



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y
Computación

Control de Robot Pololu con sensores de distancia para
mantener equidistancia a referencia móvil

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN
ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentado por:

Jessica Isabel Saavedra Castro
José Luis Chávez Aguilar

GUAYAQUIL – ECUADOR
2011

AGRADECIMIENTO

DEDICATORIA

A Dios, por ser la guía durante todo este camino, por llenarme de bendiciones y brindarme aliento en todo momento.

A mi familia, por los sabias consejos y el apoyo contante, por sus anhelos de verme exitosa cumpliendo mis metas y nuevos objetivos día a día.

Al Ingeniero Carlos Valdiviezo por su extraordinaria supervisión y guía durante el desarrollo del proyecto y por proyectar un profesionalismo destacable.

Jessica Isabel Saavedra Castro

DEDICATORIA

A mis padres:

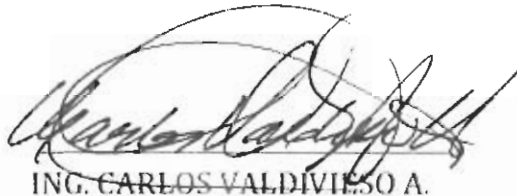
Pilar fundamental de mi vida en el desarrollo **tanto personal** como profesional, ya que gracias a ellos **preste todo** mi esfuerzo y dedicación.

Al **Ingeniero Carlos** Valdivieso:

Quien supo **guiar de manera** correcta y segura la **elaboración del** proyecto. Gracias a su **modelo a seguir como profesional** lograre **cumplir todas mis metas con éxito.**

José Luis Chávez Aguilar

TRIBUNAL DE SUSTENTACIÓN

A handwritten signature in black ink, appearing to read 'Carlos Valdivieso A.', written over a horizontal line.

ING. CARLOS VALDIVIESO A.

PROF. DEL SEMINARIO DE GRADUACIÓN

A handwritten signature in black ink, appearing to read 'Hugo Villavicencio V.', written over a horizontal line.

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 Graduación de la ESPOL)



José Luis Chávez Aguilar



Jessica Isabel Saavedra Castro

RESUMEN

Existen sensores visuales como las cámaras de video y algunos tipos de sensores no visuales como el sonar, el radar, los sensores inerciales y los de activación por presión, estos sensores se activan de acuerdo a la proximidad de un obstáculo, generando un patrón diferente de acuerdo al ángulo de choque.

El proyecto consiste en la visualización y detección de obstáculos a través de dos sensores de distancia los cuales funcionan como emisores y receptores a la vez, para el cual estos proyectan una luz infrarroja y de acuerdo al tiempo que se demoren en recibir la señal de retorno ellos avanzan o se detienen.

Para que el robot al detectar el obstáculo sea eficiente se debe tener una localización del vehículo precisa para con esto poder orientar al mismo en que ruta debe tomar cuando se encuentre con un objeto en su trayectoria.

El proyecto fue desarrollado en lenguaje C haciendo uso del AVR Studio con su compilador GCC, el cual permite la compilación no solo de lenguaje C sino también de C++. Cabe recalcar que al ser un software libre y trabajar bajo el entorno de AVR no estamos limitados a un tamaño mínimo de generación de código y la mayoría de funciones se encuentran a disposición para ser modificados, además para la simulación se hizo uso de las herramientas de software Proteus7.

INDICE GENERAL

CONTENIDO

RESUMEN.....
INTRODUCCION.....
CAPITULO 1.....	1
1 DESCRIPCIÓN GENERAL DEL PROYECTO.....	1
1.1 HISTORIA Y ANTECEDENTES.....	1
1.2 DESCRIPCIÓN DEL PROYECTO.....	2
1.3 APLICACIONES.....	5
1.4 PROYECTOS SIMILARES.....	6
1.4.1 DETECTOR DE OBSTÁCULOS Y GRAFICACIÓN DE POSICIÓN DE OBSTÁCULOS.....	6
1.4.2 CONTROL DE MICROBOT SIKO CON TARJETA CT6811.....	7
1.4.3 RESUELVE LABERINTOS.....	8
CAPITULO 2.....	9
2 FUNDAMENTO TEÓRICO.....	9
2.1 ROBOT POLOLU 3PI.....	9
2.1.1 BATERIAS.....	11
2.1.2 GESTION DE LA ENERGIA.....	12
2.1.3 MOTORES Y ENGRANAJES.....	15
2.2 PROGRAMADOR AVR VIA USB.....	18
2.3 DESCRIPCION DEL SOFTWARE.....	20
2.3.1 AVR STUDIO 4.....	20
2.3.2 PROTEUS 7.7.....	24

CAPÍTULO 3	25
3 DISEÑO E IMPLEMENTACION DEL SISTEMA.....	25
3.1 DISEÑO PRELIMINAR.....	25
3.1.1 MOVIMIENTO DE UN MOTOR CON CONTROL DE VELOCIDAD Y DIRECCION.....	26
3.2 IMPLEMENTACION FISICA.....	29
3.2.1 SENSOR SHARP GP2Y0A21YK.....	30
3.2.1.1 CONVERSION ANALOGICA DIGITAL VS VOLTAJE VS DISTANCIA.....	31
3.2.2 PANTALLA LCD 8X2.....	32
3.3 DESCRIPCION DEL PROYECTO FINAL.....	33
3.4 DIAGRAMA DE BLOQUES.....	35
3.5 DIAGRAMA DE FLUJO.....	37
3.6 CODIGO DETALLADO DEL PROYECTO	38
CAPITULO 4	45
4 ANALISIS DE LOS RESULTADOS, VALIDACION Y SIMULACION DE PRUEBAS..	45
4.1 SIMULACION EN PROTEUS.....	45
4.2 FOTOS DE LOS RESULTADOS.....	47
CONCLUSIONES.....	
RECOMENDACIONES.....	
BIBLIOGRAFÍA	
ANEXOS	

INDICE DE FIGURAS

Figura 1.2.1.- Robot Pololu 3pi.....	3
Figura 1.2.2.- Sensor de distancia Sharp GP2Y0A21YK.....	4
Figura 1.3.1.- Aplicaciones en el mundo.....	6
Figura 1.4.1.1.- Robot XBOT.....	7
Figura 1.4.1.2.- Graficación de posición en StampPlot.....	7
Figura 1.4.2.1.- Microbot SIKO.....	8
Figura 1.4.2.2.- Tarjeta CT6811.....	8
Figura 1.4.3.1.- Laberinto para resolver con Pololu 3pi.....	8

Figura 2.1.1. Robot Pololu 3pi vista superior.....	10
Figura 2.1.1.1.-Baterías recargables NiMH.....	11
Figura 2.1.1.2.- Grafico de proporcionalidad de las baterías.....	12
Figura 2.1.2.1.- Circuito interno con las baterías.....	13
Figura 2.1.2.2.- Circuito de monitorización de las baterías.....	14
Figura 2.1.3.1.- Funcionamiento del motor: corriente y velocidad vs torque.....	16
Figura 2.2.1.- Programador USB AVR - Pololu 3pi.....	18
Figura 2.2.2. Programador USB AVR Pololu con cable ISP 6 pines y cable USB A mini B.....	19
Figura 2.3.1.1.- Pantalla de inicio del AVR Studio 4.....	21
Figura 2.3.1.2.- Pololu USB AVR Programmer.....	22
Figura 2.3.1.3.- Interfaz de programación con Puerto ISP.....	22
Figura 2.3.2.1.- Ventana principal del programa de simulación Proteus 7.7.....	24
Figura 3.1.1.1.- Control de rotación por puente H.....	26
Figura 3.1.1.2.- Señal de modulación PWM.....	28
Figura 3.1.1.3.- Control de motores por conducción diferencial.....	28
Figura 3.2.1.1.- Conexión de cables entre el Sharp y Pololu.....	30
Figura 3.2.1.1.1.-Gráfica de linealidad de los datos del Sharp.....	31
Figura 3.2.2.1.- Pantalla LCD de 8x2 caracteres.....	32
Figura 3.4.1.- Diagrama de Bloques del Proyecto.....	36
Figura 3.5.1.- Diagrama de Flujo Principal.....	37
Figura 4.1.1.- Presentación del LCD y las baterías.....	45
Figura 4.1.2.- Simulación de los motores.....	46
Figura 4.1.3.- Motores ejecutando el desplazamiento hacia atrás.....	46
Figura 4.1.4.- Motores ejecutando giro hacia la derecha.....	47
Figura 4.2.1.- Pololu implementado para el desarrollo del proyecto.....	47
Figura 4.2.2.- Nombres de los integrantes en la pantalla LCD.....	48
Figura 4.2.3.- Pololu leyendo los sensores S1 y S2.....	48
Figura 4.2.4.- Implementacion de sensores Sharp al Robot Pololu.....	49

INTRODUCCION

Existen sensores visuales como las cámaras de video y algunos tipos de sensores no visuales como el sonar, el radar, los sensores inerciales y los de activación por presión, estos sensores se activan de acuerdo a la proximidad de un obstáculo, generando un patrón diferente de acuerdo al ángulo de choque.

El proyecto consiste en la visualización y detección de obstáculos a través de dos sensores de distancia los cuales funcionan como emisores y receptores a la vez, para el cual estos proyectan una luz infrarroja y de acuerdo al tiempo que se demoren en recibir la señal de retorno ellos avanzan o se detienen.

Para que el robot al detectar el obstáculo sea eficiente se debe tener una localización del vehículo precisa para con esto poder orientar al mismo en que ruta debe tomar cuando se encuentre con un objeto en su trayectoria.

El proyecto fue desarrollado en lenguaje C haciendo uso del AVR Studio con su compilador GCC, el cual permite la compilación no solo de lenguaje C sino también de C++. Cabe recalcar que al ser un software libre y trabajar bajo el entorno de AVR no estamos limitados a un tamaño mínimo de generación de código y la mayoría de funciones se encuentran a disposición para ser modificados, además para la simulación se hizo uso de las herramientas de software Proteus7.

CAPÍTULO 1

1 DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 HISTORIA Y ANTECEDENTES

La Robótica, desde sus inicios hasta la época actual, ha evolucionado y experimentado cambios que, gracias a la tecnología, en los últimos años ha permitido optimizar el control de robots por medio de controladores programables facilitando la utilización de motores que necesitan un mayor torque, debido a que poseen drivers que satisfacen las condiciones requeridas y de eso depende que modelo se selecciona de mejor manera.

Existe una gran cantidad de proyectos hechos por estudiantes de distintas universidades en el mundo que tratan sobre automóviles inteligentes utilizando todo tipo de controladores, si bien unos han sido programados para esquivar obstáculos, guiarse mediante una línea, colores e incluso por ultrasonido, otros que guían vehículos en caminos o carreteras, todo esto se hace con la ayuda de sensores y con el ante mencionado controlador.

El pololu es un robot nuevo y diferente ya que se lo puede programar con la plataforma AVR-GCC. El fabricante ofrece una multitud de librerías de fácil manejo que nos permiten controlar tanto el LCD, leer el valor de los sensores o ajustar la velocidad de rotación de los motores y así optimizar el seguimiento. Se pueden descargar varios programas como por ejemplo un simple seguidor de línea básico, otro programado con un cierto nivel de inteligencia y así hacer que pueda resolver un laberinto al mismo tiempo que grabar todo el recorrido.

Es que estos robots a parte de la tecnología incorporada con la que cuentan, está la circuitería y programación utilizados en un automóvil inteligente para guiarse de forma independiente y sin la intervención directa de humanos. Como ayuda a la sociedad ya que los robots cumplen una mejor eficiencia que una persona normal, esto en cierto periodo de tiempo.

Experiencias anteriores tanto investigadas como consultadas por estudiantes que han desarrollado ejemplos o proyectos parecidos, informaron que han podido observar y darse cuenta que este robot es de fácil uso tanto en software como en hardware aunque ya depende de la función que queramos que el robot desempeñe, Aunque el uso común que le dan a este robot es de hacer uso de sus 5 sensores frontales que sirven para reconocer colores o plataformas.

1.2 DESCRIPCIÓN DEL PROYECTO

El proyecto consiste en la implementación de un sistema que permita el control de un robot Pololu con el microcontrolador ATmega328p, por medio de sensores de distancia para mantener equidistancia a una

referencia móvil dependiendo de las instrucciones ejecutadas por el código previamente cargado.

El robot Pololu es un pequeño robot autónomo de alto rendimiento, diseñado generalmente para competencias de seguimiento de líneas y resolución de laberintos. Es alimentado por 4 pilas AAA y posee un sistema de tracción que trabaja a 9.25v para los motores.



Figura 1.2.1.- Robot Pololu 3 Pi

El Atmel ATmega328 es un microcontrolador programable el cual permite entre varias de sus aplicaciones controlar leds, LCDs y motores debido a un integrado que suministra la corriente requerida sin dañar este módulo ni otros componentes constituidos en el mismo.

Además, de los cinco sensores de reflexión que posee este robot, se añadirán dos sensores de distancia, los cuales controlaran la equidistancia requerida por el usuario respecto de una referencia móvil por medio de un voltaje análogo.

A partir de estos elementos ya mencionados, dentro de la características principales de este proyecto están; el desarrollo de un robot inteligente programado para seguir objetos en movimiento, como por ejemplo, puertas abriendo o cerrando; optimizar el código previamente programado para ampliar su función evitando colisiones con objetos a su paso manteniendo una distancia fija proporcionada por el usuario; realizar un programa que permita esquivar el objeto luego de verificar, mediante un tiempo determinado, como verdadero o falso la realización de un movimiento por parte de la referencia ya sea esta fija o móvil.

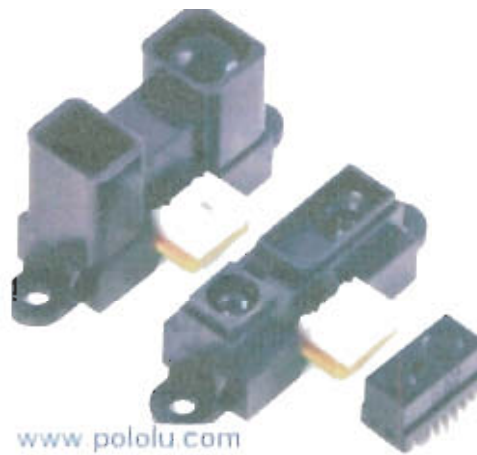


Figura 1.2.2- Sensor de distancia Sharp GP2Y0A21YK

En este caso que es un pololu 3pi programado para mantener equidistancia a referencia móvil el cual utiliza un sensor de distancia modelo Sharp GP2Y0A21YK el cual tiene características como: rango de detección de 4" a 32", un indicador análogo de voltaje que señala la distancia de entre diez a ochenta centímetros de alcance ante una referencia.

El objetivo principal del robot es el de seguir en caso la referencia se mueva, cabe recalcar que algo adicional al proyecto principal es de programarle un tiempo de espera para que reconozca en caso sea o no una referencia móvil, caso contrario el robot busque otro camino para seguir con su movimiento, esto se hará con la ayuda de varios menús programados en el pololu para no opacar el objetivo principal del proyecto.

Desde hace ya varios años, dentro de la industria se ha hecho una extensa investigación en el campo de los automóviles inteligentes. Compañías tales como Motorola, Delphi Automotive Systems, Chrysler y NASA, han prestado especial atención a este tema ya que se ha descubierto que existe un amplio mercado disponible para el desarrollo de vehículos inteligentes, debido a que estos ayudan a incrementar los factores de seguridad, así como la comodidad y la conveniencia que estos ofrecen para los conductores en general, ya que este tipo de tecnología representa un auxiliar en cuanto a la prevención de accidentes, facilitando el desplazamiento de un sitio a otro con comodidad y tranquilidad

1.3 APLICACIONES

Entre las aplicaciones se puede observar el comercio, la seguridad de los autos y la exploración espacial como parte importante del interés adquirido a la investigación de este tipo de automóviles o robots bajo control, sin embargo existen otras aplicaciones que se ven increíblemente beneficiadas con desarrollos de ingeniería y ciencia como estos; por ejemplo, podemos encontrar pequeños vehículos autónomos enfocados hacia la minería y a exploración subterránea.



