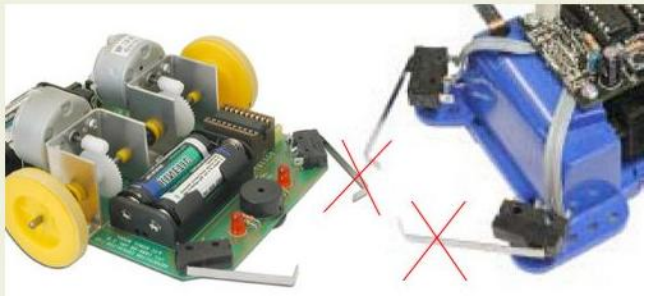


FIGURA 1.2 Robots con sensores mecánicos de choque

Hay otros detalles menores de la construcción de parachoques que harán que éstos sean óptimos para detectar obstáculos, tales como agregar unos rectángulos de goma que suavicen los impactos y dividir el cilindro del parachoques en secciones para aportar una información sensorial más precisa.

1.5.2. Seguidor de distancia

De la misma forma tenemos muchas posibilidades si variamos la distancia a la que ha detectado un obstáculo y a la que el robot ejecute una respuesta. Además del tipo de respuesta que emita, viene a ser desde un simple led indicador hasta una posible compleja resolución con muchas opciones de respuesta rápida o giros y almacenamientos de datos.

1.5.3. Seguidor de línea

Otra aplicación es el seguidor de línea, que se aplica bastante como ejercicio para aprendizaje de la programación para la cual se puede utilizar desde tiras de papel blanco hasta cinta de colores brillantes ó negra adhesiva como camino para que el robot siga.

FIGURA 1.3 Robot Pololu 3Pi

1.5.4. Resuelve laberintos

Uno de las aplicaciones más conocidas con el Pololu 3Pi es la resolución de laberintos, puesto que por medio de la programación logramos obtener una enorme cantidad de maneras de resolverlo. Como el de la posibilidad de memorizarlo para ir directamente hacia la llegada en un posible segundo recorrido.

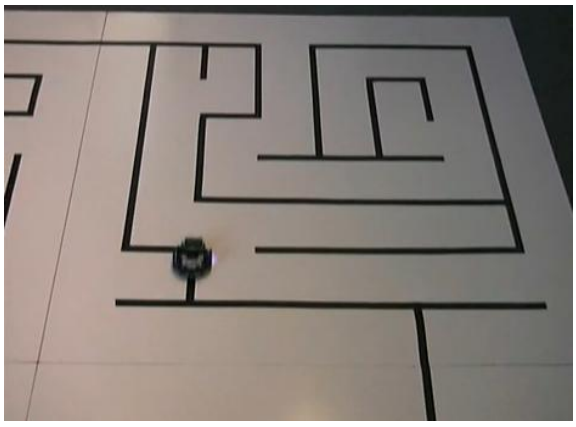


FIGURA 1.4 Robot Pololu 3Pi resolviendo laberinto con seguidor de línea negra

CAPÍTULO 2

2. FUNDAMENTOS TEORICOS

2.1. POLOLU 3Pi

El nombre del robot viene del tamaño; el diámetro en centímetros es el valor de pi multiplicado 3 veces o 3pi, en pulgadas es alrededor de 3,7 " de diámetro. El 3pi es un robot autónomo con motores duales, cinco sensores de reflectancia, una pantalla LCD de 8 × 2, un zumbador, 4 pilas AAA, y tres botones de usuario. Todos conectados a un microcontrolador ATMega328 programable. Capaz de alcanzar velocidades superiores a 3 pies por segundo, el 3pi es un gran robot primero para los principiantes ambiciosos y segundo un robot perfecto para aquellos que buscan pasar de los robots para principiantes no programables o más lentos. El 3pi está diseñado principalmente como un seguidor de línea, así que esto no es una gran preocupación.

El software libre y la cadena de herramientas GNU GCC proporcionan un compilador maduro, de gran alcance que puede hacer casi cualquier cosa que se necesite. Para hacer las cosas aún más interesantes, el controlador es compatible con el microcontrolador Arduino y todo su software.

2.2. FUNCIONAMIENTO DEL POLOLU

2.2.1. Baterías

La potencia del sistema del 3pi empieza con las baterías, por eso es importante conocer cómo trabajan las baterías. La batería contiene unos elementos químicos que reaccionan moviendo electrones desde el positivo (+) al terminal negativo (-). El tipo más conocido es la pila alcalina compuesta de zinc y manganeso en una solución de hidróxido potásico. Cuando se descargan completamente deben ir al reciclado ☺. Para el 3pi recomendamos las baterías de níquel-manganeso (NiMH) que pueden recargarse una y otra vez. Estas baterías realizan una reacción diferente a las alcalinas y no es aquí donde vamos a explicarlo; lo importante es conocer cómo podemos saber su estado (medición) con unos simples números. Lo primero a conocer es que la cantidad de electrones que se mueven de un terminal a otro se mide en Voltios (V) o diferencia de potencial. En las baterías de NiMH es de 1,2V. Para entender la fuerza de la batería es necesario conocer cuántos electrones circulan por segundo esto es intensidad que se mide en amperios (A) Una corriente de 1A equivale a 6x1018 electrones saltando por segundo. Un amperio es la fuerza que un motor de tamaño mediano puede necesitar y sería la corriente que necesitan dar las pequeñas pilas AAA.

Para cualquier batería en funcionamiento, el voltaje suministrado se reduce con el tiempo bajando hasta perder toda la energía (procurar no llegar a ello, puede producir cortocircuito y quedar inutilizada para siempre). El

gráfico siguiente muestra un modelo de cómo debe la tensión aumentar la potencia requerida:

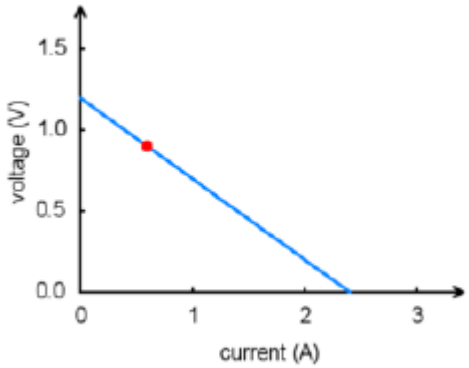


FIGURA 2.1 Curva de corriente - voltaje del nivel de batería del robot Pololu 3Pi

La potencia de una batería se mide multiplicando los voltios por los amperios, dando una medida en vatios (W=VxA).

Por ejemplo, en el punto marcado en el gráfico, tenemos una tensión de 0,9 V y una corriente de 0,6 A, esto significa que la potencia de salida es de 0,54 W. Si desea más es necesario agregar más baterías, y hay dos maneras de hacerlo: juntarlas en paralelo o en serie. En paralelo se juntan todos los terminales positivos por un lado y todos los negativos por otro, la potencia se suma (A1+A2+...) pero la tensión es la misma. Cuando las conectamos en serie, terminal positivo de una con terminal negativo de la otra se suma la tensión (V1+V2+...). De cualquier manera, la máxima potencia de salida se multiplicará con el número de baterías.

En la práctica, sólo se conectan las baterías en serie. Esto se debe a que aun siendo del mismo tipo las baterías, no todas tienen la misma carga y conectándolas en serie la corriente se compensa entre ellas, Si queremos que duren más podemos usar pilas más grandes que el AAA como por ejemplo las AA, C, y baterías tipo D con el mismo voltaje pero con más fuerza.

El total de energía de la batería está limitado por la reacción química; cuando deja de reaccionar la fuerza se para. Este es un gráfico entre voltaje y tiempo:

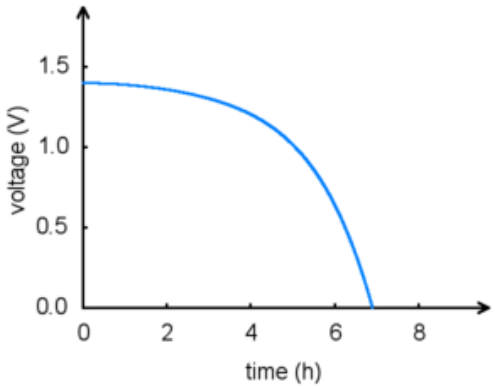


FIGURA 2.2 Curva de voltaje - tiempo del Pololu 3Pi

La cantidad de energía de las baterías está marcada en la misma como miliamperios/hora (mAH). Si estás usando en el circuito que consume 200mA (0,2 A) durante 3 horas, una batería de 650 mAH necesitará una recarga transcurrido este tiempo. Si el circuito consume 600 mA en una hora quedará descargada (¡NO! descargues del todo la batería, podría quedar inutilizada).