Figura 1.3.1- Aplicaciones en el mundo

El mismo tipo de lógica e ingeniería ha sido aplicado ya a otro tipo de robots que se podrían considerar para funcionar con la base de los vehículos autónomos, tales como son los robots de rescate que planean utilizar distintos países para recuperar a soldados desaparecidos.

Otro prototipo es en la guerra en donde se ha comenzado el uso de aviones y tanques inteligentes que puedan combatir sin la necesidad de la intervención directa de los soldados para de esta manera salvaguardar a los involucrados en las misiones.

1.4 PROYECTOS SIMILARES

1.4.1 DETECTOR DE OBSTÁCULOS Y GRAFICACIÓN DE POSICIÓN DE OBSTÁCULOS

El proyecto consiste en la detección de obstáculos de manera que puedan ser evitados y enviar mediante comunicación inalámbrica la posición del obstáculo para mostrarla a través del PC en el programa Stamplot.

Para la elaboración del proyecto se usa un robot XBOT el cual cuenta con servomotores, un sensor ultrasónico y un módulo inalámbrico. El

control del robot será realizado a través del microcontrolador PIC16F876.

Al funcionar el robot detecta todo obstáculo a una distancia de 10 cm. En ese momento, por programación, el robot gira 90° hacia la izquierda, envía su posición a través del módulo inalámbrico. Al no encontrar obstáculo continuará su recorrido actualizando cada ciclo de programa su posición. El receptor consiste en otro PIC16F876 que tiene el módulo de recepción inalámbrica en donde cada dato que es recibido es enviado al StampPlot para la graficación.



Figura 1.4.1- Robot XBOT

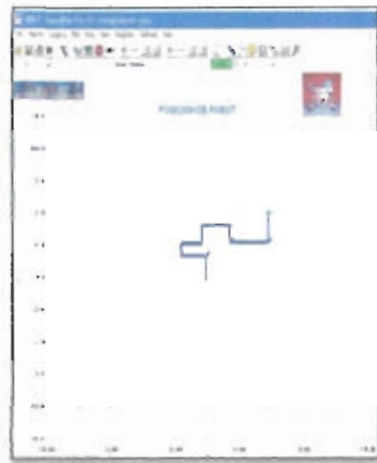


Figura 1.4.2- Graficación de posición en StampPlot

1.4.2 CONTROL DE MICROBOT SIKO CON TARJETA CT6811

SIKO es un robot oruga de bajo costo que puede ser utilizado por aquellas personas que se inician en el mundo de la robótica. El esqueleto está armado con una plancha de aluminio y cuenta además con servomotores modificados para poder funcionar como motores de corriente continua normal y poder rotar 360°. Se tiene como finalidad obtener un robot de bajo costo capaz de movilizarse a través de superficies no lisas y con pequeños obstáculos.

Para el control del robot se utiliza la tarjeta CT6811, basada en el microcontrolador 68HC11 de Motorola y la tarjeta CT293+ para el

control de los motores y sensores de infrarrojo, aunque se puede usar otro microcontrolador.

SIKO es un robot abierto, es decir que tanto los esquemas como la documentación de su estructura mecánica están disponibles y su elaboración y modificación están permitidas.



Figura 1.4.2.1- Microbot SIKO

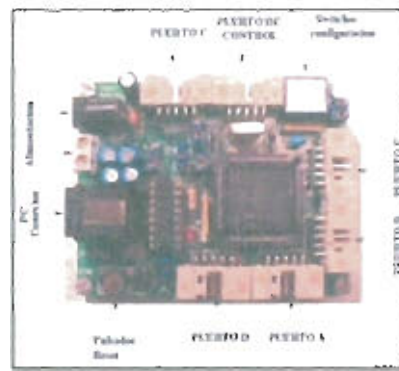


Figura 1.4.2.2- Tarjeta CT6811

1.4.3 RESUELVE LABERINTOS

Una 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 caminos para resolverlo. Existe también la posibilidad de memorizarlo para ubicarse directamente hacia la llegada en un futuro segundo recorrido.

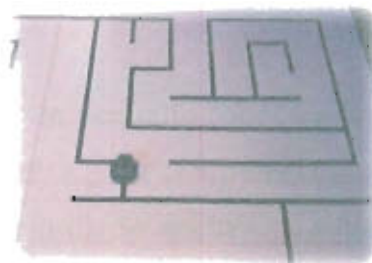


Figura 1.4.3.1 Laberinto para resolver con Pololu 3pi

CAPÍTULO 2

2 FUNDAMENTACIÓN TEÓRICA

Para el desarrollo de este proyecto, basado en el Control de un Robot Pololu, se precisará realizar la construcción del mismo con elementos físicos, más conocidos como Hardware; así como también se utilizarán varias herramientas de programación y simulación, las cuales se involucran directamente con la explicación del Software, con el fin de que el usuario conozca el tipo de tecnología junto con las herramientas que se pueden agregar para el análisis del propósito en cuestión.

2.1 ROBOT POLOLU 3PI

El 3pi de Pololu es un pequeño robot autónomo de alto rendimiento, diseñado comúnmente para competencias de seguimiento de línea y resolución de laberintos. Se encuentra alimentado por 4 pilas AAA y posee un único sistema de tracción para los motores que trabajan a 9.25V, el 3pi es capaz de alcanzar velocidades por encima de los 100cm/s mientras realiza vueltas precisas y cambios de sentido que no varían con el voltaje de las baterías.

Los resultados son consistentes y están bien sintonizados con el código aún con baterías bajas. El robot está totalmente ensamblado con dos micromotores de metal para las ruedas, cinco sensores de reflexión, una pantalla LCD de 8x2 caracteres, un buzzer, más de tres pulsadores y todo ello está conectado a un microcontrolador programable. El 3pi mide aproximadamente 9,5 centímetros de diámetro y pesa alrededor de 83 gramos sin baterías.

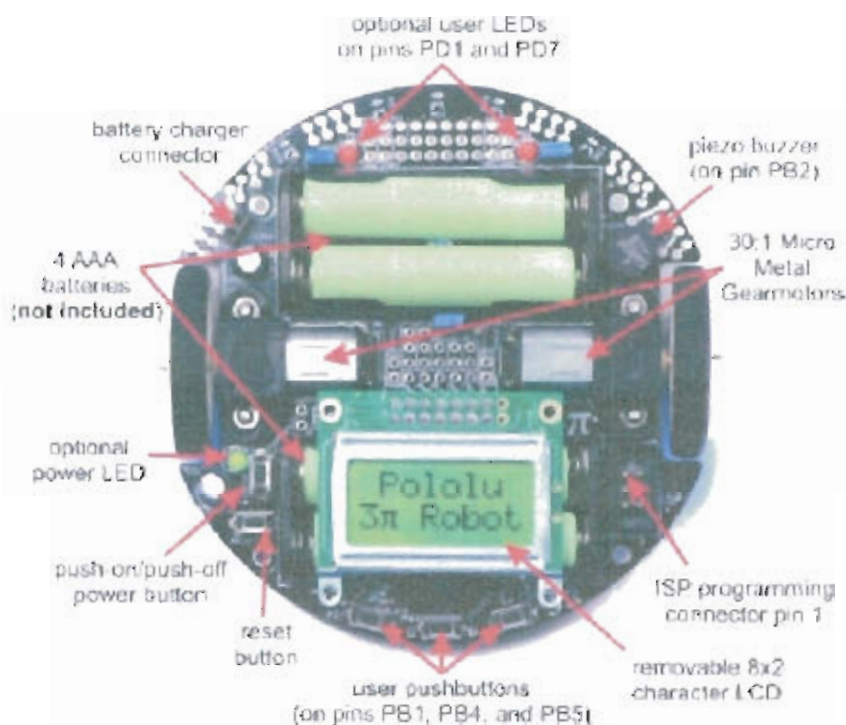


Fig. 2.1.1- Robot Pololu 3pi vista superior

Este robot contiene un microcontrolador Atmel ATmega168 o un ATmega328 a 20MHz con 16KB de memoria flash y 1KB de RAM, el doble, 32 KB y 2KB en el Atmega328 y 1KB de EEPROM. El uso del ATmega328 lo hace compatible con la plataforma de desarrollo Arduino. Las herramientas gratuitas de desarrollo en C y C++, así como un extenso paquete de librerías que están disponibles, se pueden trabajar con el hardware que lleva incorporado.

2.1.1 BATERÍAS

La potencia del sistema del Pololu empieza con las baterías, por eso es importante conocer su funcionamiento. Las baterías contienen unos elementos químicos que reaccionan moviendo electrones desde un terminal positivo (+) a uno negativo (-). El tipo más conocido es la pila alcalina compuesta de zinc y manganeso en una solución de hidróxido potásico. Por bien del medio ambiente, cuando las baterías se descargan completamente deben ir al reciclado.



Figura 2.1.1.1.- Baterías recargables NiMH

Para este robot es recomendable utilizar las baterías de níquel-manganeso, expresado como NiMH, que tienen la propiedad de recargarse una y otra vez: de ahí el nombre de baterías recargables. Estas baterías realizan una reacción diferente a las alcalinas y es de gran importancia conocer su estado de medición con unos simples números. Lo primero a comprender será que la cantidad de electrones que se mueven de un terminal a otro se miden en Voltios (V) o también llamado, diferencia de potencial. En las baterías de NiMH es de 1,2V.

Para justificar la fuerza de la batería es necesario averiguar cuántos electrones circulan por segundo, esto es, la intensidad que será medida en Amperios (A). Una corriente de 1A equivale a 6×10^{18} electrones saltando por segundo. Un amperio, es la fuerza que un motor de tamaño

mediano podría necesitar y sería la corriente que las pequeñas pilas AAA deberían de proporcionar.

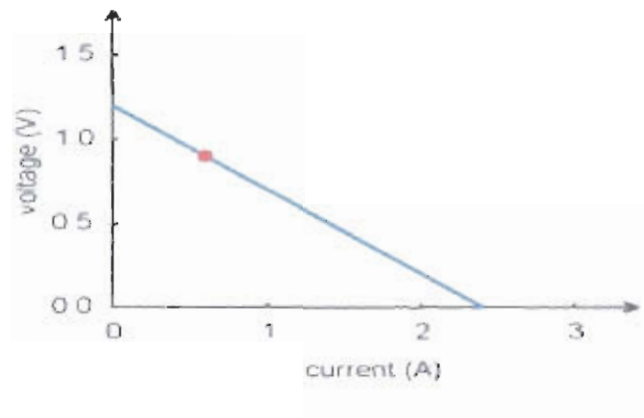


Fig. 2.1.1.2.- Gráfico de proporcionalidad de las baterías

2.1.2 GESTIÓN DE LA ENERGÍA

El voltaje de la batería se reduce con el uso, pero los componentes eléctricos usados precisan de un 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.

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

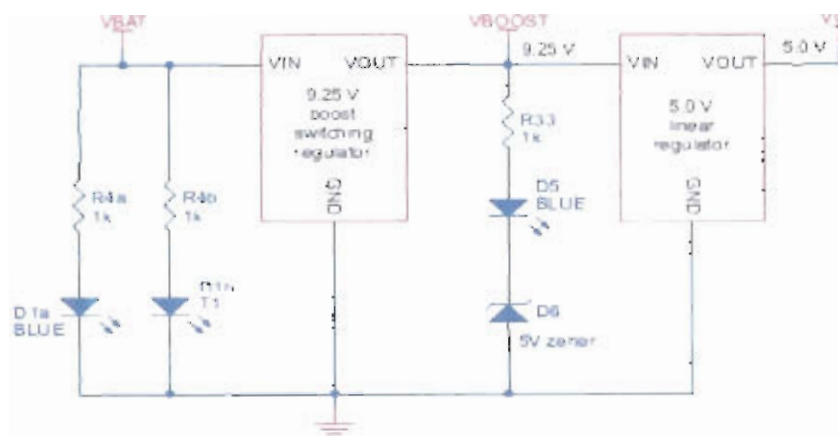


Fig. 2.1.2.1- Circuito interno con las baterías.

El voltaje de cuatro pilas AAA 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. Se utiliza un regulador **switching** para elevar el voltaje a 9,25 V, regulador **VBoost**, y reguladores lineales para obtener 5V de VCC. **VBoost** sirve para los motores y los leds sensores en línea, mientras que el **VCC** es para el microcontrolador y las señales digitales. Usando el **Vboost** para 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 90° varias veces, a pesar de la cantidad de tiempo que esto lleva consigo.
- Tercero, a 9,25 V los leds conectados en línea consumen una cantidad más pequeña de energía, se puede alternar la conexión y 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 detenga. Esto puede sorprender, pero al mismo tiempo podría servir para monitorizar la tensión de la batería e indicar la recarga de las baterías.

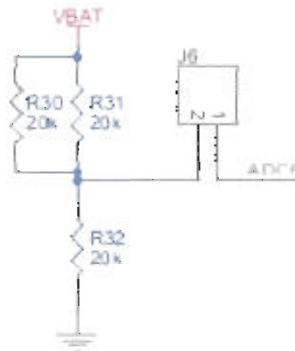


Figura 2.1.2.2.-Circuito de monitorización de la batería

Un circuito simple de monitorización de la batería se encuentra en el 3pi como se muestra en la figura previa. Tres resistencias en el circuito manejan un divisor de tensión de $2/3$ el voltaje de las baterías. Este se encuentra conectado a una entrada analógica del microcontrolador y mediante la programación produce por ejemplo: a 4,8 V el pin ADC6 tiene un nivel de 3,2V. Usando una conversión analógica de 10 bit, un valor de 5V se lee como 1023 y un valor de 3,2 se lee como 655. Para convertir el actual estado de la batería multiplicamos $5000\text{mV} \cdot 3/2$ y dividimos por 1023. Para ello se hace útil la función `read_battery_millivolts()` función que puede promediar diferentes lecturas y devolver el resultado en mV:

```
unsigned int read_battery_millivolts()
{
    return readAverage(6,10)*5000L*3/2/1023
}
```


2.1.3 MOTORES Y ENGRANAJES

El motor es una maquina 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. El típico motor DC contiene imanes permanentes en el exterior bobinas electromagnéticas montadas en el eje del motor. Las escobillas son piezas deslizantes que suministran corriente desde una parte del bobinado hacia la otra, produciendo una serie de pulsos magnéticos que permites que el eje gire en la misma dirección.

El principal valor que describe a los motores es la velocidad representada en rpm, revoluciones por minuto, y la fuerza de torque, medido en kg·cm (kilogramos - centímetros) o en oz·in (onzas - pulgadas). El torque es la medida que muestra directamente la dependencia entre la fuerza y la distancia, por lo tanto si se multiplica el torque y la velocidad se obtiene la potencia entregada por el motor.

Cada motor posee una velocidad máxima, cuando no existe ninguna fuerza aplicada, y un torque máximo, cuando el motor se encuentra en reposo completamente. A esto se le conoce como velocidad normal de funcionamiento y torque máximo.

Generalmente, un motor utiliza el mínimo de corriente cuando no hay fuerzas aplicadas al mismo, por lo tanto la corriente que viene de las baterías aumenta teniendo en cuenta a la corriente de funcionamiento

y a la corriente máxima, lo cuales son parámetros de gran importancia dentro de la característica de un motor igualmente. La corriente máxima usualmente es mucho mayor que la corriente de funcionamiento como se muestra en la figura a continuación:

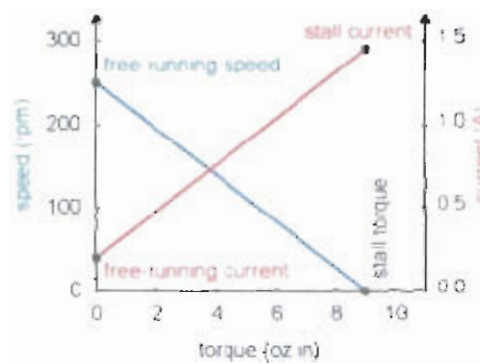


Figura 2.1.3.1- Funcionamiento del Motor: corriente y velocidad vs torque

La velocidad normal de funcionamiento de motor DC pequeño usualmente es miles de rotaciones por minuto, es decir, mucho más de la velocidad que se desea para que las llantas del robot giren. El sistema de engranajes consiste en convertir una velocidad máxima - torque mínimo de salida, en velocidad mínima - torque máximo de salida, lo cual es lo ideal para ser aplicado a un robot de las características previamente descritas.

El engranaje usado en el 3pi, utiliza una relación de 30:1, la cual significa que, por cada 30 vueltas de motor, la salida será una vuelta de rueda. Esto reduce la velocidad en un factor de 30, lo cual incrementa el torque de igual forma. Estos parámetros están representados en la siguiente tabla:

Tabla de parámetros (Motor - 3pi)	
Engranaje	30:1
Velocidad	700 rpm
Corriente	60 mA
Torque máximo	6 onz.in
Corriente máximo	540 mA

Tabla 2.1.3.1- Parámetros del Motor Pololu 3pi

Cada una de las dos llantas del 3pi tienen un radio de 0.67 pulgadas, es decir que la máxima fuerza que se puede producir con los dos motores en funcionamiento será:

$$\frac{2 \times 6}{0.67} = 18 \text{ onz.}$$

El robot 3pi pesa alrededor de unas 7onz. Incluidas las baterías, por lo tanto los motores son lo suficientemente fuertes para moverlo en una pendiente o acelerarlo a 2g., lo cual es el doble de la aceleración de la gravedad.

2.2 PROGRAMADOR AVR VIA USB

Este dispositivo es un programador para controladores de la familia de los AVR, tales como los controladores del robot Orangután y el Pololu 3pi. El programador emula un AVRISP v2 en un puerto serie virtual, convirtiéndole compatible con el software estándar de programación AVR.

Dos características adicionales ayudan a generar y depurar proyectos: un puerto serie nivel - TTL para la comunicación de uso general y un SIO-osciloscopio de aplicación para el monitoreo de señales y los niveles de tensión. Un cable USB y el cable ISP están incluidos.

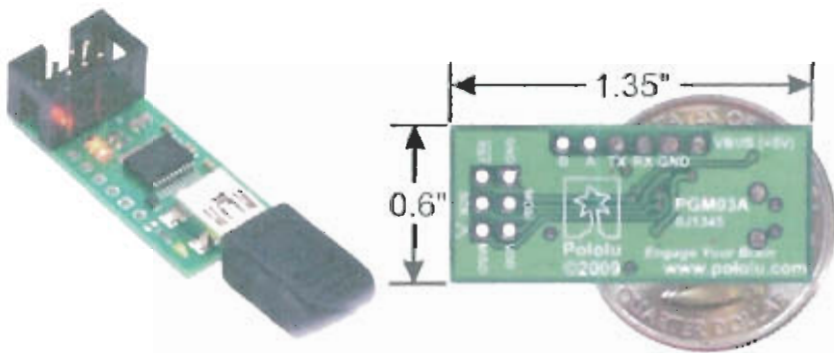


Figura 2.2.1- Programador USB AVR - Pololu 3pi

El programador AVR Pololu USB es un muy compacto y posee un sistema de programación (ISP) de bajo costo para microcontroladores

AVR de Atmel, lo que hace a este dispositivo una solución de programación atractiva para los equipos basados en controladores AVR como los controladores del Orangután y 3pi.

El programador USB AVR se conecta al computador vía USB mediante un puerto incluido USB A, a un cable mini-B que se comunica con el software de programación, tales como AVR STUDIO o AVRDUDE, a través de un puerto virtual COM utilizando el protocolo AVRISPv2/STK500. El programador se conecta al dispositivo de destino a través de un cable de programación ISP de 6 pines. El programador es energizado a través de un bus USB de 5v y está diseñado para programar los AVR que se funcionen con este voltaje. Es importante recalcar que el programador no proporciona energía directamente hacia el equipo.



Figura 2.2.2- Programador USB AVR Pololu con cable ISP 6 pines y cable USB A mini B.

El programador USB AVR tiene doble función como adaptador, es decir, de USB a Puerto Serial. Este programador se instala como dos puertos virtuales COM, uno para la comunicación de la programación con el software y otro de propósito general. A través de estas características se logra manejar la programación en AVR mientras que

se realiza la depuración por medio del puerto serial TTL sin necesidad de abrir o cerrar el programa.

Además de las líneas de transmisión, el programador permite utilizar los pines A y B como entrada o salidas señales de ondas. El programador también permite el acceso a la fuente regulada de cinco voltios USB a través del pin VBUS.

2.3 DESCRIPCION DEL SOFTWARE

2.3.1 AVR STUDIO 4

AVR Studio es un ambiente de desarrollo IDE (Integrated Development Environment), que se utiliza para escribir y simular aplicaciones para los microcontroladores de la familia ATME1.

Provee herramientas de manejo, editor de código fuente, que puede ser en lenguaje ensamblador o lenguaje C. Permite visualizar los diferentes bancos que posee el microcontrolador que estemos programando y además facilita manipular esos valores durante la ejecución del programa para observar los cambios generados por los mismos.

El lenguaje ensamblador nos permite manipular con mayor detalle los valores de los registros que posee en microcontrolador que estemos utilizando. Al mismo tiempo posee herramientas que permiten ver los valores que toman las diferentes variables que hayamos declarado a lo largo del desarrollo del programa.

El lenguaje GCC (GNU C Compiler) permite una programación de alto nivel de modo que consigamos desarrollar aplicaciones de mayor grado de desarrollo.

AVR Studio 4 posee además la opción de programación de los microcontroladores ATMEL mediante interfaz JTAG, ISP y Serial, de las cuales serán usadas las dos últimas para realizar la programación de los módulos: la programación Serial para programar el AVR Butterfly mediante la opción de Bootloader y la programación ISP para el módulo Orangután SV-328. Cabe mencionar que el Butterfly también permite cargar una aplicación mediante su puerto ISP de 6 pines con la diferencia que se debe conectar además una polarización externa de 5V y retirar la batería.

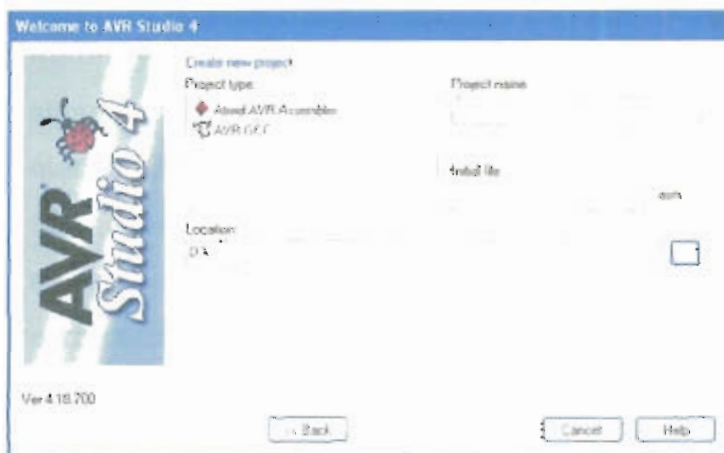


Figura 2.3.1.1- Pantalla de inicio del AVR Studio 4

AVR Studio dispone de varios métodos para poder programar los microcontroladores de la familia ATMEL, de las cuales serán usados para la realización de nuestro proyecto las siguientes opciones:

- La programación ISP, a la cual se accede mediante la opción AVRISP, permite grabar el microcontrolador tanto del Orangután SV-328 como del AVR Butterfly. Se hace uso del Pololu USB AVR Programmer el cual se conecta al puerto ISP de los módulos a través de un cable de 6 líneas.

Una vez conectado el módulo a programar el software permite la lectura de la firma del módulo para detectar su estado, lo

cual permite comprobar la correcta conexión entre el computador y el módulo. Una vez verificada la conexión se procede al proceso de grabación del microcontrolador en el cual se realiza en primera instancia el borrado del programa que ya contenía antes de cargar la nueva aplicación y finalmente realiza la verificación de la aplicación cargada. Una vez terminado el proceso abandona el modo de programación para que sea seguro desconectar el módulo sin causarle algún daño.

Cabe destacar que para realizar la programación del AVR Butterfly se debe conectar, además del programador, una alimentación externa de 5V y retirar la pila que energiza el módulo. Al encender el Butterfly con la alimentación externa éste debería encender sin ningún problema lo cual sería muestra de que la conexión de la polarización externa ha sido correctamente realizada y podemos continuar con el proceso de grabación normalmente.



Figura 2.3.1.2- Pololu USB AVR Programmer



Figura 2.3.1.3.- Interfaz de programación con puerto ISP

- La programación serial a la cual se accede mediante la opción AVRProg hace uso de la aplicación de Bootloader precargada en el Butterfly para su programación. En este caso se utiliza el puerto serial del computador conectado al puerto UART del AVR Butterfly el cual posee un convertidor de nivel lo cual realiza la conversión de nivel de voltaje de RS-232 a un voltaje que es soportado por el módulo.

Para realizar el proceso de grabado se debe presionar el botón del Joystick en su posición central y en ese momento hacer click sobre la opción Tools -> AVRProg. Si se ha realizado la acción correctamente aparecerá la ventana de programación, caso contrario se mostrará un mensaje de error. Se debe notar que durante el proceso de programación el AVR Butterfly no muestra nada en su pantalla.

Al momento de realizar la programación el software realizar el borrado de la aplicación anteriormente cargada para programar la que nosotros hayamos seleccionado y luego realiza la verificación antes de abandonar el modo de programación. Antes de desconectar el AVR Butterfly se debe hacer click sobre el botón EXIT del AVRProg para que se desconecte por completo y no vaya a causar algún daño al módulo por mala desconexión.

A pesar que al cargar una nueva aplicación al Butterfly se realiza el borrado del programa anterior la opción de Bootloader no se borra de modo que el módulo puede ser grabado varias veces por este mismo método.

2.3.2 PROTEUS 7.7

Proteus 7.7 es un software de simulación y diseño de PCB distribuido por Labcenter Electronics, que permite analizar el comportamiento de los circuitos, previo a la implementación física del circuito.

Provee de herramientas de edición, a su vez de librerías que contienen componentes análogos y familias de microcontroladores entre ellas Microchip, Motorola, ARM, AVR, y microprocesadores de la familia ATMEL, Motorola entre otros.

Una de las ventajas para el desarrollo del proyecto es que se hizo uso de los ejemplos para ATMEL en el cual encontramos el Kit AVR Butterfly funcionando con su programa demo, con lo cual no fue necesario el diseño del esquemático correspondiente.

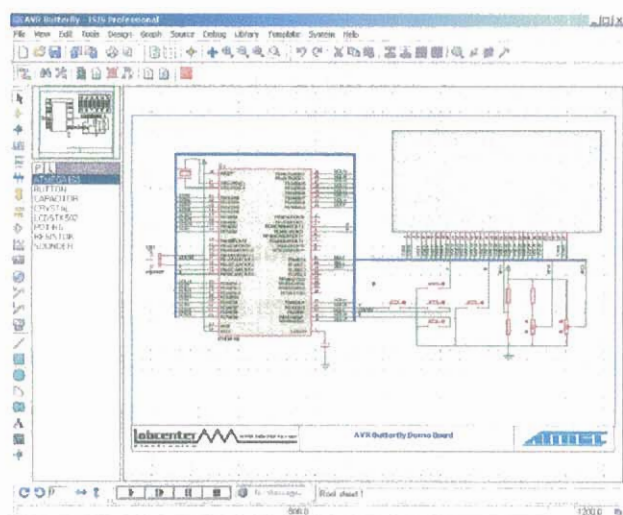


Figura 2.3.2.1.- Ventana Principal del programa de Simulación Proteus 7.7

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1 DISEÑO PRELIMINAR

El diseño de este proyecto se basa en controlar el Robot Pololu utilizando la ayuda de sensores de distancia trabajando a manera de convertidores de señal. El Pololu deberá conservar una equidistancia proporcionada por el usuario, según el requerimiento de su proceso, la cual deberá mantener aún cuando el objeto localizado o a seguir se encuentre en movimiento.

El Robot también contará con la implementación de un control esquivador de objetos, ya sean estos estáticos o móviles, haciendo uso de un tiempo referencial para la ejecución de una decisión propuesta por el programador en el pololu 3pi.

3.1.1 MOVIMIENTO DE UN MOTOR CON CONTROL DE VELOCIDAD Y DIRECCIÓN.

Los motores DC cambian de dirección de rotación alternando la polaridad del voltaje aplicado, por lo tanto no se cambiará la conexión de pilas ni la de los motores para tener un control de dirección, por lo que se utilizan los llamados puentes H como muestra el diagrama:



Figura 3.1.1.1.- Control de rotación por puentes H

Los cuatro puntos de corte de corriente permiten el cambio de sentido. Los puentes en H se construyen mediante transistores que realizan las funciones de los interruptores. Se utilizan puentes para ambos motores en el 3pi mediante el chip TB6612FNG conectando las salidas de los puertos del micro-controlador correspondiente a los pines PD5 y PD6 para el motor M1 y para el motor M2 se utilizan los pines de control en PD# y PB3.

Funcionamiento descrito en la tabla a continuación:

PD5	PD6	1	2	3	4	M1		PD3	PB3	1	2	3	4	M2
0	0	off	off	off	off	off (coast)		0	0	off	off	off	off	off (coast)
0	1	off	on	on	off	forward		0	1	off	on	on	off	forward
1	0	on	off	off	on	reverse		1	0	on	off	off	on	reverse
1	1	off	off	on	on	off (brake)		1	1	off	off	on	on	off (brake)

Tabla 3.1.1.1.- Pines de salida para motores 1 y 2 del Pololu

Dado que el voltaje suministrado al motor es una serie de pulsos de anchura variable, a este método de control de velocidad se le llama modulación del ancho de pulso (PWM). En el 3pi, el control de velocidad se consigue usando las salidas de PWM del microcontrolador que generan los temporizadores Timer0 y Timer2.

Esto significa que se puede establecer un ciclo de trabajo PWM para los dos motores a la vez e independiente del resto de código por lo que seguirá produciendo señales en segundo plano, pudiendo prestar atención a otras necesidades. La función *set_motors()* de la librería AVR Pololu crea el ciclo de trabajo utilizando una precisión de 8 bits por lo que un valor de 255 corresponderá al 100%. Para una velocidad del 67% en el M1 y la otra del 33% en el M2 se llamará a la función de la siguiente forma: *set_motors(171,84)*. Para obtener un descenso lento de la secuencia del PWM será de gran utilidad el gráfico a continuación, con el cual se deberá escribir un bucle que gradualmente haga decrecer la velocidad del motor en el tiempo.