2.2.2. Gestión de la energía

El voltaje de la batería se reduce con el uso, pero los componentes eléctricos usados precisan de una voltaje controlado. Un componente llamado *regulador de voltaje* ayuda a que este voltaje se mantenga constante. Por mucho tiempo los 5V regulados han sido para los dispositivos electrónicos digitales llamados de nivel TTL. El microcontrolador y muchas partes del circuito operan a 5V y su regulación es esencial. Hay dos tipos de reguladores de voltaje:

Lineales.- Los reguladores lineales utilizan un circuito de retroalimentación simple para variar la cantidad de energía que pasa a través de cómo y cuánto se descarga. El regulador de tensión lineal produce una disminución del valor de entrada a un valor determinado de salida y el resto de potencial se pierde. Este despilfarro es mayor cuando hay gran diferencia de voltaje entre la entrada y la salida. Por ejemplo, unas baterías de 15 V reguladas para obtener un valor de 5 V con un regulador lineal perderán dos tercios de su energía. Esta pérdida se transforma en calor y como consecuencia es necesario utilizar disipadores que lo general no funcionan bien si los utilizamos con aplicaciones de alta potencia.

Switching.- Este tipo de reguladores alternan la tensión on/off a una frecuencia generalmente alta y filtrando el valor de la salida, esto produce una gran estabilidad en el voltaje que hemos deseado. Es evidente que este tipo de reguladores son más eficientes que los lineales y por ello se utilizan especialmente para aplicaciones con corrientes altas en donde la precisión es importante y hay varios cambios de voltaje. También pueden

convertir y regular voltajes bajos y convertirlos en altos. La clave del regulador de switching está en el inductor que es el que almacena la energía y la va soltando suavemente; en el 3pi el inductor es el chip que se encuentra cerca de la bola marcado como “100”.
En los PC se usan esos inductores que son como unos donuts negros con espiras de cable de cobre.

El sistema de potencia del 3pi corresponde al siguiente diagrama:

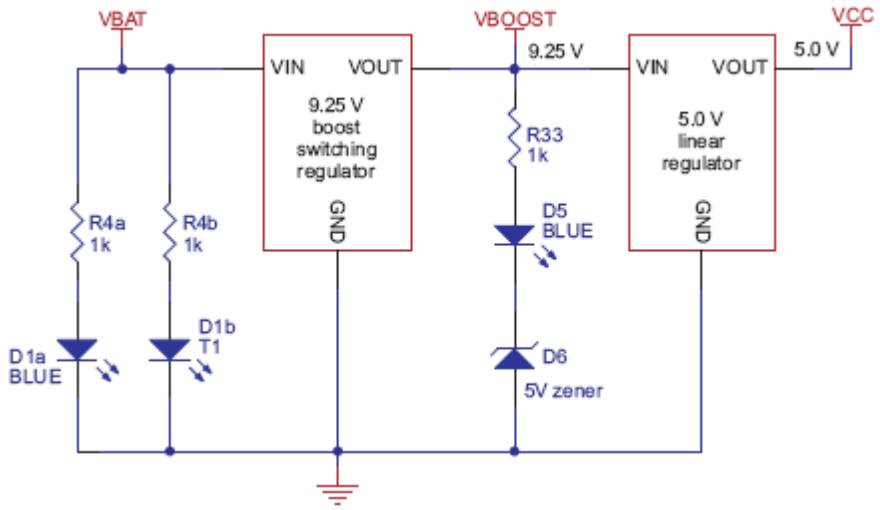


FIGURA 2.3 Esquemático de la fuente de alimentación del Pololu 3Pi

El voltaje de 4xAAA pilas puede variar entre 3,5 a 5,5 voltios (y hasta 6v si se usan alcalinas). Esto no podría ir bien si no fuera por la regulación del voltaje a 5V. Usamos un regulador switching para elevar el voltaje a 9,25 V (Vboost) y reguladores lineales para obtener 5V (VCC). Vboost sirve para los motores y los leds sensores IR en línea, mientras que el VCC es para el microcontrolador y las señales digitales. Usando el Vboost para los motores

y sensores obtenemos tres ventajas sobre los típicos robots que trabajan con baterías directamente:

- Primero, el voltaje alto se reserva para los motores.
- Segundo, mientras el voltaje está regulado, los motores trabajan a la misma velocidad aun cuando las baterías oscilen entre 3,5 y 5,5 voltios. Esto tiene la ventaja de que al programar el 3pi, puedes calibrar los giros de 90º varias veces, a pesar de la cantidad de tiempo que esto lleva consigo.
- Tercero, a 9,25 V los cinco led IR conectados en serie, consumen una cantidad más pequeña de energía (Puedes alternar la conexión/desconexión de los IR para ahorrar energía).

Una cosa interesante acerca de este sistema de energía es que en lugar de agotarse progresivamente como la mayoría de los robots, el 3pi funcionará a máximo rendimiento, hasta que de repente se para. Esto puede sorprender, pero al mismo tiempo podría servir para monitorizar la tensión de la batería e indicar la recarga de las baterías.

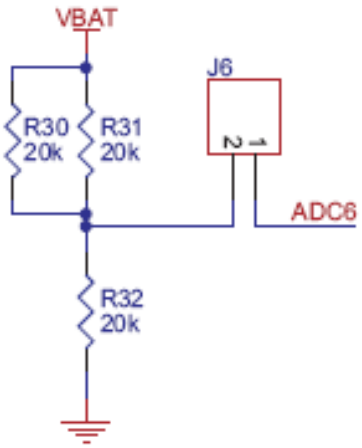


FIGURA 2.4 Esquemático del divisor para la medición del nivel de voltaje del Pololu 3Pi

Un circuito simple de monitorización de la batería se encuentra en el 3pi. Tres resistencias como muestra el circuito comportan un divisor de tensión de 2/3 el voltaje de las baterías. Este conectado a una entrada analógica del microcontrolador y mediante la programación produce que por ejemplo: a 4,8 V el pin ADC6 tenga un nivel de 3,2V.

2.2.3. Motores y engranajes.

El motor es una máquina que convierte la energía en tracción. Hay diferentes tipos de motores pero el más importante para robótica es el motor DC de escobillas y que usamos en el 3pi.

FIGURA 2.5 Tipo de motor utilizado en el Pololu 3Pi

El típico motor DC contiene imanes permanentes en el exterior y bobinas electromagnéticas montanas en el eje del motor. Las escobillas son piezas deslizantes que suministran corriente desde una parte del bobinado a la otra produciendo una serie de pulsos magnéticos que permiten que el eje gire en la misma dirección.

El primer valor usado en los motores es la velocidad representada en rpm (revoluciones por minuto) y el par de fuerza medio en kg-cm o en oz-in (onzas – pulgadas). Las unidades de par muestran la dependencia entre fuerza y distancia. Multiplicando el par y la velocidad (medidos al mismo tiempo) encuentras la potencia desarrollada por el motor. Cada motor tiene una velocidad máxima (sin resistencia aplicada) y un par máximo (cuando el motor está completamente parado). Llamamos a esto, funcionamiento a velocidad libre y al par de parada.

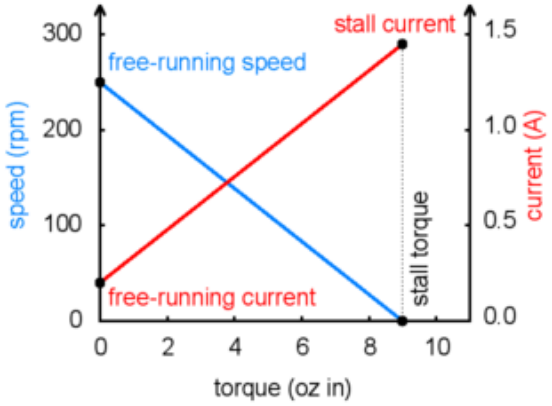


FIGURA 2.6 Curva de torque - velocidad de los motores del Pololu 3Pi

Naturalmente, el motor usa el minimo de corriente cuando no se aplica una fuerza, y si la corriente que viene de la batería aumenta es por fuerzas de rodamiento o engranajes de modo que son parámetros importantes del motor. Según se muestra en el gráfico adjunto.

La velocidad libre de rodamiento de un pequeño motor DC es de varios miles de revoluciones por minuto (rpm) muy alta para el desplazamiento del robot, por lo que un dispositivo de engranajes permite reducir estas revoluciones y aumentar el par, la fuerza rodamiento. El ratio de engranaje es de 30:1 en el 3pi, es decir 30 vueltas de motor, una vuelta de rueda. Estos parámetros están representados en la tabla siguiente:

Par máximo:	oz·in
Velocidad libre:	700 rpm
Consumo min:	60 mA
Par máximo:	60 oz-in
Consumo máx.:	6 540 mA

TABLA 2.1 Características del funcionamiento de los motores del Pololu 3Pi

2.3. COMPONENTES DEL DISEÑO

2.3.1. Sensor óptico SHARP GP2Y0A21YK0F

El GP2Y0A21YK0F es un sensor de medida de distancia compuesto por una combinación integrada de PSD (Detector Sensible a la Posición), IRED (Diodo Emisor Infrarrojo) y la señal del circuito en proceso.