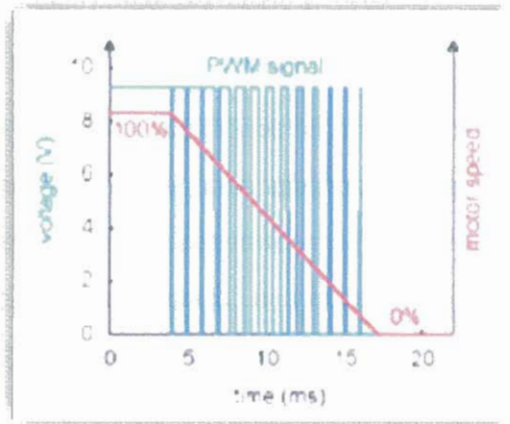


Figura 3.1.1.2.- Señal de modulación PWM

El 3pi tiene motores independientes a cada lado que crean un método de conducción denominado conducción diferencial, el cual también se conoce como "conducción de tanques". Para girar mediante este método es necesario hacer rodar los motores a diferentes velocidades. En el ejemplo de función anterior la rueda izquierda se mueve más deprisa que la derecha con lo que el robot avanza girando a la derecha. La diferencia de velocidades determina que el giro sea más suave o más brusco, e incluso moviendo un motor adelante y el otro atrás se consigue un cambio de dirección total.

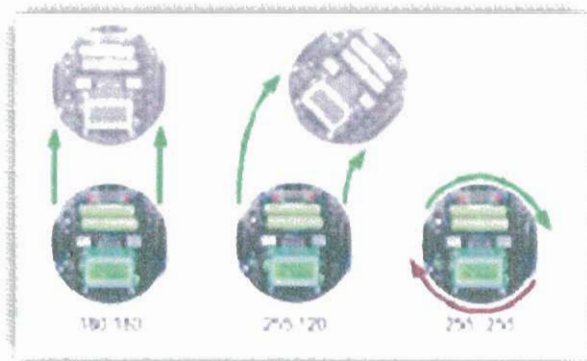


Figura 3.1.1.3.- Control de motores por Conducción Diferencial

Para este proyecto en particular el robot Pololu avanzará hasta que identifique un obstáculo, para efectos del tema propuesto, se tiene como principal objetivo responder al movimiento de una puerta; es decir que el 3pi avanzará o retrocederá guardando una cierta distancia, según esta referencia móvil, se cierre o se abra.

Luego de encontrarse con el obstáculo, la primera verificación que realizará el Robot, será la comprobación del tiempo de espera, definido por el programador, durante el cual debe existir movimiento por parte del objeto y de ser así seguirlo o caso contrario esquivarlo. Al momento de ser bloqueado por una referencia estática, éste debe ejecutar el control esquivador, el cual hace girar al Pololu 45° buscando orientación a la derecha.

Para el caso de toparse con una referencia móvil que se esté acercando al Robot, éste idealmente deberá ir retrocediendo, pero es de gran importancia que al realizar este movimiento no se choque con ningún obstáculo en la parte de atrás.

3.2IMPLEMENTACIÓN FÍSICA

En la implementación de este proyecto es necesario el empleo de varios elementos propios y externos al Pololu, los cuales fueron descritos previamente, dentro de la fundamentación teórica.

Parte estratégica de este control de distancia son los sensores que se encuentran ubicados a los lados del Pololu, ya que de éstos dependen las decisiones que este robot ejecute.

3.2.1 Sensores Sharp GP2Y0A21

Los sensores de distancia Sharp son una elección común para múltiples proyectos que requieren de una medición de distancia precisa. Estos sensores IR son más económicos y proveen muy buenos resultados. La interfaz que utiliza con la mayoría de microcontroladores es simple, es decir, la salida analógica deberá ser conectada al convertidor analógico – digital para que este lea las mediciones de distancias necesarias.

El GP2Y0A21 utiliza un conector 3-pin JST, el cual conecta los cables de alimentación al sensor soldándolos con mucha precaución.

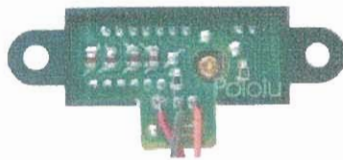


Figura 3.2.1.1- Conexión de cables entre el Sharp y el Pololu

En la tabla a continuación se presentará información adicional del Sharp GP2Y0A21:

Características Sharp GP2Y0A21	
Voltaje de Operación	4.5v – 5.5v
Consumo de corriente promedio	30mA
Rango de medición de distancia	10cm – 80cm
Tipo de salida	Voltaje analogo
Tiempo de respuesta	38 ± 10 ms
Peso	3.5g

Tabla 3.2.1.1- Características principales del sensor Sharp

3.2.1.1 Conversión Analógica ~ Digital vs Voltaje vs Distancia

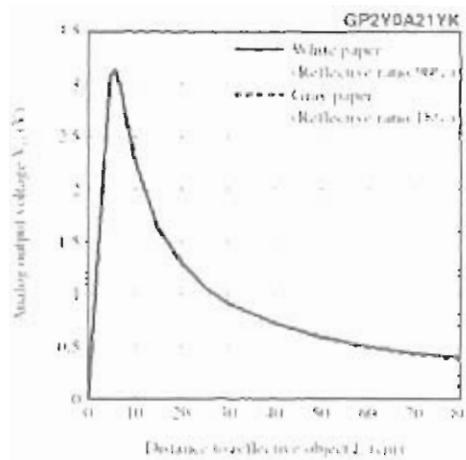


Figura 3.2.1.1.1- Gráfica para linealidad de los datos del Sharp

La relación entre el voltaje de salida de los sensores versus la distancia se la puede aproximar utilizando la gráfica a continuación para su conversión respectiva.

Para el desarrollo de este proyecto fue necesario la construcción de una tabla con los valores proporcionales de distancia versus salidas ADC obtenida por los sensores, la cual se detalla a continuación:

Tabla de Conversión (Cm - ADC)	
Cm	ADC
1	204
2	307
3	409

4	512
5	614
10	471
20	266
30	184
40	153
50	122
60	102
70	81
80	61
90	40
100	20

Tabla 3.2.1.1.1- Conversión ADC vs Distancia

3.2.2 Pantalla LCD 8x2

Esta pantalla 8x2 LCD compacta es ideal para proyectos de microcontroladores donde se requiere una **interfaz de resultados** más sofisticada que un solo led parpadeando. Esta unidad utiliza la interface paralela HD44780.



www.pololu.com

Tabla 3.2.2.1.- Pantalla LCD de 8x2 caracteres

Características LCD		
PIN	Símbolo	Función
1	Vss	Tierra (0v)
2	Vdd	5v (Fuente)
3	Vo	Ajuste de contraste
4	RS	H-L registro de señal
5	R/W	H-L señal lectura y escritura
6	E	H-L habilitación de señal
7-14	DB0 – DB7	H-L Data bus de 4 – 8 bits

Tabla 3.2.2.1.- Características principales de la pantalla LCD

Es de gran importancia utilizar baterías recargables ya que al momento de programar esto puede ser causa de errores y conflictos, provocando así hasta el daño el todo el equipo del Pololu.

3.3 DESCRIPCIÓN DEL PROYECTO FINAL

El robot avanza hasta que vea un obstáculo el cual interprete mediante un tiempo de espera si es móvil o estable, en este caso un tiempo de cuatro segundos. Al aproximarse al obstáculo se detiene a una distancia aproximada de doce centímetros, luego espera a que se abra la puerta para seguir caso contrario espera el tiempo ya mencionado y gira cuarenta y cinco grados hacia la izquierda o hacia la derecha y sigue avanzando hasta que ocurra algo similar y repite el proceso.

En el momento que se encuentre con una puerta y esta se abre, el robot avanza hasta que recupera la distancia de doce centímetros, pero si la puerta retrocede el también retrocede para mantener la distancia inicial de doce centímetros.

En el programa al inicio se declaran las variables:

- ✓ s1: valor del sensor de la derecha
- ✓ s2: valor del sensor de la izquierda
- ✓ Cnt: valor del contador del tiempo que espera frente al obstáculo.

Al encender el pololu muestra el mensaje de presentación la pantalla LCD:

JESSICA SAAVEDRA

JOSE CHAVEZ

A continuación espera que se presione el botón B que inicia todo el proceso mientras se muestra el valor de voltaje presente en la batería.

Una vez encendido lo mostrado en el LCD emite un sonido de la escatado re mi a volumen 10 (v10) con retardo de corche (18) y de las tres entradas analógicas esta lee el solo las del puerto 6 y 5.

Luego cargo los valores de los sensores s1 y s2, estos detectan si el robot esta a cierta distancia, en este caso en un rango de entre diez a quince centímetros y se detiene a esperar y empieza a contar.

Una vez detenido el robot, si el obstáculo se mueve el espera los cuatro segundos programados, si en este tiempo el obstáculo esta inmóvil busca otro camino girando hacia la derecha o hacia la izquierda cuarenta y cinco grados.

Caso contrario si en ese tiempo programado el obstáculo se mueve el robot lo sigue, sea para delante o hacia atrás, esto lo detecta el sensor el cual si ve una distancia corta retrocede para evitar colisiones mientras emite un sonido.

Por el contrario si los dos sensores reciben señales de distancias grandes y parecidas, él se detiene.

3.4 DIAGRAMA DE BLOQUES

El diagrama de bloques del sistema se encuentra constituido por las entradas y salidas directas del microcontrolador de la familia ATmega328p.

Tanto la etapa de transmisión como la de recepción está constituida por los sensores de distancia que envían señales de distancia al Pololu, el cual en la programación, las convierte en tiempos con los cuales el robot se mantendrá en un lugar determinado o podrá moverse según el código descrito.

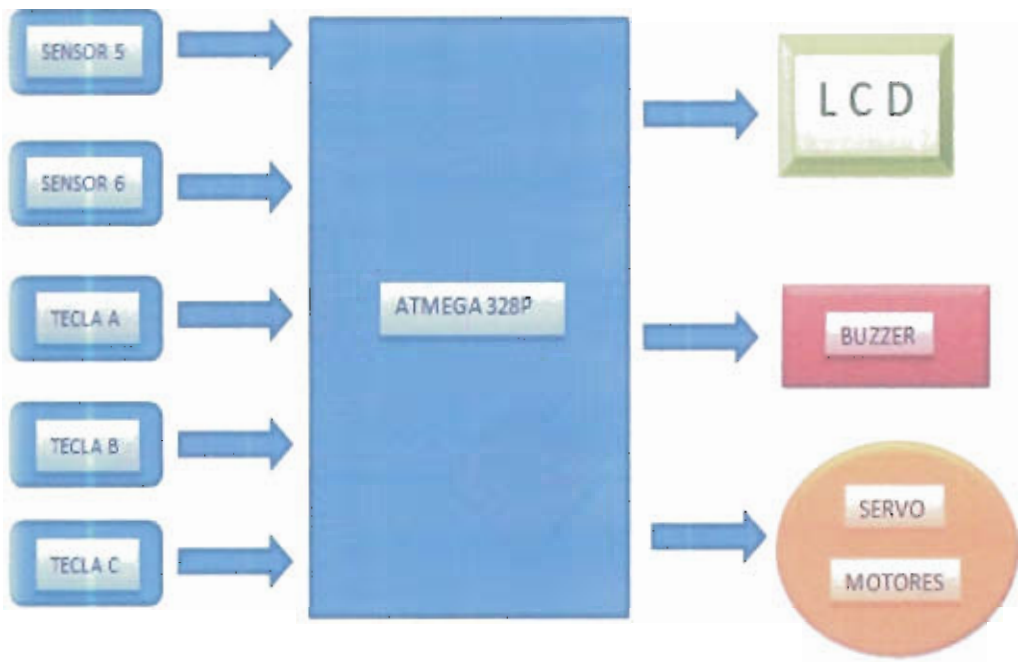


Figura 3.4.1- Diagrama de Bloques del Proyecto

3.5 DIAGRAMA DE FLUJO

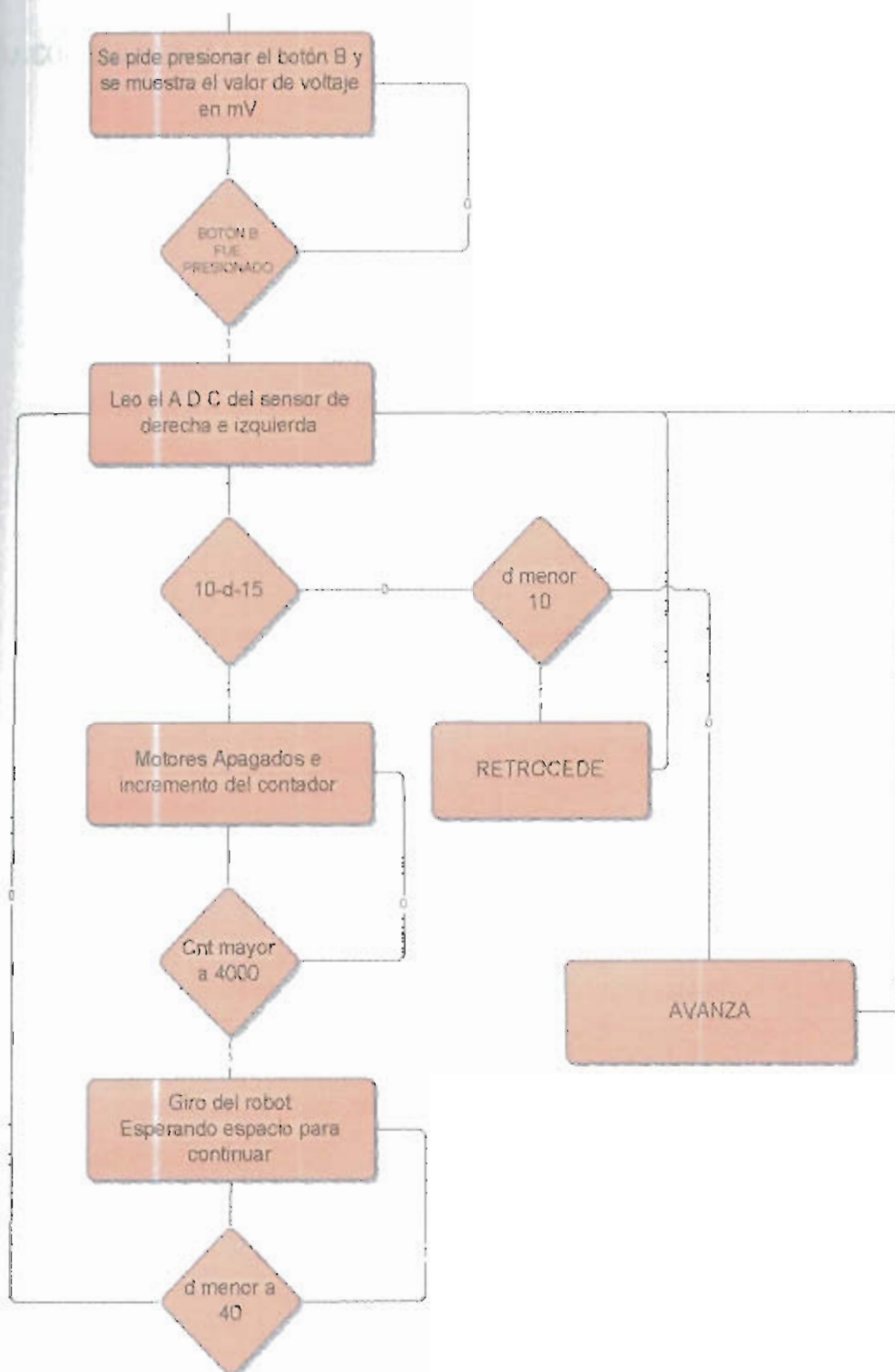


Figura 3.5.1- Diagrama de Flujo Principal

3.6 CÓDIGO DETALLADO DEL PROYECTO

A continuación se muestra el código hecho en lenguaje C con el cual pudimos diseñar el diagrama de flujo e implementar el robot.

```
#include <pololu/3pi.h>

//El robot avanza hasta que vea un obstaculo (Puerta) y se detiene a
una distancia

//aprox 12 cm, espera a que se abra la puerta para salir pero si en 4 seg
no susede

//nada el gira 45 grados y sigue avanzando hasta que ocurra algo
similar y repite

//el proceso, en el momento que se encuentre con una puerta y esta se
abre el robot

//avanza hasta que recuperar la distancia de 12 cm pero si la puerta
retrocede el

//tambien retrocede para mantener la distancia inicial de 12 cm si
durante este

//tiempo de espera sea retrocediendo o vanzando pasan 4 segundos sin
cambios el gira

//360 grados (el giro siempre lo hace hacia su derecha)

int main()
{
```



```
int s1,s2,cnt,cnt2;
```

```
s1=0; //valor del sensor de la derecha
```

```
s2=0; //valor del sensor de la izquierda
```

```
cnt=0; //valor del contador del tiempo que espera frente de la puerta
```

```
cnt2=0; //valor del contador para giro 500-45 1000-90 1500-135 ya  
asi, no fue necesaria su utilizacion
```

```
//Mensaje de Presentación
```

```
print("JESSICA");
```

```
lcd_goto_xy(0,1);
```

```
print("SAAVEDRA");
```

```
delay_ms(1500);
```

```
clear();
```

```
print("JOSE");
```

```
lcd_goto_xy(0,1);
```

```
print("CHAVEZ");
```

```
delay_ms(1500);
```

```
clear();
```

```
//Espera que se presione el botón b que inicia todo el proceso
```

```
//mientras muestra el valor de voltaje presente en la batería
```

```
while(!button_is_pressed(BUTTON_B))  
{  
  clear();  
  print_long(read_battery_millivolts());  
  print("mV");  
  lcd_goto_xy(0,1);  
  print("Press B");  
  delay_ms(100);  
}
```