La variedad de la reflectividad del objeto, la temperatura del ambiente y la duración del funcionamiento no son influenciados fácilmente a la distancia de detección debido a la adopción del método de triangulación. Este dispositivo envía el voltaje correspondiente a la distancia de detección, así que este sensor también puede ser utilizado como un sensor de proximidad.

Ejemplo de las características de las mediciones de distancia.

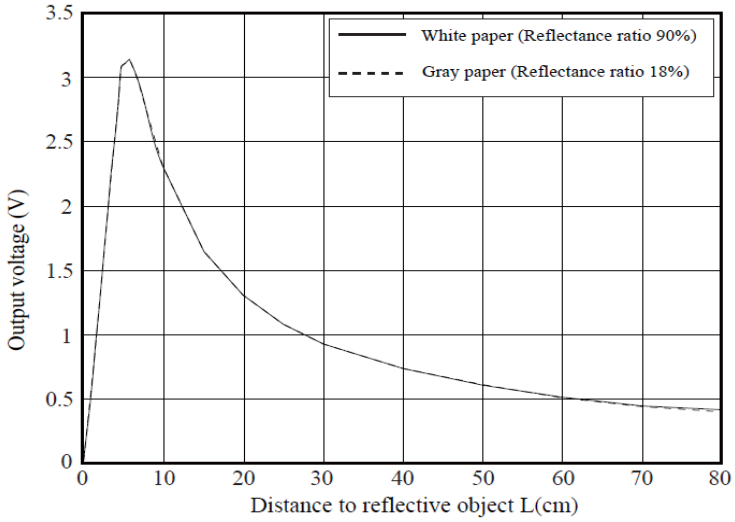


FIGURA 2.7 Curva de voltaje - distancia de los sensores ópticos

Se puede apreciar que hay una relación inversa entre el voltaje y la distancia del objeto encontrado. Mientras menor sea la distancia, mayor será el voltaje entregado por el sensor y viceversa.

2.4. HERRAMIENTAS DEL DISEÑO

2.4.1. Pololu 3Pi USB AVR programmer PGM03A.

Este dispositivo es un programador de controladores AVR-base, tales como los controladores de robots Orangután y el robot 3pi. El programador emula un AVRISP v2 en un puerto serie virtual, por lo que es compatible con el software estándar de programación AVR. Dos características adicionales ayudan a generar y depurar proyectos: un puerto serie TTL-nivel para la comunicación de uso general y un SLO-ámbito de aplicación para el

monitoreo de señales y los niveles de tensión. Un cable USB y el cable ISP están incluidos.

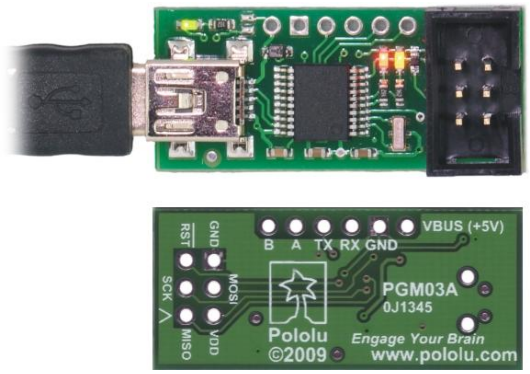


FIGURA 2.8 Programador Pololu 3Pi USB AVR

El programador AVR Pololu USB es un muy compacto, de bajo coste en el programador del sistema (ISP) para mirocontroladores AVR de Atmel, lo que hace más versátil este dispositivo, es que es una solución de programación atractiva para los controladores AVR-base como los controladores de robots Pololu orangután. El programador AVR USB se conecta al puerto USB de su ordenador a través de un USB incluido a un cable mini-B y se comunica con el software de programación, tales como AVR Studio o AVRDUDE, a través de un puerto COM virtual utilizando el protocolo AVRISPV2/STK500. El programador se conecta al dispositivo de destino a través de un cable de programación incluido de 6 pines ISP (el mayor, de 10 clavijas conexiones ISP no son compatibles directamente, pero es fácil de crear o comprar una de 6 pines a adaptador de 10 pines ISP).

CAPÍTULO 3

3. DISEÑO DE LA SOLUCIÓN

3.1. FUNCIONAMIENTO BÁSICO

El sistema de sensores le indica al robot que hay un objeto en su recorrido, con forma y tamaño sin determinar. Ante el suceso, la lógica de control puede ordenar una maniobra que le permita salir de la situación.

Aunque muchos robots experimentales básicos se construyen con una cantidad mínima de sensores en la parte delantera, y a veces uno en la parte trasera, lo que permite una navegación primitiva, una disposición así no será suficiente para lograr que un robot se mueva por los ambientes de una casa sin quedar atrapado en algún sitio.

3.2. DIAGRAMA DE BLOQUES

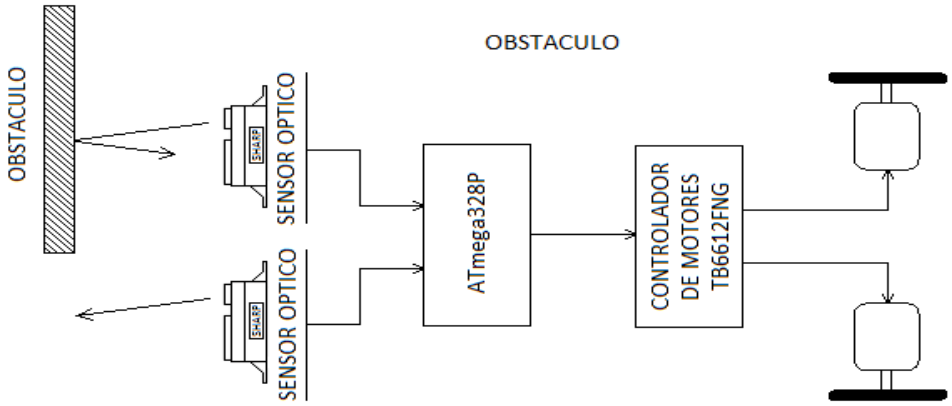


FIGURA 3.1 Diagrama de bloques del proyecto

3.3. TÉCNICAS PARA EVITAR COLISIONES

En una implementación básica, lo más típico es que la lógica funcione con las siguientes reacciones:

- Si detecta un obstáculo a la izquierda, detener la marcha y girar un poco a la derecha, luego continuar.
- Si detecta un obstáculo a la derecha, detener la marcha y girar un poco a la izquierda, luego continuar.

Esto funciona, pero lamentablemente también es una buena fórmula para garantizar que un robot quede atrapado muy pronto en un rincón de una casa, o incluso en el recorrido más controlado de una zona de pruebas.

3.3.1. PRIMERAS DIFICULTADES CON LOS SENSORES

En primer lugar, tan pronto como el robot encuentre una esquina con paredes en ángulo podrá quedar atrapado en un movimiento sin fin de ida y vuelta, hasta que se le acaben las baterías. A esto se le llama "encajonarse", y hay ciertas soluciones que se pueden utilizar para corregir la condición. En segundo lugar, en áreas estrechas o cerradas, el robot puede quedar atrapado entre dos paredes muy juntas, rebotando entre las paredes por horas, sin ir a ningún lado. Éste es el "problema del túnel".

Si no se ha implementado una manera de saber si la parte posterior va a chocar con algo, el robot hará fuerza con sus motores para intentar retroceder. Finalmente, existiendo sensores traseros, cuando luego de una

detección el robot retrocede una distancia apreciable, también puede quedar atrapado, rebotando sin cesar.

3.4. SELECCIÓN DE SENSORES A UTILIZAR

Es obvio que agregar un anillo continuo de sensores alrededor de la periferia del robot puede ser algo excesivo, un desperdicio de recursos. Los experimentos en ambientes caseros normales han determinado que cinco sensores en el frente y uno en la parte posterior ofrecen un funcionamiento óptimo dentro de lo que es una red sensorial algo limitada.

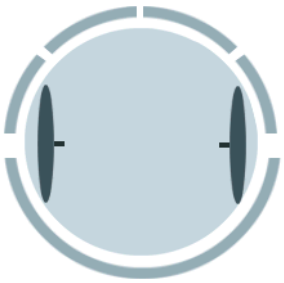


FIGURA 3.2 Configuración de varios sensores

El funcionamiento de estos topes cubrirá las siguientes detecciones:

- 1. Frente izquierdo izquierdo
- 2. Frente izquierdo
- 3. Centro
- 4. Frente derecho
- 5. Frente derecho derecho
- 6. Parte posterior

En nuestro caso utilizaremos la configuración de dos sensores en la parte delantera del robot ubicados ligeramente laterales para ampliar el campo de visión que tendrá para evitar las colisiones en su recorrido.

Antes de que definamos qué comportamiento se debe producir cuando se activa cada sensor, recordemos que hay tres tipos de situaciones que debemos considerar. Primero, un comportamiento simplemente reactivo o de reflejos. Se trata de una lógica simple de lecturas que activa un sistema preestablecido de maniobras de escape, que dependerán de sensor se haya activado.

A estas maniobras se les llama "balísticas", porque una vez que comienzan, continúan hasta el final, por lo general sin admitir nuevas entradas sensoriales. Un ejemplo sería: "Si se activa el sensor derecho, primero retrocedemos medio segundo, damos vuelta a la izquierda un segundo, y luego continuamos avanzando". Es decir, el comportamiento es totalmente reactivo, con cantidades realmente insignificantes de inteligencia implicadas.

En segundo lugar, y un poco más avanzado, se pueden analizar una serie de posibles choques que se repiten y están relacionados de manera cronológica. Esta secuencia se presenta, por ejemplo, cuando el robot queda atrapado en una situación de "encajonado", y también cuando el robot queda en una reiteración hacia adelante y hacia atrás de la que no puede escapar.

Si empleamos cierto nivel de inteligencia de supervisión, detectaríamos estos sucesos que se repiten en el tiempo y reaccionaríamos en consecuencia. Por ejemplo, si descubrimos que se producen media docena

de choques sucesivos y repetitivos en unos segundos, esto puede indicarnos que estamos en una trampa del tipo "encajonado".

En tercer lugar, agregamos un factor de variación. En "inteligencia" robótica, agregar variación es, simplemente, introducir una pequeña cantidad de valores azarosos a la inteligencia estándar de reflejos de la máquina.

Esto tiene peso cuando se toma una decisión lógica con el procesador de un robot o, de manera similar, en la estructura neural de un insecto. Una pequeña cantidad de "variación al azar" puede aportar nuevas soluciones a los problemas que se presentan, e incluso puede producir comportamientos inesperados. Para lograr esto, se debe agregar aleatoriedad matemática a nuestras reacciones.

3.5. RESPUESTAS BASÍLICAS Y POR REFLEJO

3.5.1. Ubicación de sensores

Debido a la curvatura del cuerpo, para abarcar un mayor campo de visión del robot tenemos sensores en la parte frontal ubicados en un ángulo de unos 30 a 40 grados respecto al eje de la dirección de avance del robot, de modo que el ajuste de dirección del robot, en caso de producirse este tipo de obstáculo, debe estar en un valor de ángulo similar al de esta posición. A un movimiento de este tipo se la llama "reacción proporcional".

Cuando uno de estos parachoques hace contacto con un obstáculo, se retrocede hasta que quede liberado desactivado el sensor y después se gira unos 45 grados en dirección opuesta a la del impacto.

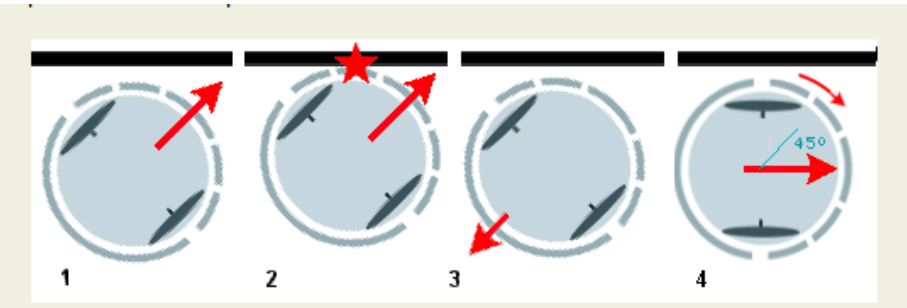


FIGURA 3.3 Solución en base a la ubicación de los sensores y el giro

Una vez más, un valor de azar agregado al giro nos asegurará que el robot no se quede en una repetición hacia adelante y hacia atrás en situaciones en las que el espacio para el movimiento es ajustado. El giro, entonces, es de 22 grados, y se le agrega una cantidad al azar de entre 0 y 23 grados para sumar un total de 45 grados de rotación.

3.5.2. Respuesta en base al tiempo entre los sucesos

Además de estas respuestas puramente reflejas, hay posibilidad de establecer respuestas a secuencias de sucesos que se producen dentro de un plazo determinado. Por ejemplo, si se detectan media docena de choques dentro de una cantidad limitada de segundos, podríamos asegurar que el robot está atrapado en un espacio confinado. Observando el tiempo

que transcurre entre choques, es posible detectar estos episodios y realizar maniobras evasivas.

3.5.3. Impactos múltiples muy seguidos

Hay por lo menos dos tipos de secuencias de impactos por encierro o entrampado del robot, y será necesario analizarlas para obtener información de lo que está sucediendo. Las dos producen impactos múltiples en poco tiempo y no siempre es fácil determinar cuál es cual. La respuesta a cada una de estas situaciones es algo diferente.

3.5.4. "Encajonamiento" (atrapado en una esquina)

Esta clase de sucesión de choques es la más común que se presenta al probar un nuevo robot. En este caso, el robot quedará atrapado en una esquina de una habitación, o en partes de un mobiliario, rebotando una y otra vez entre paredes adyacentes. Por ejemplo, cuando un robot detecta con su sensor izquierdo, retrocede un poco y gira hacia la derecha para evitarla. Pero ahora avanza y un segundo después detecta la pared adyacente con su sensor derecho, retrocede nuevamente un poco y gira hacia la izquierda, en un movimiento evasivo reflejo. Esto puede seguir para siempre, o hasta que el robot se quede sin baterías.

FIGURA 3.4 Robot enfrentando un encierro en una esquina

La detección y corrección de esta condición es de gran importancia si se desea que el robot se mueva en un entorno casero. Para solucionar esta situación, primero debemos detectarla. Esto se hace registrando en la memoria cuáles fueron los últimos cuatro toques y el tiempo que transcurrió entre ellos. Luego se examina esta información para determinar si se está produciendo un encajonamiento.

Cuando detectamos un obstáculo en un sensor, respondemos con una acción usual, pero guardamos el valor que alcanzó el contador, y luego ponemos el contador en cero. Luego, en cada impacto sucesivo vamos guardando los tiempos entre cada contacto en distintas variables, hasta haber registrado los últimos cuatro obstáculos.

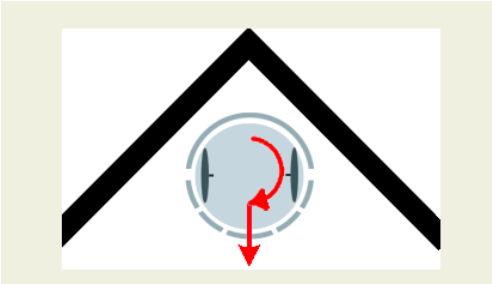


FIGURA 3.5 Solución ante encierro en una esquina

Cuando tenemos los cuatro valores, los sumamos, y si la suma de los tiempos que transcurrieron nos da un valor menor a 5 segundos, esto significa que hemos detectado una secuencia de choques que se han producido en un corto tiempo. En la mayoría de los casos, para escapar de esta condición se hace que el robot gire un ángulo de 180°.

3.5.5. El problema del túnel

Sin la ayuda de un sistema de detección de obstáculos por infrarrojos o por ultrasonidos (sonar) que nos permita "ver" por qué lado podemos escapar, la situación de encierro en un túnel o estrecho pasillo sólo se puede solucionar probando reiteradas veces. Imagínese que su robot ingresa a un pasillo angosto o en el espacio entre el respaldo de un sofá y la pared. En algún lugar, si la dirección de marcha del robot no es totalmente paralela a las paredes, éste comenzará a chocar con ellas. Lo que ocurra a partir de aquí depende mucho de la forma del robot, además de del espacio que tiene, pero lo más probable es que el robot empiece a rebotar de pared a pared, muy rápido, en una maniobra de evasión tras otra, y quede atrapado.

FIGURA 3.6 Problema del túnel en el los robots

Dado que sólo puede detectar objetos cuando toca con ellos, no es fácil encontrar qué debe hacer el robot para liberarse de este problema y salir del túnel. Muchos robots caseros pueden moverse horas dentro de una casa, y cuando se produce el ingreso a un túnel, el robot queda atrapado allí hasta que se le agotan las baterías. Parece un situación muy parecida a la del encajonamiento en una esquina, pero hay diferencias a estudiar y una solución muy diferente.

Para detectar la situación de entrampado en un túnel, debemos registrar, además del tiempo transcurrido entre las activaciones del sensor, la cantidad de veces que hemos debido pasar a la reacción de escape de una esquina, y el tiempo total en que se produjeron. Después de dos o tres acciones de evasión de esquina que han ocurrido muy seguidas, es muy probable que en realidad estemos atrapados entre dos paredes cercanas, y debamos entrar en nuestra rutina de escape del túnel.