//Limpia la pantalla para borrar el mensaje que decía presionar b

```
clear();
```

*//Emite un sonido de la escala do re mi... a volumen 10 (v10) con
retardo de corche(L8)*

```
play("L8 V10 crrdrerrrfrgrarrbr");
```

```
while(1){
```

//Lee el analogico en el puerto 6 y 5

```
s1= analog_read(6);
```

```
s2= analog_read(5);
```

```
//imprime el valor del sensor s1 y s2
```

```
print("s1 ");
```

```
print_long(s1);
```

```
lcd_goto_xy(0,1);
```

```
print("s2 ");
```

```
print_long(s2);
```

```
//Si se encuentra a una distancia entre 10 a 15 cm se detiene a esperar
```

```
while(s1<360 && s1>300){ //(d<15 && d>10) La relación es inversa
```

```
s1= analog_read(6);
```

```
s2= analog_read(5);
```

```
//imprime el valor del sensor s1 y s2
```

```
print("s1 ");
```

```
print long(s1);
```

```
lcd_goto_xy(0,1);
```

```
print("s2 ");
```

```
print_long(s2);
```

```
set_motors(0,0);
```

```
cnt++;
```

```
cnt2=0;
```

```
delay_ms(1);
```

```
if(cnt>=4000){  
  
    //Giro hasta ver espacio y salir emitiendo un sonido de aviso de giro  
    while(s1>100 && s2>100){  
  
        play("L8 V10 crdrerfgrarbr");  
  
        s1= analog_read(6);  
        s2= analog_read(5);  
  
        //imprime el valor del sensor s1 y s2  
        print("s1 ");  
        print_long(s1);  
        lcd_goto_xy(0,1);  
        print("s2 ");  
        print_long(s2);  
        delay_ms(100);  
        set_motors(40,-40);  
  
        //Signos diferentes para que gire  
        cnt2++;  
        cnt=0;  
    }  
}
```

}

}

//Si el sensor derecha e izq ve una distancia grande avanza

```
if(s1<360 && s2<360){//(d1>15 && d2>15)
```

```
s1= analog_read(6);
```

```
set_motors(40,40);
```

```
cnt=0;
```

}

//Si el sensor der ve una distancia corta retrocede para evitar colisiones mientras emite un sonido

```
if(s1>300){//(d1<10)
```

```
play("1.8 V10 cmerrrrrrrrrrrrrrre");
```

```
s1= analog_read(6);
```

```
set_motors(-40,-40);
```

```
cnt=0;
```

}

```
s1= analog_read(6);
```

```
s2= analog_read(5);
```

//imprime el valor del sensor s1 y s2

```
print("s1 ");
```

```
print_long(s1);  
lcd_goto_xy(0,1);  
print("s2 ");  
print_long(s2);  
  
//Deja limpia la LCD  
clear();  
  
}  
  
return 0;  
  
}
```

CAPÍTULO 4

4 ANÁLISIS DE LOS RESULTADOS, VALIDACIÓN Y PRUEBAS

4.1 SIMULACIÓN EN PROTEUS

Con la ayuda de la plataforma de simulación Proteus se hicieron pruebas divididas en partes ya que el sistema se complico al realizarlo en conjunto. Además la mayor parte de este proyecto se desarrollo en base a materiales existentes.

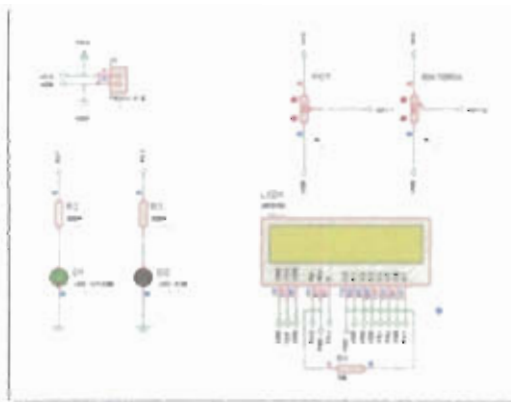


Figura 4.1.1- Presentación del LCD y las baterías

En la figura se muestra la simulación en parte del Pololu, mostrando cómo se enciende cada led y posterior a eso la presentación en la pantalla LCD.

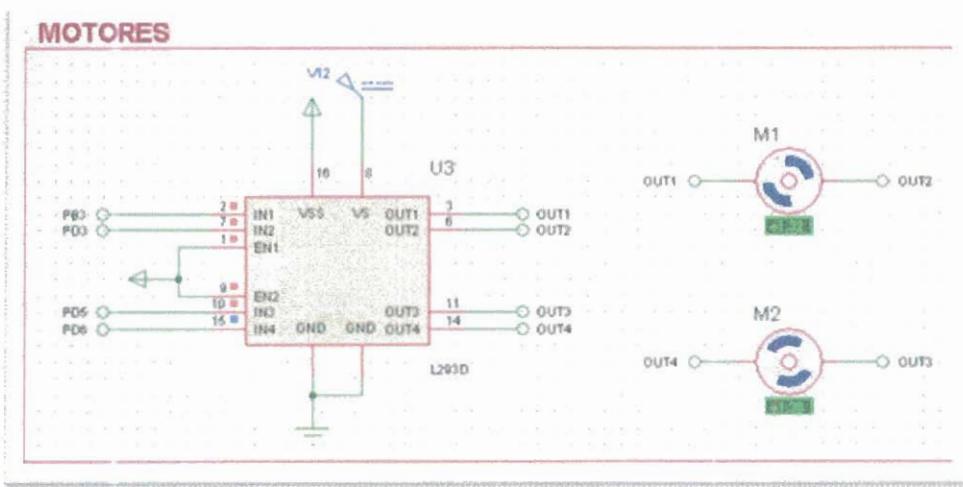


Figura 4.1.2- Simulación de los motores

Para que el robot se mueva hacia atrás al recibir el carácter específico los motores giran en el sentido contrario permitiendo de esa manera que el robot retroceda, así como del mismo sentido para avanzar.

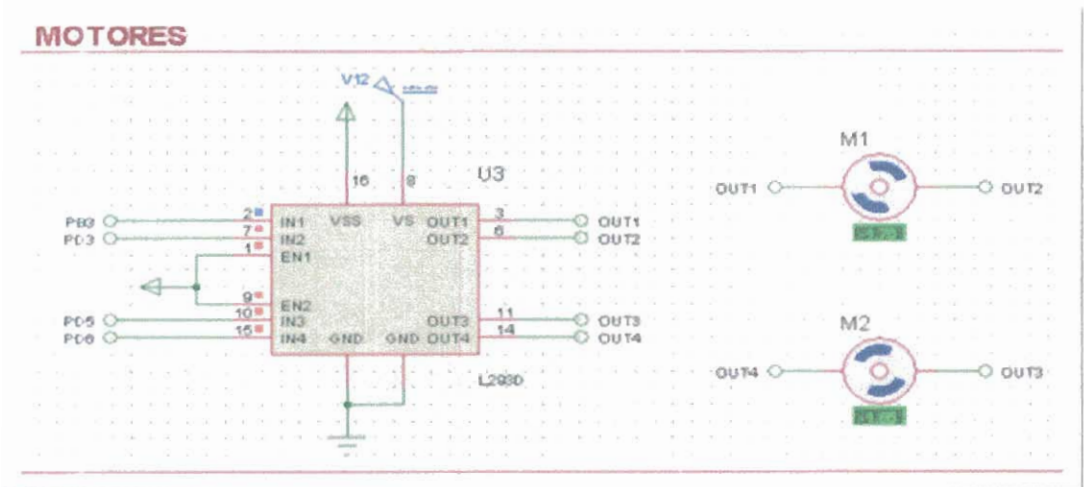


Figura 4.1.3.- Motores ejecutando el desplazamiento hacia atrás

Para realizar los giros correspondientes se debe presionar la dirección a la cual se desea girar presionando la

correspondiente botonera, y el giro consiste en girar un motor en sentido horario y el otro motor en sentido anti horario.

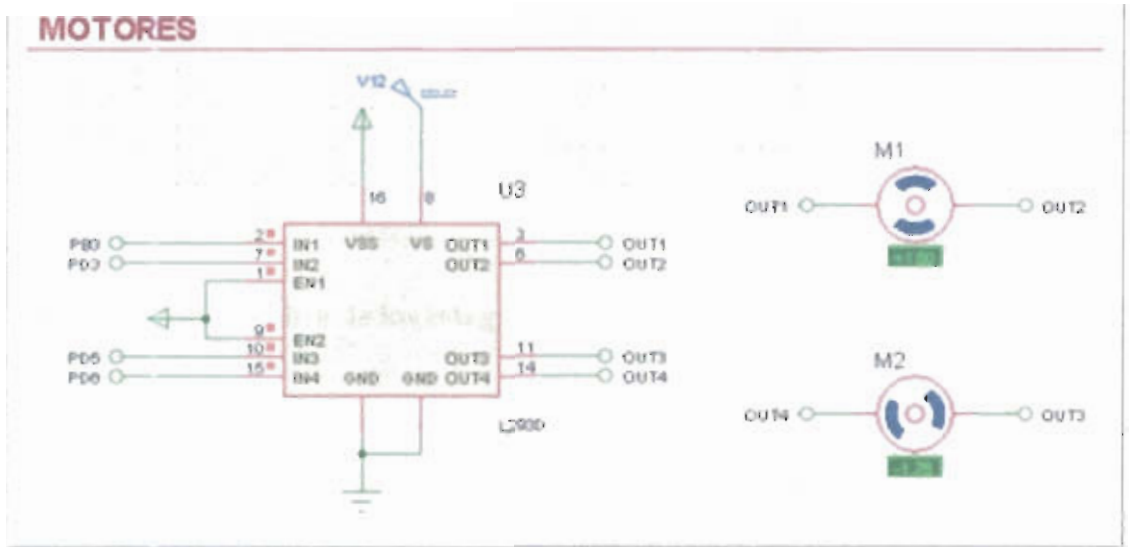


Figura 4.1.4- Motores ejecutando el giro hacia la derecha

4.2 FOTOS DE LOS RESULTADOS

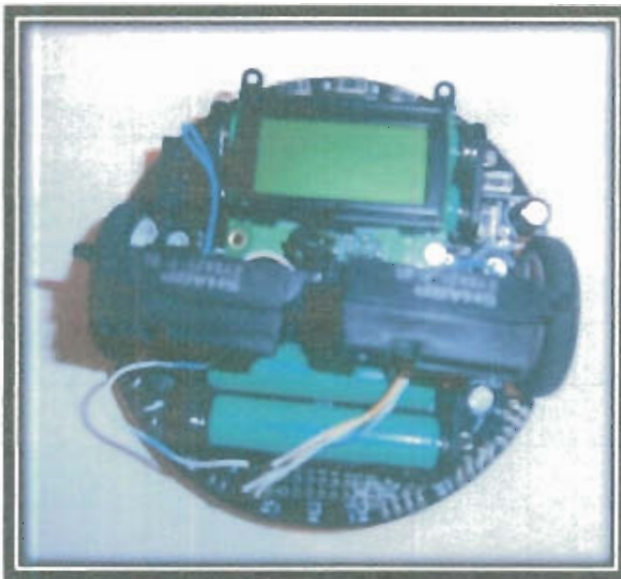


Figura 4.2.1.- Pololu implementado para el desarrollo del proyecto

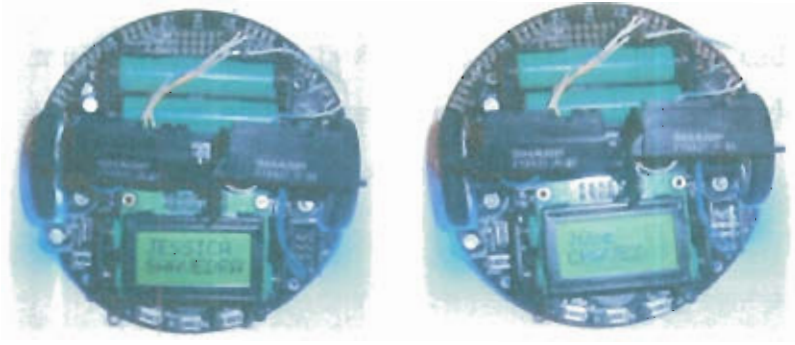


Figura 4.2.2.- Nombre de los integrantes en la pantalla LCD

Al encender el robot mediante el botón POWER del Robot Pololu, en la pantalla LCD se muestran los nombres de los autores del proyecto como se muestra en la figura anterior. Esto lo realiza con un retardo de 1.5 segundos programado según sea conveniente.

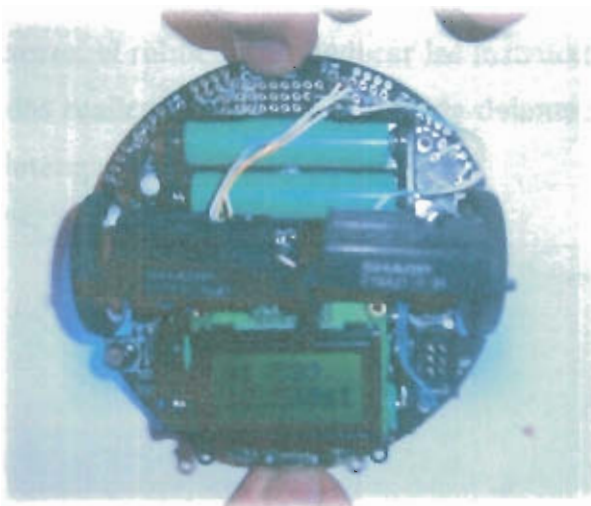


Figura 4.2.3.- Pololu leyendo los sensores S1 y S2

Una vez en movimiento el display o pantalla LCD del robot muestra el valor de la distancia detectada por los cada uno de los sensores: en la parte superior se detalla el valor del sensor S1 y en la parte inferior el valor del sensor S2.