Si bien no hay ningún método absolutamente seguro, el que vamos a describir puede sacar al robot de esta situación de trampa sin salida. La respuesta es entrar en una rutina de seguimiento de pared, una condición que, bien instrumentada, debería llevarnos a alguna salida.

En esta rutina de seguimiento de pared, cada impacto debe ser seguido de un movimiento de evasión minimizado. Es decir, se ejecuta la serie de movimientos habituales ante un obstáculo, pero se retrocede muy poco y sólo se gira un pequeño ángulo. Esto debería hacer que el robot siga las paredes del corredor en un movimiento de pequeños arcos, pegado a ellas, sin llegar hasta la pared opuesta. Se sale del modo de seguimiento de pared cuando el o los sensores vinculados (los del lado de esa pared)

dejan de detectar por un tiempo, lo que indicaría que hemos alcanzado el final del túnel.

El truco principal para seguir una pared es que los sensores sobresalgan lateralmente lo suficiente para que un sensor del robot en dirección casi paralela a una pared seguro los actúe. El tamaño de los arcos recorridos se determina en base al ángulo de regreso al contacto con la pared. Si el arco es muy amplio, habrá problemas en una pared curvada; si es muy pequeño, se perderá mucha energía rebotando contra las paredes. La amplitud del arco depende mucho del tamaño del robot, por supuesto. Para un robot pequeño (de unos 20 centímetros de longitud, por ejemplo), será apropiado un arco de unos 20 a 30 centímetros.

Y aquí es cuando descubrimos que es necesario otro control. Hay que limitar el tiempo en que el robot estará en el modo de seguimiento de pared. Una cantidad determinada de segundos, y luego salir de él, porque podría ocurrir que el robot se quede girando para siempre alrededor de algún mueble redondeado.

3.5.6. Atascos de todo tipo

Otro problema es que el robot quede atascado en algún objeto que no ha detectado, o en el cable de un electrodoméstico, o en un desnivel del piso, con sus ruedas girando, y ante la falta de detecciones de choque suponga que se está moviendo libremente por una enorme habitación.

3.6. FALSAS DETECCIONES

Si queremos un robot independiente, hay que tener en cuenta las fallas que se pueden producir en los sensores. Agreguemos también que los detectores también pueden fallar por sí mismos, internamente. Sea como sea, si un sensor queda activado permanentemente, el robot va a responder constantemente, moviéndose en un baile en círculos para tratar de escapar de un obstáculo fantasma.

De la misma manera en caso de no detectar los obstáculos es difícil de detectar. Se podría observar a lo largo de un buen tiempo si algún sensor jamás ha detectado un obstáculo. Si esto afecta, lo más probable es que el robot se detenga contra el obstáculo y quede con las ruedas girando sin cesar. Esta situación se detecta usando la lógica de detección de atascos.

3.7. DIAGRAMA DE FLUJO

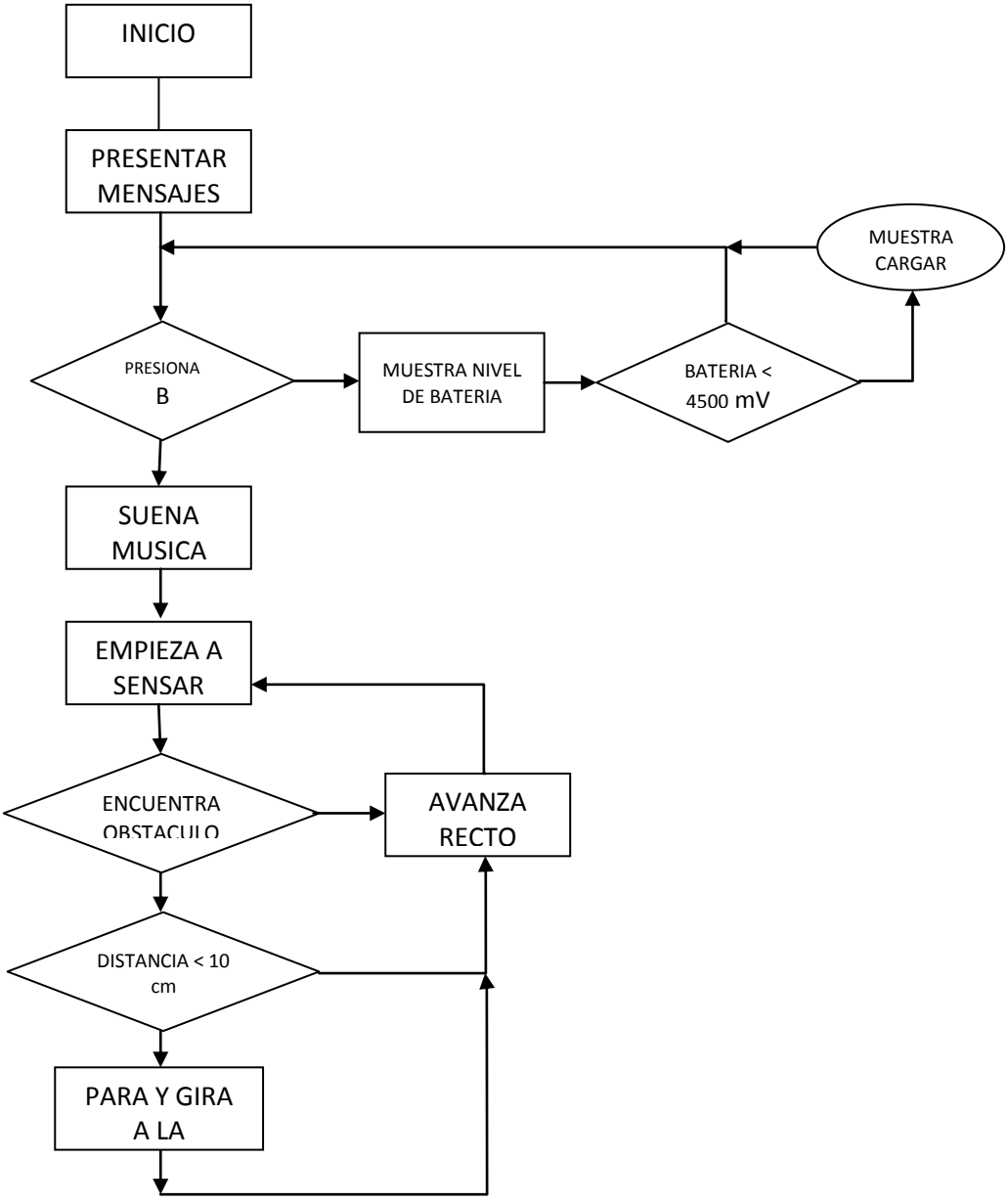


FIGURA 3.7 Diagrama de flujo

3.8. CÓDIGO

A continuación se muestra el código hecho en lenguaje C con el cual logramos configurar el robot pololu para realizar lo antes dicho en el diagrama de flujo.

```
/*
        Grupo # 6
Integrantes:
- César Espinoza
- Jhonny Barzola
*  Descripción: Robot Pololu 3pi evasor de obstaculos
*  Este programa utiliza dos sensores SHARP GP2Y0A21YK0F  conectados
*  en ADC7 (izquierda) y ADC5 (derecha) y que le permite esquivar obstáculos
*/
//   Includes
#include <pololu/3pi.h>        // especifico 3pi
#include <avr/pgmspace.h>      // variables en memoria de programa

// Mensajes de introducción guardados en memoria de programa
const char linea1[] PROGMEM = " GRUPO ";
```

```
const char linea2[] PROGMEM = "    6 ";

const char linea3[] PROGMEM = " CESAR ";

const char linea4[] PROGMEM = "ESPINOZA";

const char linea5[] PROGMEM = " JHONNY ";

const char linea6[] PROGMEM = "BARZOLA";


// Tonos musicales guardados en memoria de programa

const char tono1[] PROGMEM = ">g32>>c32";

const char tono2[] PROGMEM = "L16 cdegreg4";


// Variables generales

unsigned int sens_der;   // sensor derecha

unsigned int sens_izq;   // sensor izquierda

int motor_der;           // motor derecha

int motor_izq;           // motor izquierda


void iniciar(){

  emitters_off();
```

```
//Muestra el nombre del grupo e integrantes
print_from_program_space(linea1);
lcd_goto_xy(0,1);
print_from_program_space(linea2);
play_from_program_space(tono1);
delay_ms(1000);
      clear();
      print_from_program_space(linea3);
lcd_goto_xy(0,1);
print_from_program_space(linea4);
      delay_ms(1000);
      clear();
      print_from_program_space(linea5);
lcd_goto_xy(0,1);
print_from_program_space(linea6);
      delay_ms(1000);
//Muestra el voltaje de la batería y espera botón
while(!button_is_pressed(BUTTON_B))
```

```
    {
        clear();
        print_long(read_battery_millivolts());  // usa ADC6 para sensar la batería del circuito
        print("mV");
        lcd_goto_xy(0,1);
      if (read_battery_millivolts()<4800){
         print (" !CARGAR¡");
         red_led(1);
      }
      else
           print("Pulsa B");
        delay_ms(100);
    }
    // Espera que se presione el botón B para empezar a moverse
    wait_for_button_release(BUTTON_B);
    clear();
    // Toca música y espera a que termine para empezar
    play_from_program_space(tono2);
```

```
    while(is_playing());
}

void lee_sensores(){
   // Mira a la derecha si hay obstáculos
   if (!analog_is_converting())
   sens_der = analog_read(5);
   // Mira a la izquierda si hay obstáculos
   if (!analog_is_converting())
   sens_izq = analog_read(7);
}

void busca_salida(){
   clear();
   lcd_goto_xy(0,0);
   print("Buscando");       // Busca salida
   red_led(1);              // Encienda leds rojo y verde
   green_led(1);
```

```
  while (sens_der>400 || sens_izq>400){
    play ("c32");
    set_motors(40,-40);      // gira or -40,40
    delay_ms(200);
    lee_sensores();
    play ("g32");
  }
  red_led(0);           // OK salida encontrada
  green_led(0);                 // Apaga leds rojo y verde
}

int main(){
   // inicializa el robot pololu 3pi
   iniciar();
   // Bucle principal
   while(1){
     lee_sensores();
     if (sens_der>400 || sens_izq>400){
```

```
   set_motors(0,0);      // para los motores
   delay_ms(200);
   busca_salida();
}
if (sens_izq > 200 && sens_der < 200){
// obstaculo izq entonces gira a la derecha ------>
   motor_izq=100-sens_der/10;   // acelera
   motor_der=100-sens_izq/10;   // reduce
}
if (sens_izq < 200 && sens_der > 200){
// obstaculo der entonces gira a la izquierda <-----
   motor_izq=100-sens_der/10;   // reduce
   motor_der=100-sens_izq/10;   // acelera
}
if (sens_izq < 200 && sens_der < 200){
   motor_izq=100;
   motor_der=100;
}
```

```
    if (sens_izq < 100 && sens_der < 100){
       motor_izq=127;
       motor_der=127;
     }

      set_motors(motor_izq, motor_der);
  delay_ms(100);
  clear();          // Muestra los valores
     lcd_goto_xy(0,0);
     print_long(sens_izq);
     lcd_goto_xy(5,0);       // valores de sensores
     print_long(sens_der);
     lcd_goto_xy(0,1);
     print_long(motor_izq);
     lcd_goto_xy(5,1);       // valores de motores
     print_long(motor_der);
    }
}
```

CAPÍTULO 4

4. ANÁLISIS DE LOS RESULTADOS, VALIDACIÓN Y PRUEBAS

4.1. SIMULACIÓN EN PROTEUS

Gracias a la ayuda de la herramienta de simulación Proteus, se logró realizar la mayor parte de este proyecto, en ciertos casos fue necesario recurrir a la utilización de elementos ya existentes.

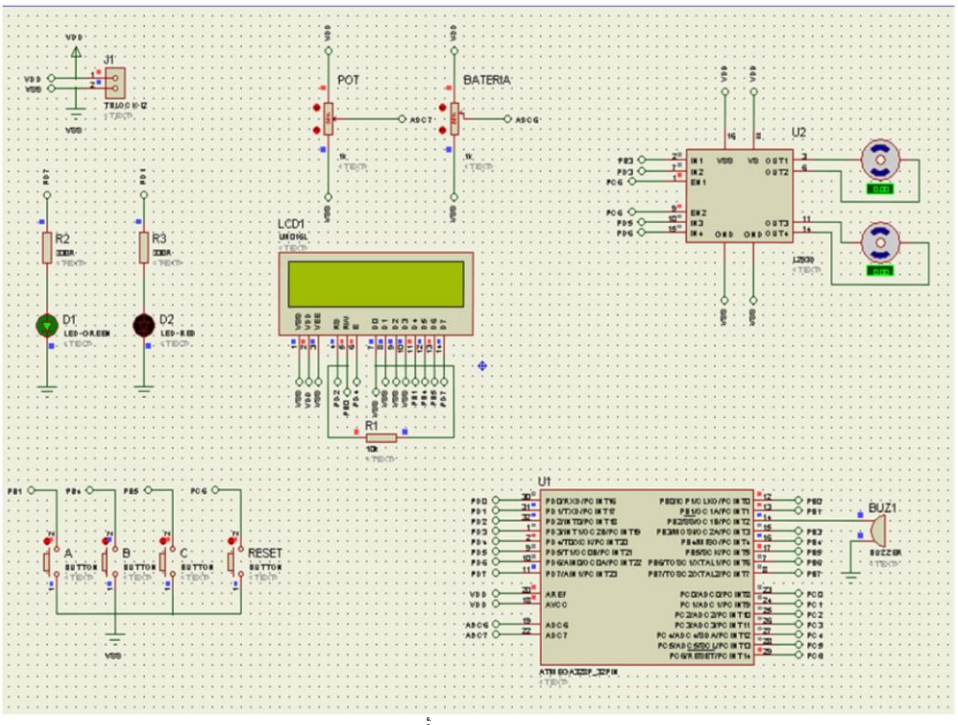


FIGURA 4.1 Diagrama en Proteus del Pololu 3Pi

La simulación de nuestros sensores Sharp lo hemos implementado con el uso de unos potenciómetros conectados a una fuente y variamos el potenciómetro para simular la señal que emite al detectar un obstáculo a una distancia que emiten los sensores Sharp.

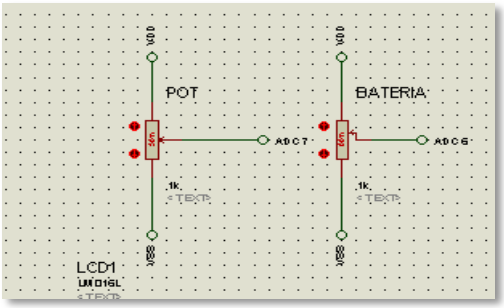


FIGURA 4.2 Esquemático de las entradas analógicas del Pololu

El problema de la simulación de los sensores Sharp es por su curva característica la cual nos representa un rango de valores de voltaje, los cuales pueden ser mal interpretados en distancias entre 10cm y 80 cm debido a que son valores de voltaje de similares magnitudes para las respectivas distancias.