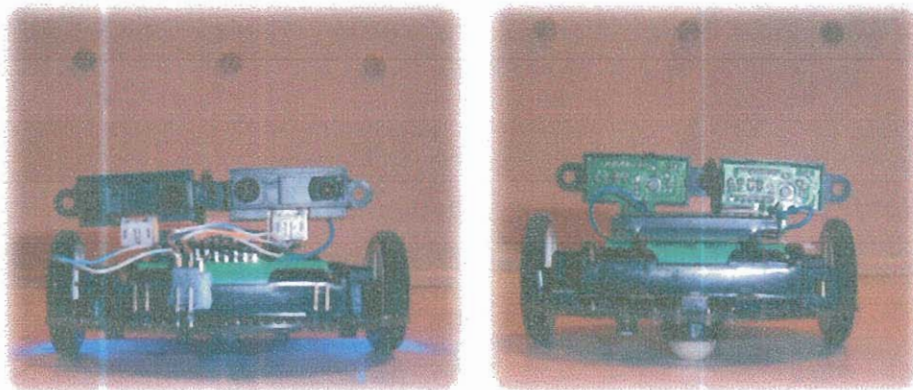


Figura 4.2.4- Implementación de Sensores Sharp al Robot Pololu

Como ya mencionamos de acuerdo a lo que detecten los sensores, el robot va a obedecer las instrucciones para que las ruedas realicen el movimiento hacia delante, hacia atrás o que se detenga.

CONCLUSIONES

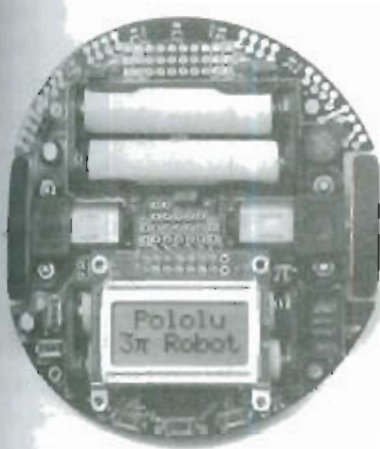
- 1) El proyecto realizado es un robot demostrativo el cual necesita mucho tiempo en su investigación, ya que es un ejemplar que contiene un sin número de funciones, librerías y rutinas que sirven para mejorar su diseño y así convertirse en un producto de tipo comercial.
- 2) Se logró diseñar un sistema que no sólo detecta objetos u obstáculos fijos sino que además divisa los móviles, es decir se programa al robot Pololu para guardar siempre una distancia para que en su recorrido no permita ningún tipo de colisión.
- 3) El uso de drivers capaces de manejar niveles de corriente que un microcontrolador no puede suministrar facilita mucho el desarrollo de proyectos de control de motores como son el control de un robot ya que todas las instrucciones son realizadas por el microcontrolador y el mismo se encarga de enviar las señales a los diferentes drivers para que puedan suministrar el nivel de potencia necesario para dicha labor.
- 4) Se puede tomar a este prototipo como una base para futuros proyectos similares, en los cuales se puede mejorar la forma de detectar obstáculos y evitarlos usando nuevas tecnologías. Ya que este tipo de programación utilizada representa un auxiliar en cuanto a la prevención de accidentes, facilitando el desplazamiento de un sitio a otro con comodidad y tranquilidad.

RECOMENDACIONES

- 1) Se debe tomar en cuenta el medio o ambiente en donde se realicen las pruebas del robot o donde se movilice, ya que los sensores son muy sensibles a la luz y eso impediría la fácil detección de los obstáculos frente al robot, ya que esto hace referencia a los objetos que tendría que esquivar.
- 2) Es preciso tener bajo consideración que el robot solo cuenta con dos sensores frontales es decir se complicaría el movimiento al retroceder ya que primero debería dar el giro y luego el movimiento. Al no contar con un sensor en la parte posterior es preferible buscar un lugar abierto para su ejecución.
- 3) Al trabajar con motores se recomienda no realizar un trabajo a velocidades altas partiendo desde el reposo porque eso genera un gran consumo de potencia durante el arranque, por lo tanto es recomendable ir aumentando la velocidad de manera gradual para un uso más eficiente de la energía.
- 4) La velocidad a la que el robot se mueve es un poco lenta debido a que los sensores necesitan un tiempo para realizar varias funciones como cambiar su valor en la curva, que el ADC haga la conversión análoga - digital y siga con las ordenes del algoritmo previamente cargado al Pololu.

ANEXOS

Robot Pololu 3pi Guía de usuario



Nota: Los robots 3pi que empiezan con el número de serie **0J5840**, se suministran con el nuevo microcontrolador **ATmega328P** en el sitio del ATmega168. El número de serie está escrito en la etiqueta del código de barras situada en la parte posterior de la PCB del 3pi PCB.

El ATmega328 tiene esencialmente lo mismo que el ATmega168 pero va provisto del doble de memoria (32 KB flash, 2 KB RAM, y 1 KB de EEPROM), por lo que el código escrito para uno puede trabajar, con mínimas modificaciones, en el nuevo ATmega328 (la **Pololu AVR Library** (<http://www.pololu.com/docs/0J20>) lleva soporte específico para el ATmega328P).

1. Introducción.....	2
2. Contactando con Pololu.....	2
3. Advertencias de seguridad y precauciones en su manipulación.....	2
4. Empezar con tu robot 3pi.....	3
4.a Que necesitas.....	3
4.b Enciende el 3pi.....	4
4.c Funcionamiento del programa demo pre-instalado.....	4
4.d Accesorios incluidos.....	5
5. Como trabaja el 3pi.....	5
5.a Baterías.....	5
5.b Gestión de la energía.....	6
5.c Motores y engranajes.....	7
5.d Entradas digitales y sensores.....	10
5.e 3pi. Esquema del circuito simplificado.....	11
6. Programando tu 3pi.....	12
6.a Descargar e instalar la Librería C/C++.....	12
6.b Compilando un Programa simple.....	13
7. Ejemplo Project #1: Siguiendo la línea.....	14
7.a Acerca del seguimiento de línea.....	14
7.b Algoritmo para 3pi de seguimiento de línea.....	15
7.c Seguimiento de línea avanzado con 3pi: PID Control.....	19
8. Ejemplo Project #2: Resolución de laberintos.....	20
8.a Resolución de laberinto de línea.....	20
8.b Trabajar con múltiples ficheros C en AVR Studio.....	21
8.c Mano izquierda contra el muro.....	23
8.d Bucle(s) principal.....	23
8.e Simplificando la solución.....	25
8.f Mejorar el código de solución del laberinto.....	27
9. Tablas de asignación de pins.....	30
9.a Tabla de asignación de PINS según función.....	30
9.b Tabla de asignación de PINS por pin.....	30
10. Información para su expansión.....	31
10.a Programa serie para esclavo.....	31
10.b Programa serie para maestro.....	38
10.c I/O disponibles en los 3pi ATmegaxx8.....	42
11. Enlaces relacionados.....	42
3pi kit de expansión.....	43
Ensamblado.....	44

1. Introducción

El 3pi de Pololu es un pequeño robot autónomo de alto rendimiento, designado para competiciones de seguimiento de línea y resolución de laberintos. Alimentado por 4 pilas AAA (no incluidas) y un único sistema de tracción para los motores que trabaja a 9.25V, el 3pi es capaz de velocidades por encima de los 100cm/s mientras realiza vueltas precisas y cambios de sentido que no varían con el voltaje de las baterías. Los resultados son consistentes y están bien sintonizados con el código aún con baterías bajas. El robot está totalmente ensamblado con dos micromotores de metal para las ruedas, cinco sensores de reflexión, una pantalla LCD de 8x2 caracteres, un buzzer, tres pulsadores y más, todo ello conectado a un microcontrolador programable. El 3pi mide aproximadamente 9,5 cm (3,7") de diámetro y pesa alrededor de 83 gr. (2,9 oz.) sin baterías.

El 3pi contiene un microcontrolador Atmel ATmega168 o un ATmega328 (los nombraremos como ATmegaxx8) a 20 MHz con 16KB de memoria flash y 1KB de RAM, el doble (32 KB y 2KB) en el ATmega328 y 1KB de EEPROM. El uso del ATmegaxx8 lo hace compatible con la plataforma de desarrollo Arduino. Las herramientas gratuitas de desarrollo en C y C++ así como un extenso paquete de librerías que pueden trabajar con el hardware que lleva integrado están disponibles. También hemos diseñado simples programas que muestran como trabajan los diferentes componentes del 3pi y que puedes mejorar o crear nuevo código para el seguimiento de línea y laberintos.

Debes tener en cuenta que es *necesario* un PROGRAMADOR AVR ISP externo como el **USB AVR Programmer** [<http://www.pololu.com/catalog/product/1300>] para programar el robot 3pi.

2. Contactando con Pololu

Puedes visitar la página de 3pi [<http://www.pololu.com/catalog/product/975>] para información adicional, fotos, videos, ejemplos de código y otras referencias.

Nos encantaría saber de tus proyectos y sobre tu experiencia con el 3pi robot. Puedes ponerte en contacto con nosotros directamente [<http://www.pololu.com/contact>] o en el foro [<http://forum.pololu.com/>]. Cuéntanos lo que hicimos bien, lo que se podría mejorar, lo que te gustaría ver en el futuro, o compartir tu código con otros usuarios 3pi.

3. Advertencias de seguridad y precauciones en su manipulación

El robot 3pi no es para niños. Los más jóvenes deben usar este producto bajo supervisión de adultos. Mediante el uso de este producto, te compromete a no señalar a Pololu como responsable por cualquier lesión o daños relacionados con el uso incorrecto del mismo producto.

Este producto no está diseñado para jugar y no debe utilizarse en aplicaciones en las que el mal funcionamiento del producto podría causar lesiones o daños. Por favor, tome nota de estas precauciones adicionales:

- **No intentes programar el 3pi con las baterías descargadas**, puedes inutilizarlo de forma permanente. Si le compras baterías recargables, comprueba que estén cargadas. El 3pi puede comprobar su nivel de carga de batería. En los programas de ejemplo hemos previsto esta habilidad y deberías incluirla en tus programas para conocer el momento de realizar la recarga.
- El robot 3pi robot contiene plomo, no lo laves ni le eches líquidos.
- Está diseñado para trabajar en interiores y pequeñas superficies deslizantes.
- Si lo pones en contacto con superficies metálicas la PCB puede estar en contacto y producirse cortocircuitos que dañarían el 3pi.

- Dado que la PCB y sus componentes son electrónicos tome precauciones para protegerlo de ESD (descargas electrostáticas) que podrían dañar sus partes. Cuando toques el 3pi principalmente en ruedas, motores, baterías o contactos de la PCB, puedes haber pequeñas descargas. Si manejas el 3pi con otra persona, primero toca tu mano con la suya para igualar las cargas electrostáticas que podáis tener y estas no descarguen al 3pi.
- Si quitas la LCD asegúrate de que está apagado el 3pi y que la pones en la misma dirección que estaba, encima del extremo de la batería, ya que podrías estropear la LCD o el 3pi. Es posible tener la LCD sin iluminación o parada.

4. Empezar con tu robot 3pi

Para comenzar con tu 3pi solo debes sacarlo de la caja, ponerle pilas y encender. El 3pi se inicia con un programa demo que te muestra el funcionamiento de sus características.

Los siguientes apartados de darán información necesaria para poder hacer que funcione tu 3pi.

Características del robot Pololu 3pi

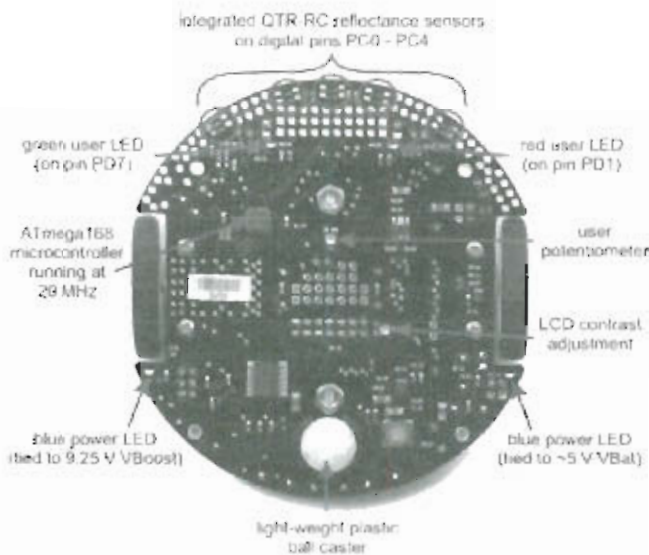
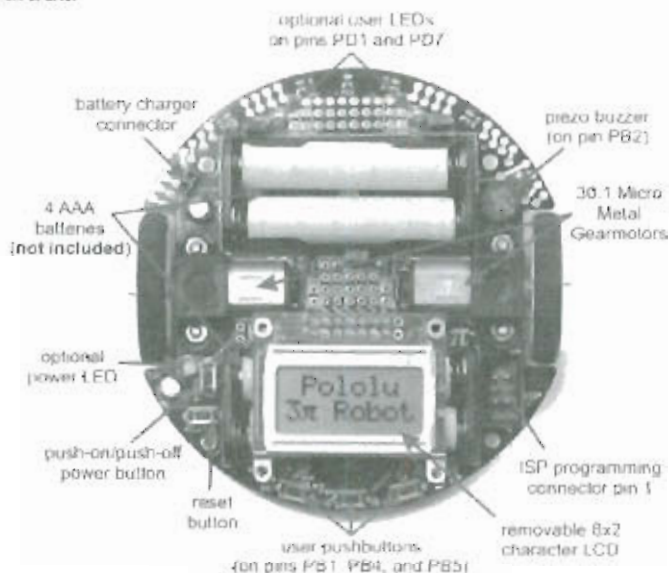
4.a Que necesitas

Son necesarios para empezar:

- **4 pilas AAA.** Cualquier pila AAA puede hacerlo funcionar, pero recomendamos las de NiMH recargables, fáciles de comprar en Pololu u otro comercio. Si usa pilas recargables necesitaras un cargador de baterías. Los cargadores diseñados para conectar un pack de baterías externos pueden usarse con el puerto cargador del 3pi.
- **AVR Conector ISP programador de 6-pin.** Las características del 3pi y el microcontrolador ATmega168 requieren de un programador externo como el programador Pololu Orangután USB o la serie AVRISP de Atmel. El 3pi tiene un conector estándar de 6 pins que permite la programación directa del micro mediante un cable ISP que se conecta al dispositivo programador. (También necesitas un cable USB que conecte el programador con el PC. No está incluido. (Si, que lo tienes en el *paquete combinado de 3pi+programmer*).
- **Ordenador de mesa o portátil.** Necesitas un PC para desarrollar código y grabarlo en el 3pi. Se puede programar en Windows, Mac y Linux, pero el soporte para Mac es limitado.

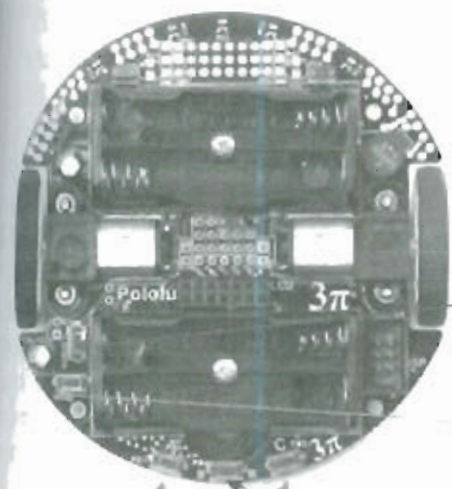
Además, materiales para crear superficies por donde corra el 3pi:

- Hojas de papel largo y gruesas blancas o una pizarra blanca borrable (usadas por los constructores de casas).
- Cinta colores brillantes para rejuntar múltiples hojas a la vez.
- Cinta adhesiva de 3/4" negra para crear líneas a seguir por el robot.