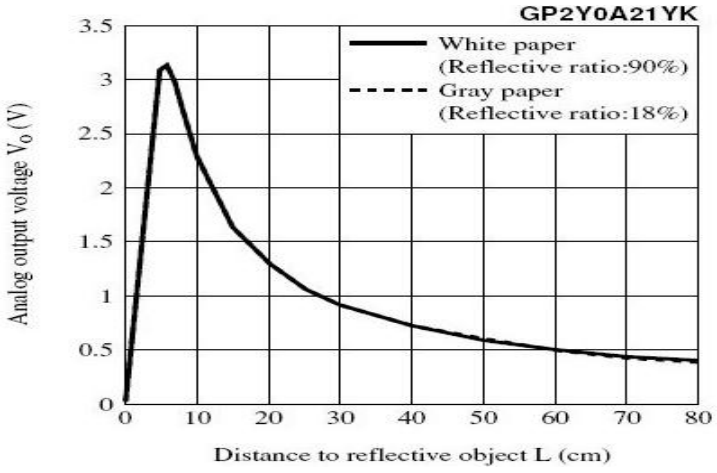


FIGURA 4.3 Curva de voltaje - distancia de los sensores Sharp

4.2. IMÁGENES DEL PROYECTO FUNCIONAL

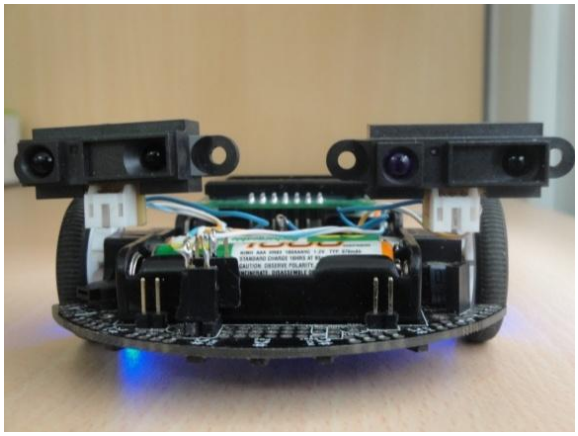


FIGURA 4.4 Robot Pololu 3Pi con los sensores acoplados

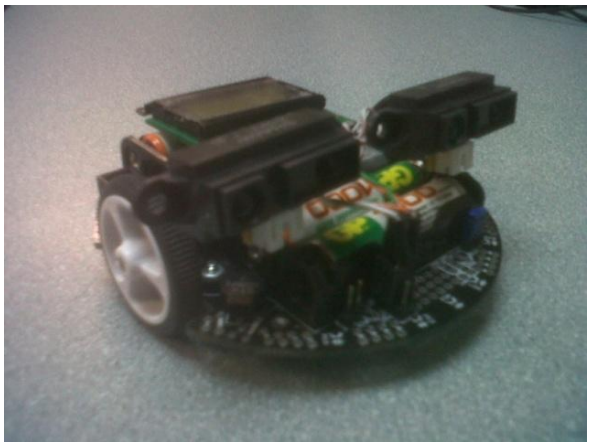


FIGURA 4.5 Ubicación de los sensores en el robot Pololu 3Pi

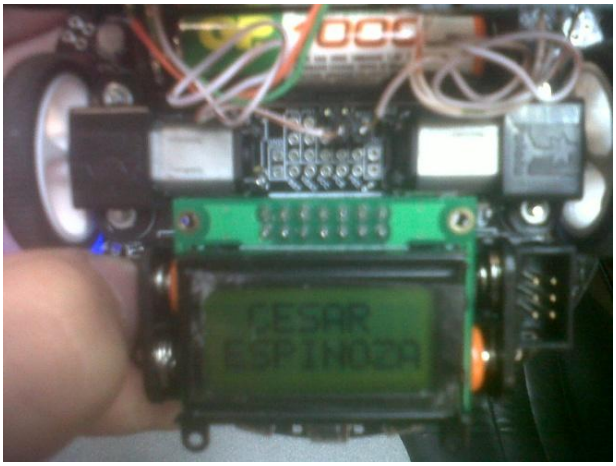


FIGURA 4.6 Nombre de integrante 1 en la pantalla LCD

FIGURA 4.7 Nombre de integrante 2 en la pantalla LCD

FIGURA 4.8 Nivel actual del voltaje mostrado en la LCD funcionando

El inicio del proyecto empieza mostrando en la pantalla LCD los nombres de los integrantes del proyecto. Además nos pide presionar la tecla B del POLOLU para que los motores puedan arrancar progresivamente desde el reposo hasta alcanzar una velocidad constate y adecuada para que permita que los sensores pueden estimar correctamente las distancias de los obstáculos que se presenten en la trayectoria del robot. Mientras espera por el botón B, se muestra en la LCD el valor del voltaje de las baterías expresada en mili voltios; si el valor es menor a 4500 mV se presentara el mensaje “CARGAR”.

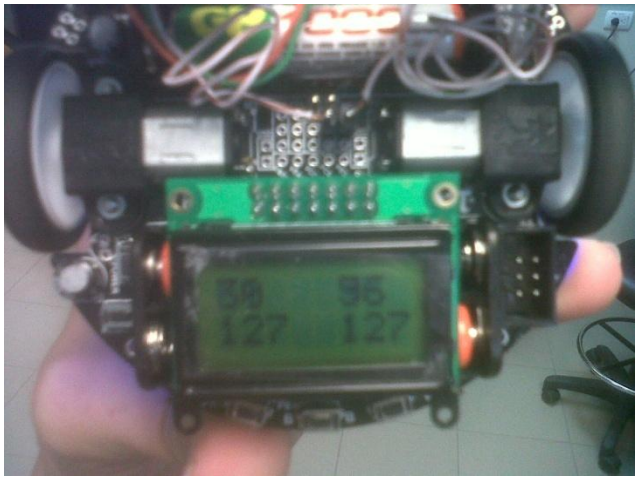


FIGURA 4.9 Nivel de velocidad de cada motor y distancia respectiva de cada sensor mostrados en la LCD

FIGURA 4.2 Mensaje “Buscando” salida durante el funcionamiento

Cuando el robot se encuentra en movimiento, los mensajes que se presenta en la LCD son los datos correspondientes a las velocidades de los motores y sus respectivas distancias si se presenta algún obstáculo en la trayectoria. Además cuando el robot encuentra algún objeto, describe un pequeño giro previniendo una posible colisión pero en el caso que el obstáculo aparece de improvisto, de que el robot no lo haya detectado o que el obstáculo no estuvo en una posición en la cual los sensores hayan logrado detectar el objeto. El robot se detiene inmediatamente y gira en busca de una salida y presenta un mensaje que dice “BUSCANDO”.

CONCLUSIONES

1. Con una configuración de dos sensores tenemos la posibilidad de resolver una gran cantidad de limitaciones respecto a la detección de los obstáculos, sin embargo debemos tomar en cuenta la necesidad de obtener más información del medio (mas sensores) al quedar sujeto a condiciones durante el recorrido del robot Pololu 3Pi en una pista o una sala, ya que es una limitación muy difícil de enfrentar.

2. Este proyecto nos brinda la oportunidad de resolver problemas lógicos de la vida real aplicados a la robótica por medio del ingenio en la programación; logrando demostrar que con una reducida cantidad de información, se puede conseguir resolver una gran cantidad de limitaciones de desplazamiento al momento de evadir obstáculos mediante mediciones periódicas de localización.

3. Los sensores ópticos son una manera muy acertada de obtener información de distancia para el Atmega328 del Robot Pololu para la detección de objetos. Con estas señales se hizo más sencillo relacionar constantemente la ubicación de los obstáculos en el medio, así como la capacidad de realizar una respuesta inmediata y eficaz.

RECOMENDACIONES

1. Se recomienda colocar los sensores de tal manera que estén en línea con el horizonte, es decir evitando que se encuentren en ángulos de elevación o depresión. Con esto se logrará obtener un mejor rendimiento del robot mejorando su capacidad para movilizarse sin colisionar.

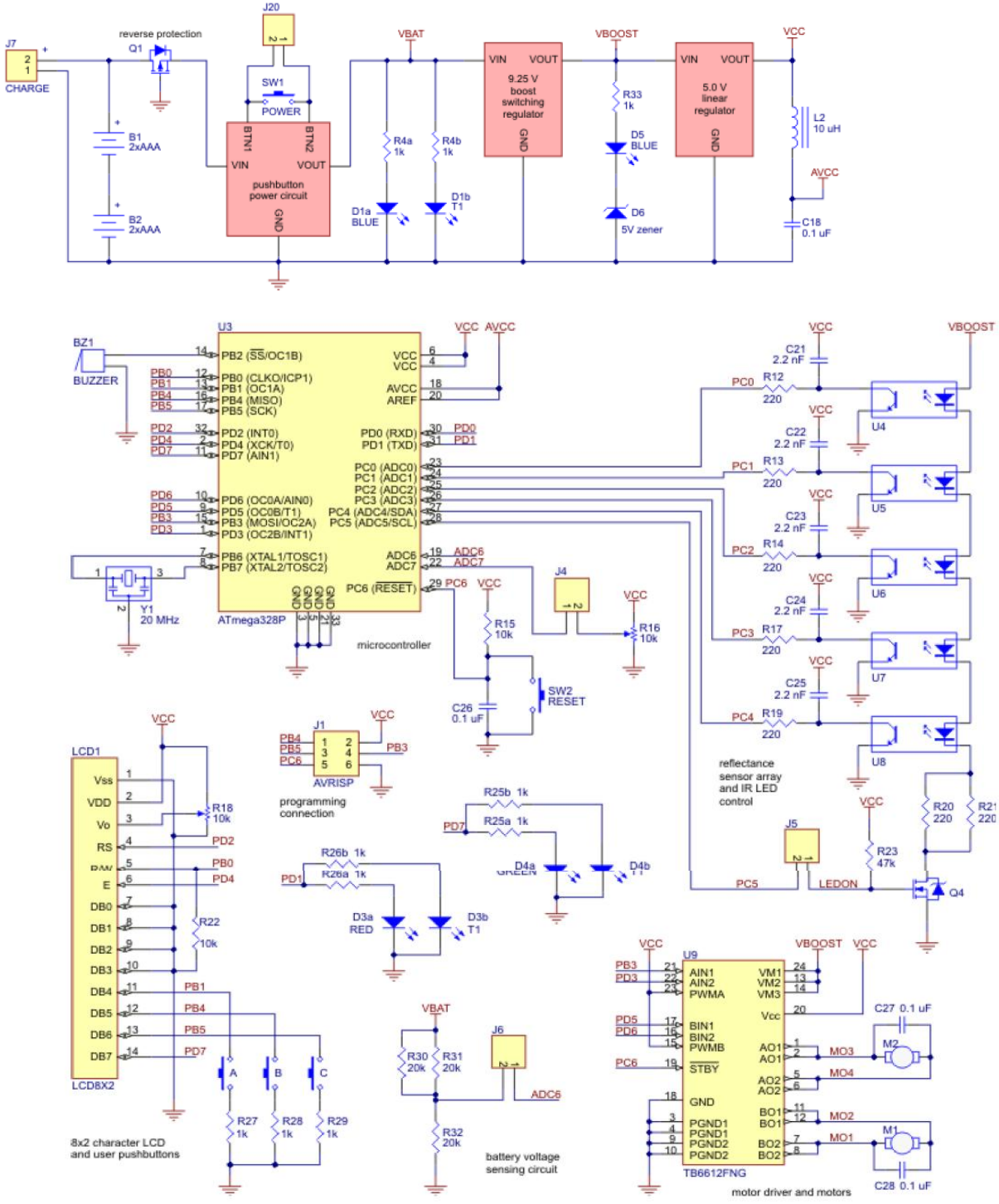
2. Se recomienda regular la velocidad de los motores del Pololu 3Pi y probar con distintos valores hasta obtener el valor que brinde un mejor resultado, de esta manera se evitarán giros bruscos y errores por encontrarse muy cerca de los obstáculos.