4b Enciende el 3pi

La primera vez que uses el 3pi debes insertar 2+2 pilas AAA en cada sitio. Para ello debes quitar las LCD. Presta atención a la orientación para insertarla después. Con la LCD quitada mira las figuras marcadas a la derecha.



En cuanto estén puestas las pilas, inserta la LCD en su posición encima del paquete de baterías y botones. Fijate bien en los pins que deben quedar cada uno en su conector hembra correspondiente.

Después, pulsa el botón de ENCENDER (a la izquierda al lado del porta pilas) para conectar el 3pi. Verás que dos leds azules se encienden y el 3pi empieza a ejecutar el programa de demo.

Con pulsar el botón de nuevo se apagará el 3pi o pulsa el botón de RESET situado justo mas abajo para resetear el robot mientras está funcionando.

4c Funcionamiento del programa demo pre-instalado

El 3pi viene con un programa pre-instalado de demostración y testeo de sensores, motores, leds y buzzer para ver su correcto funcionamiento.

Cuando se enciende por primera vez oírás un pitido y verás en pantalla "Pololu 3pi Robot" y luego aparece "Demo Program", indicando que está en funcionamiento el mismo. Si oyes el beep pero no aparece nada en la LCD puedes ajustar el contraste de la LCD con el mini potenciómetro que está debajo de la placa. Seguir el programa pulsando el botón B para proceder con el menú principal.

Pulsa A o C para avanzar o retroceder a través del menú y de nuevo B para salir.

Hay siete demos accesibles desde el menú:

Batería: Muestra el voltaje de las pilas en milivoltios, así, si marca 5000 (5.0 V) o más, es porque las baterías están a tope. Removiendo el jumper marcado como ADC6 separa la batería del pin analógico de medida produciendo que se muestre un valor muy bajo.

LEDs: Parpadeo de led verde y rojo que hay bajo la placa o los de usuario si los has puesto.

Trimmer: Muestra la posición del mini potenciómetro trimmer localizado en la parte inferior de la placa marcando un numero entre 0 y 1023. Al mostrar el valor parpadean los LEDs y toca una nota musical cuya frecuencia está en función a la lectura. Puedes hacer girar los micro potenciómetros con pequeño destornillador de 2mm.

Sensores: Muestra las lecturas actuales de los sensores IR mediante un gráfico de barras. Barras grandes indican poca reflexión (negro). Coloca un objeto reflectivo como un dedo sobre uno de los sensores y verás la lectura gráfica correspondiente. Esta demo también muestra, al pulsar C que todos los sensores están activos. En iluminación interior, cerca de bombillas de incandescencia o halógenas los sensores pueden emitir lecturas erróneas debido a la emisión de infrarrojos. Remueve el Jumper PC5 para desactivar el control de los emisores IR lo que servirá para que siempre estén activos.

Motores: Pulsando A o C hará que funcionen cada uno de los motores en su dirección. Si pulsas ambos botones, ambos motores funcionan simultáneamente. Los motores aumentan gradualmente la velocidad; si realizas nuevos programas estudia detenidamente el funcionamiento de aceleración. Pulsa A o C para invertir la dirección del motor

correspondiente (la letra del botón se vuelve minúscula si el motor funciona en sentido contrario).

Música: Toca una melodía de J. S. Bach's Fuga en D Menor en el buzzer, mientras muestra unas notas. Es para mostrar la habilidad de 3pi como músico.

Timer: Un simple reloj contador. Pulsa C para iniciar o parar el reloj y A para reset. El reloj puede seguir contando mientras exploras otras demos.

El código fuente del programa demo está incluido en las librerías de Pololu AVR C/C++ descritas en la sección 5. Después de descargar y desempaquetar las librerías el programa se encuentra en el directorio

4.d Accesorios incluidos

Los robots 3pi se envían con dos LEDs rojos y dos verdes. Tienes tres puntos de conexión para leds opcionales: uno al lado del botón de POWER para indicar cuando el 3pi está encendido y dos puntos más controlables por el usuario en el frontal.



El uso de leds es opcional y el 3pi funciona igual sin ellos. Puedes personalizar tu 3pi con una combinación de verdes y rojos y usarlos para opciones luminosas.

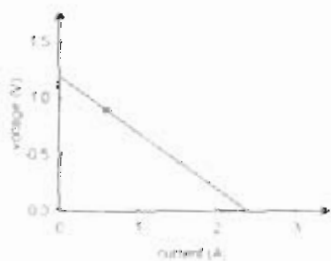
Añadir LEDs es fácil, pero ten en cuenta que si tienes que desoldarlos después, los componentes que se encuentran cerca de donde hagas las soldaduras. Los LEDs tienen polaridad, fíjate, el trozo más largo corresponde al +. Antes de soldar asegúrate de la función que van a realizar y sujeta bien el led y. Recorta el exceso de patas sobrante. El 3pi también viene con cuatro juegos de tres jumpers en colores: azul, rojo, amarillo y negro. Son para personalizarlos si tienes más de un 3pi con diferentes colores.

5. Como trabaja el 3pi

5.a Baterías

Introducción al funcionamiento de las baterías.

La potencia del sistema del 3pi empieza con las baterías, por eso es importante conocer como 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 como 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 cuantos electrones circulan por segundo esto es intensidad que se mide en amperios (A) Una corriente de 1A equivale a 6×10^{18} 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.



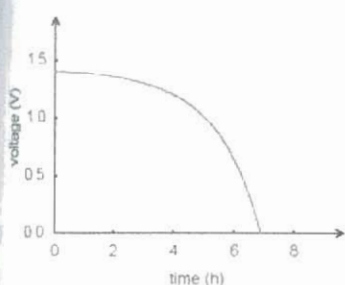
Para cualquier batería en funcionamiento, el voltaje suministrado se reduce con el tiempo bajando hasta perder toda la energía (procurar no llegar a ello, puede producir cortocircuito y quedar

inutilizada para siempre). El gráfico siguiente muestra un modelo de cómo la tensión al aumentar la potencia requerida:

La potencia de una batería se mide multiplicando los voltios por los amperios, dando una medida en vatios ($W=V \times A$).

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 ($A1+A2+\dots$) 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 ($V1+V2+\dots$). 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 conectandolas 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:

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).

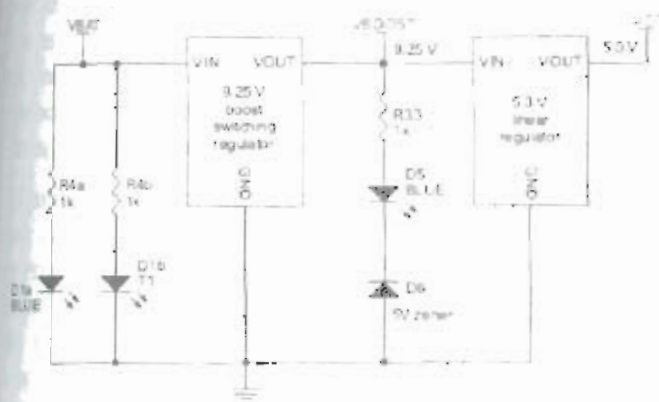
5.b Gestión de la energía

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 esta 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:



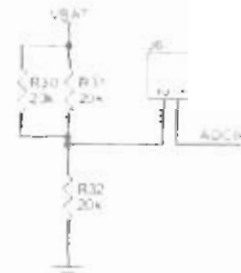
El voltaje de 4xAAA pilas puede variar entre 3,5 a 5,5 voltios (y hasta 6v si se usan alcalinas). Esto no podría ir bien si no fuera por la regulación del voltaje a 5V. Usamos un regulador switching para elevar el voltaje a 9,25 V (Vboost) y reguladores lineales para obtener 5V (VCC). Vboost sirve para 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 90° 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.

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. Usando una conversión analógica de 10 bit un valor de 5V se lee como 1023 y un valor de 3,2 se lee como 655. Para convertir el actual estado de la batería multiplicamos $5000\text{mV} \times 3/2$ y dividimos por 1023. Para ello disponemos de la función `read_battery_millivolts()` función que puede promediar diferentes lecturas y devolver el resultado en mV:



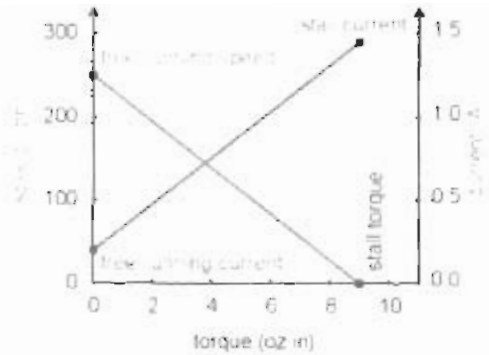
5.c Motores y engranajes

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. El típico motor DC contiene imanes permanentes en el exterior y bobinas electromagnéticas montadas 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 medido 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 esta completamente parado).

Llamamos a esto, funcionamiento a velocidad libre y al par de parada. Naturalmente, el motor usa el mínimo de corriente cuando no se aplica una fuerza, y si la corriente que viene de la batería aumenta es por fuerzas de rozamiento o engranajes de modo que son parámetros importantes del motor. Según se muestra en el siguiente gráfico:



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 de 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.

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.

Engranaje:	30:1
Velocidad libre:	700 rpm
Consumo mín:	60 mA
Par máximo:	6 oz·in
Consumo máx:	540 mA

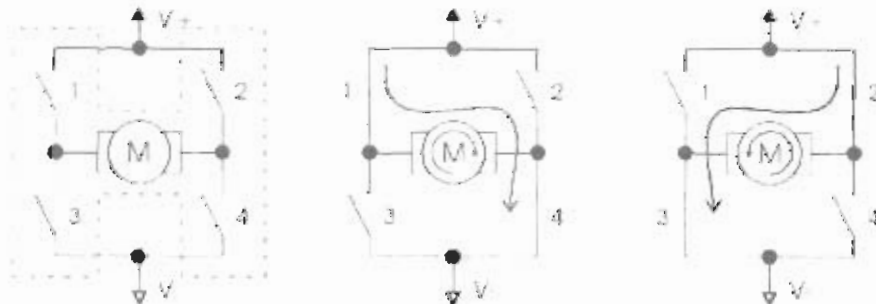
Las dos ruedas del 3pi tienen una radio de 0.67 inch, con lo que la máxima fuerza que pueden producir los dos motores en funcionamiento será de $2 \times 6 / 0.67 = 18$ oz.



El 3pi pesa 7 oz con las pilas insertadas y estos motores son lo suficientemente fuertes como para moverlo en una pendiente de 2g (2 veces la gravedad). El rendimiento está limitado por la fricción de las gomas, podemos deducir que puede trabajar con pendientes de entre 30° a 40°.

Mover un motor con control de velocidad y dirección.

Una cosa que tienen los motores DC es que para cambiar de dirección de rotación debe alternar la polaridad del voltaje aplicado. Como es lógico no cambiaremos la conexión de pilas ni la de los motores para tener un control de dirección. Para eso se usan los llamados puentes H como muestra el diagrama.



Los cuatro puntos de corte de corriente permiten el cambio de sentido. Observando las figuras se deduce su funcionamiento. Los puentes en H se construyen mediante transistores que realizan la funciones de los interruptores.

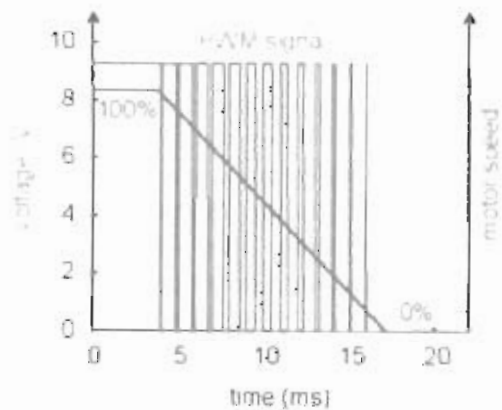
Se usan puentes para ambos motores en el 3pi mediante el chip TB6612FNG conectando las salidas de los puertos del micro-controlador correspondientes a los pins PD5 y PD6 para el motor M1 y para el motor M2 se utilizan los pins de control en PD3 y PB3.

Podemos ver su funcionamiento en la tabla siguiente:

PD5	PD6	1	2	3	4	M1	PD3	PB3	1	2	3	4	M2
0	0	off	off	off	off	off (coast)	0	0	off	off	off	off	off (coast)
0	1	off	on	on	off	forward	0	1	off	on	on	off	forward
1	0	on	off	off	on	reverse	1	0	on	off	off	on	reverse
1	1	off	off	on	on	off (brake)	1	1	off	off	on	on	off (brake)

La velocidad se consigue alternando pulsos altos y bajos. Supongamos que PD6 está alto (a 5 V, la lógica será "1") y alternativamente el PD5 está en bajo (0 V es decir "0") y alto. El motor funcionará entre "adelante" y "paro" causando un descenso de velocidad en el motor M1. Por ejemplo, si PD6 está en alto 2/3 del tiempo (67% del ciclo de trabajo) el motor rodará aproximadamente al 67% de su velocidad total. Dado que el voltaje suministrado al motor es una serie de pulsos de anchura variable a este método de control de velocidad se le llama modulación del ancho de pulso (PWM). Un ejemplo de PWM se muestra en el gráfico: el ancho de los pulsos decrece desde el 100% del ciclo de trabajo hasta el 0%, por lo que el motor rodará desde el máximo de velocidad hasta pararse.

En el 3pi, el control de velocidad se consigue usando las salidas de PWM del microcontrolador que generan los temporizadores Timer0 y Timer2. Esto significa que puedes establecer el ciclo de trabajo PWM para los dos motores de una vez e independiente del resto de código por lo que seguirá produciendo señales en segundo plano, pudiendo prestar atención a otras necesidades.



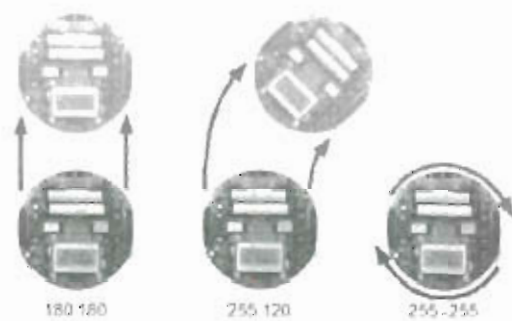
La función `set_motors()` de la librería AVR Pololu (ver sección 6.a) crea el ciclo de trabajo usando una precisión de 8 bits por lo que un valor de 255 corresponderá al 100%. Por ejemplo para una velocidad del 67% en el M1 y otra del 33% en el M2 llamaremos a la función de la siguiente forma:

```
set_motors(171, 84)
```

Para obtener un descenso lento de la secuencia del PWM fíjate en el gráfico, deberás escribir un bucle que gradualmente haga decrecer la velocidad del motor en el tiempo.

Girando con una conducción diferencial

El 3pi tiene motores independientes a cada lado que crean un método de conducción denominado conducción diferencial. También se conoce como "conducción de tanques". Para girar mediante este método es necesario hacer rodar los motores a diferentes velocidades. En el ejemplo de función anterior la rueda izquierda se mueve más deprisa que la derecha con lo que el robot avanza girando a la derecha. La diferencia de velocidades determina que el giro sea más suave o más brusco, e incluso moviendo un motor adelante y el otro atrás se consigue un cambio de dirección total.