3. Para evitar errores de detección por efecto del punto ciego se recomienda que el robot tenga colocados los sensores a una mínima distancia de separación entre ellos; ya que mientras más juntos coloquemos a los sensores, menor será el porcentaje de que un objeto caigo en el punto ciego.

ANEXOS

Anexo1

Esquemático reducido del robot Pololu 3Pi


J7
CHARGE
reverse protection
Q1
B1
2xAAA
B2
2xAAA
J20
SW1
POWER
BTN1
BTN2
VIN
VOUT
pushbutton
power circuit
GND
VBAT
R4a
1k
R4b
1k
D1a
BLUE
D1b
T1
VIN
VOUT
9.25 V
boost
switching
regulator
GND
VBOOST
R33
1k
D5
BLUE
D6
5V zener
VIN
VOUT
5.0 V
linear
regulator
GND
VCC
L2
10 uH
AVCC
C18
0.1 uF
U3
BZ1
BUZZER
PB2 (SS/OC1B)
PB0 (CLKO/ICP1)
PB1 (OC1A)
PB4 (MISO)
PB5 (SCK)
PD2 (INT0)
PD4 (XCK/T0)
PD7 (AIN1)
PD6 (OC0A/AIN0)
PD5 (OC0B/T1)
PB3 (MOSI/OC2A)
PD3 (OC2B/INT1)
PB6 (XTAL1/TOSC1)
PB7 (XTAL2/TOSC2)
PC6 (RESET)
VCC
AVCC
PD0 (RXD)
PD1 (TXD)
PC0 (ADC0)
PC1 (ADC1)
PC2 (ADC2)
PC3 (ADC3)
PC4 (ADC4/SDA)
PC5 (ADC5/SCL)
ADC6
ADC7
ATmega328P
Y1
20 MHz
microcontroller
J4
R15
10k
R16
10k
SW2
RESET
C26
0.1 uF
C21
2.2 nF
C22
2.2 nF
C23
2.2 nF
C24
2.2 nF
C25
2.2 nF
R12
220
R13
220
R14
220
R17
220
R19
220
U4
U5
U6
U7
U8
reflectance
sensor array
and IR LED
control
J5
LEDON
R23
47k
Q4
R20
220
LCD1
Vss
VDD
Vo
RS
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
LCD8X2
R18
10k
R22
10k
PD2
PB0
PD4
PB1
PB4
PB5
PD7
A
B
C
R27
1k
R28
1k
R29
1k
8x2 character LCD
and user pushbuttons
J1
AVRISP
programming
connection
PB4
PB5
PC6
PB3
R25b 1k
R25a 1k
PD7
D4a
D4b
R26b 1k
R26a 1k
PD1
D3a
RED
D3b
T1
VBAT
R30
20k
R31
20k
J6
ADC6
R32
20k
battery voltage
sensing circuit
U9
AIN1
AIN2
PWMA
BIN1
BIN2
PWMB
STBY
GND
PGND1
PGND1
PGND2
PGND2
VM1
VM2
VM3
Vcc
AO1
AO1
AO2
AO2
BO1
BO1
BO2
BO2
TB6612FNG
C27 0.1 uF
C28 0.1 uF
M1
M2
MO1
MO2
MO3
MO4
motor driver and motors

Anexo 2

Descripción de los sensores Sharp usados

GP2Y0A21YK0F

Distance Measuring Sensor Unit
Measuring distance: 10 to 80 cm
Analog output type

■Description

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

■Features

1. Distance measuring range : 10 to 80 cm
2. Analog output type
3. Package size : 29.5×13×13.5 mm
4. Consumption current : Typ. 30 mA
5. Supply voltage : 4.5 to 5.5 V

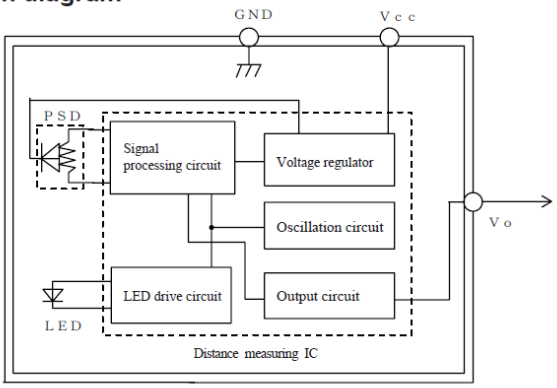
■Agency approvals/Compliance

1. Compliant with RoHS directive (2002/95/EC)

■Applications

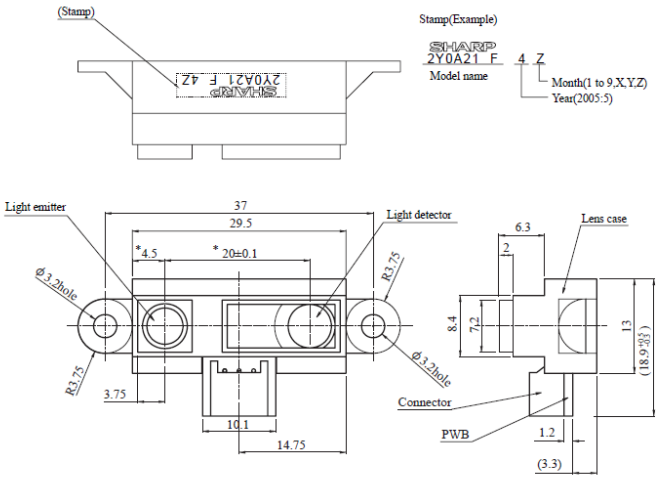
1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Robot cleaner
3. Sensor for energy saving
   (ATM, Copier, Vending machine)
4. Amusement equipment
   (Robot, Arcade game machine)

■Block diagram



■Outline Dimensions

(Unit : mm)



Connector signal

	signal name
①	Vo
②	GND
③	Vcc

Connector :
J.S.T.TRADING COMPANY,LTD,
S3B-PH

Materials
Lens :Acrylic acid resin
(Visible light cut-off resin)
Case :Carbonic ABS
(Conductive resin)
PWB :Paper phenol

Note 1. The dimensions marked * are described the dimensions of lens center position.
Note 2. Unspecified tolerances shall be ± 0.3 mm.
Note 3. The dimensions in parenthesis are shown for reference.

Product mass : Approx. 3.6g

BIBLIOGRAFÍA

1. Catálogo de características principales del Robot pololu.

http://www.pololu.com/catalog/product/136

Fecha de consulta: 20 de abril del 2011.

2. Catálogo del Sensor óptico GP2Y0A21.

http://www.pololu.com/catalog/product/975

Fecha de consulta: 27 de abril del 2011.

3. Documentación de la Librería para robots pololu.

http://www.pololu.com/docs/0J21/6.a

Fecha de consulta: 27 de abril del 2011.

4. Manual de guía de usuario del robot pololu.

http://www.pololu.com/file/0J137/Pololu3piRobotGuiaUsuario.pdf

Fecha de consulta: 27 de abril del 2011.

5. Colisiones.

http://robots-argentina.com.ar/Sensores_parachoques.htm

Fecha de consulta: 1 de mayo del 2011.

6. Origen del nombre 3pi y herramientas para su programación.

http://robots.net/article/2866.html

Fecha de consulta: 27 de Abril 2011.

7. Función para la cual fue diseñado el Pololu 3Pi.

http://www.pololu.com/catalog/product/975

Fecha de consulta: 27 de Abril 2011.

8. Guía de usuario del Robot Pololu 3pi.

http://www.pololu.com/docs/pdf/0J21/3pi.pdf

Fecha de consulta: 27 de Abril 2011.

9. Datasheet GP2Y0A21YK0F.

http://www.google.com.ec/#hl=es&source=hp&biw=1366&bih=575&q=GP2Y0A21YK0F+datasheet&aq=f&aqi=&aql=&oq=&fp=78ccf0f8cf40c309

Fecha de consulta: 27 de Abril 2011.

10. Descripción del Pololu AVR USB programmer.

http://www.pololu.com/catalog/product/1300

Fecha de consulta: 27 de Abril 2011.