Features

- High Performance, Low Power AVR[®] 8-Bit Microcontroller
- Advanced RISC Architecture
- 131 Powerful Instructions – Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 20 MIPS Throughput at 20 MHz
- On-chip 2-cycle Multiplier
- Endurance Non-volatile Memory Segments
 - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
 - 256/512/1K Bytes EEPROM
 - 512/1K/1K/2K Bytes Internal SRAM
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- Data retention: 20 years at 85°C/100 years at 25°C⁽¹⁾
- Optional Boot Code Section with Independent Lock Bits
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
- Programming Lock for Software Security
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Six PWM Channels
 - 8-channel 10-bit ADC in TQFP and QFN/MLF package
 - Temperature Measurement
 - 6-channel 10-bit ADC in PDIP Package
 - Temperature Measurement
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- Pin and Packages
 - 13 Programmable I/O Lines
 - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
 - 1.8 - 5.5V
- Temperature Range:
 - -40°C to 85°C
- Lead Grade:
 - 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Power Consumption at 1 MHz, 1.8V, 25°C
 - Active Mode: 0.2 mA
 - Power-down Mode: 0.1 µA
 - Power-save Mode: 0.75 µA (including 32 kHz RTC)



8-bit AVR[®] Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48A
ATmega48PA
ATmega88A
ATmega88PA
ATmega168A
ATmega168PA
ATmega328
ATmega328P

Summary

Rev. B271CS-AVR-06/10



ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

Pin Configurations

Figure 1-1. Pinout ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

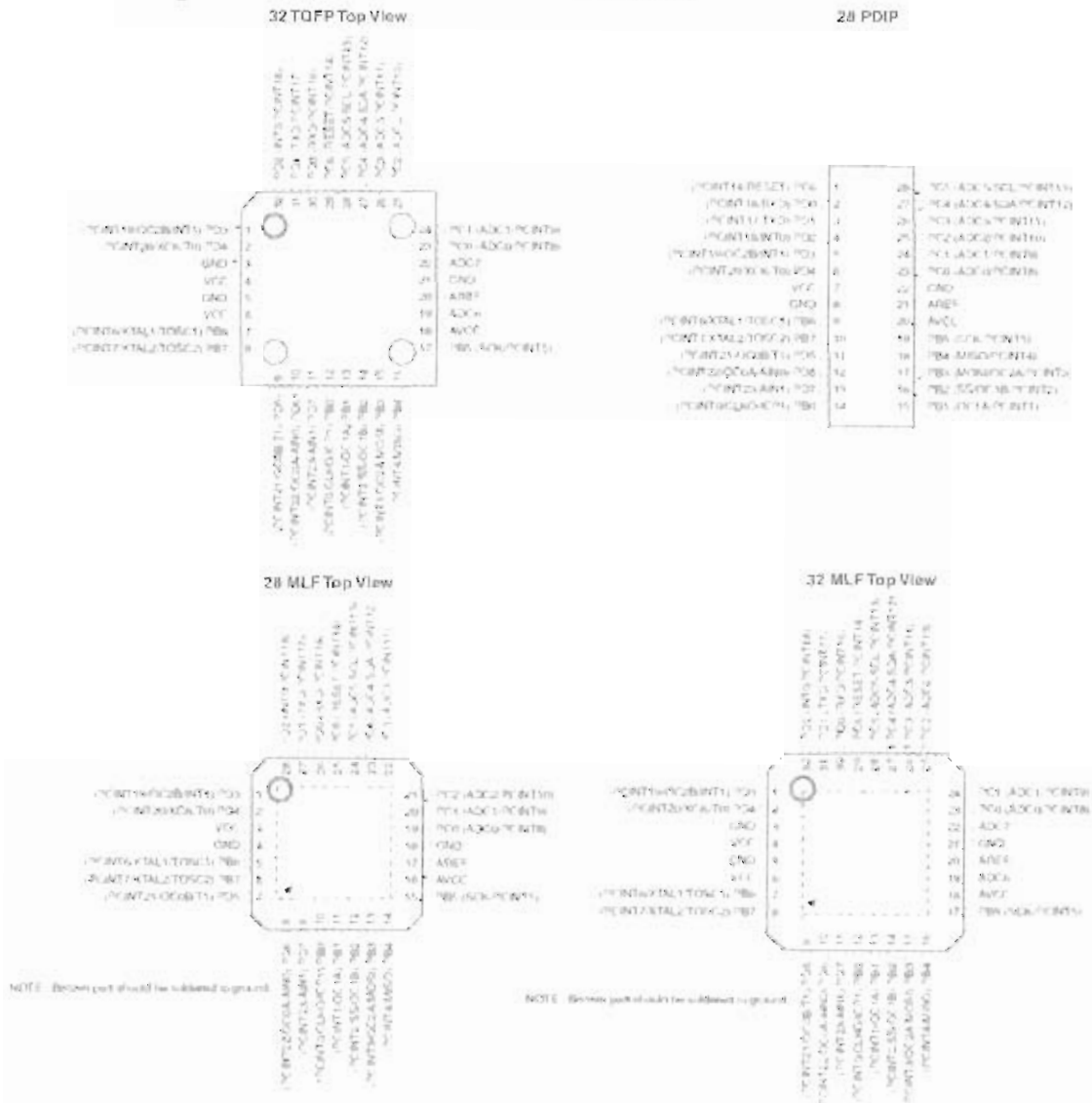


Table 1-1. 32UBGA - Pinout ATmega48A/48PA/88A/88PA/168A/168PA

	1	2	3	4	5	6
A	PD2	PD1	PC6	PC4	PC2	PC1
B	PD3	PD4	PD0	PC5	PC3	PC0
C	GND	GND			ADC7	GND
D	VDD	VDD			AREF	ADC6
E	PB6	PD6	PB0	PB2	AVDD	PB5
F	PB7	PD5	PD7	PB1	PB3	PB4



Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7...6 is used as TOSC2...1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in "System Clock and Clock Options" on page 26.

Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5...0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/RESET

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-12 on page 323. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in "Alternate Functions of Port C" on page 86.

Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.



The various special features of Port D are elaborated in "Alternate Functions of Port D" on page 89.

7 AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC6...4 use digital supply voltage, V_{CC} .

8 AREF

AREF is the analog reference pin for the A/D Converter.

9 ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

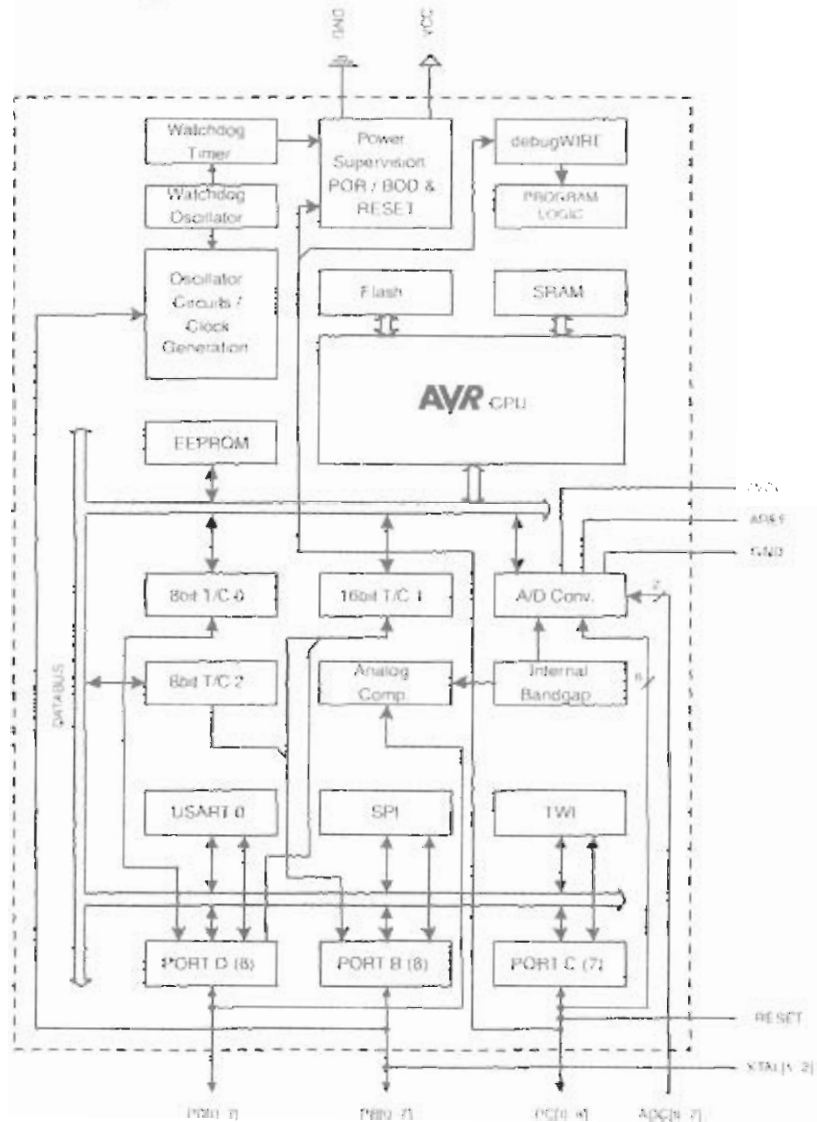


Overview

The ATmega48A/48PA/88A/88PA/168A/168PA/328/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48A/48PA/88A/88PA/168A/168PA/328/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent

registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega48A/48PA/88A/88PA/168A/168PA/328/328P provides the following features: 4K/8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 256/512/512/1K bytes EEPROM, 512/1K/1K/2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with in-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega48A/48PA/88A/88PA/168A/168PA/328/328P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega48A/48PA/88A/88PA/168A/168PA/328/328P AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2.2 Comparison Between Processors

The ATmega48A/48PA/88A/88PA/168A/168PA/328/328P differ only in memory sizes, boot loader support, and interrupt vector sizes. Table 2-1 summarizes the different memory and interrupt vector sizes for the devices.

Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88A	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168A	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector



Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector
ATmega328	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector

ATmega48A/48PA/88A/88PA/168A/168PA/328/328P support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega 48A/48PA there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

Resources

A comprehensive set of development tools, application notes and datasheets are available for download on <http://www.atmel.com/avr>.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0xFF	Reserved	--	--	--	--	--	--	--	--	
0xFE	Reserved	--	--	--	--	--	--	--	--	
0xFD	Reserved	--	--	--	--	--	--	--	--	
0xFC	Reserved	--	--	--	--	--	--	--	--	
0xFB	Reserved	--	--	--	--	--	--	--	--	
0xFA	Reserved	--	--	--	--	--	--	--	--	
0xF9	Reserved	--	--	--	--	--	--	--	--	
0xF8	Reserved	--	--	--	--	--	--	--	--	
0xF7	Reserved	--	--	--	--	--	--	--	--	
0xF6	Reserved	--	--	--	--	--	--	--	--	
0xF5	Reserved	--	--	--	--	--	--	--	--	
0xF4	Reserved	--	--	--	--	--	--	--	--	
0xF3	Reserved	--	--	--	--	--	--	--	--	
0xF2	Reserved	--	--	--	--	--	--	--	--	
0xF1	Reserved	--	--	--	--	--	--	--	--	
0xF0	Reserved	--	--	--	--	--	--	--	--	
0xEF	Reserved	--	--	--	--	--	--	--	--	
0xEE	Reserved	--	--	--	--	--	--	--	--	
0xED	Reserved	--	--	--	--	--	--	--	--	
0xEC	Reserved	--	--	--	--	--	--	--	--	
0xEB	Reserved	--	--	--	--	--	--	--	--	
0xEA	Reserved	--	--	--	--	--	--	--	--	
0xE9	Reserved	--	--	--	--	--	--	--	--	
0xE8	Reserved	--	--	--	--	--	--	--	--	
0xE7	Reserved	--	--	--	--	--	--	--	--	
0xE6	Reserved	--	--	--	--	--	--	--	--	
0xE5	Reserved	--	--	--	--	--	--	--	--	
0xE4	Reserved	--	--	--	--	--	--	--	--	
0xE3	Reserved	--	--	--	--	--	--	--	--	
0xE2	Reserved	--	--	--	--	--	--	--	--	
0xE1	Reserved	--	--	--	--	--	--	--	--	
0xE0	Reserved	--	--	--	--	--	--	--	--	
0xDF	Reserved	--	--	--	--	--	--	--	--	
0xDE	Reserved	--	--	--	--	--	--	--	--	
0xDD	Reserved	--	--	--	--	--	--	--	--	
0xDC	Reserved	--	--	--	--	--	--	--	--	
0xDB	Reserved	--	--	--	--	--	--	--	--	
0xDA	Reserved	--	--	--	--	--	--	--	--	
0xD9	Reserved	--	--	--	--	--	--	--	--	
0xD8	Reserved	--	--	--	--	--	--	--	--	
0xD7	Reserved	--	--	--	--	--	--	--	--	
0xD6	Reserved	--	--	--	--	--	--	--	--	
0xD5	Reserved	--	--	--	--	--	--	--	--	
0xD4	Reserved	--	--	--	--	--	--	--	--	
0xD3	Reserved	--	--	--	--	--	--	--	--	
0xD2	Reserved	--	--	--	--	--	--	--	--	
0xD1	Reserved	--	--	--	--	--	--	--	--	
0xD0	Reserved	--	--	--	--	--	--	--	--	
0xCF	Reserved	--	--	--	--	--	--	--	--	
0xCE	Reserved	--	--	--	--	--	--	--	--	
0xCD	Reserved	--	--	--	--	--	--	--	--	
0xCC	Reserved	--	--	--	--	--	--	--	--	
0xCB	Reserved	--	--	--	--	--	--	--	--	
0xCA	Reserved	--	--	--	--	--	--	--	--	
0xC9	Reserved	--	--	--	--	--	--	--	--	
0xC8	Reserved	--	--	--	--	--	--	--	--	
0xC7	Reserved	--	--	--	--	--	--	--	--	
0xC6	UDFR									196
0xC5	UBRR0H									200
0xC4	UBRR0L									200
0xC3	Reserved	--	--	--	--	--	--	--	--	
0xC2	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSR0SIFR0SD	UCSR0SIFR0SM0	UCPOL0	198/213



ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page	
0x7F	DIDR1	-	-	-	-	-	-	AIN10	AIN00	250	
0x7E	DIDR0	-	-	ADC50	ADC40	ADC30	ADC20	ADC10	ADC00	267	
0x7D	Reserved	-	-	-	-	-	-	-	-		
0x7C	ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0	263	
0x7B	ADCSRB	-	ACME	-	-	-	ADTS2	ADTS1	ADTS0	266	
0x7A	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	264	
0x79	ADCH	ADC Data Register High byte									266
0x78	ADCL	ADC Data Register Low byte									266
0x77	Reserved	-	-	-	-	-	-	-	-		
0x76	Reserved	-	-	-	-	-	-	-	-		
0x75	Reserved	-	-	-	-	-	-	-	-		
0x74	Reserved	-	-	-	-	-	-	-	-		
0x73	Reserved	-	-	-	-	-	-	-	-		
0x72	Reserved	-	-	-	-	-	-	-	-		
0x71	Reserved	-	-	-	-	-	-	-	-		
0x70	TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2	164	
0x6F	TIMSK1	-	-	ICE1	-	-	OCIE1B	OCIE1A	TOIE1	140	
0x6E	TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	112	
0x6D	PCMSK2	PCINT3	PCINT2	PCINT1	PCINT0	PCINT19	PCINT18	PCINT17	PCINT16	75	
0x6C	PCMSK1	-	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	75	
0x6B	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	75	
0x6A	Reserved	-	-	-	-	-	-	-	-		
0x69	EICRA	-	-	-	-	-	-	-	-		
0x68	PCICR	-	-	-	-	ISC11	ISC10	ISC01	ISC00	72	
0x67	Reserved	-	-	-	-	-	PCIE2	PCIE1	PCIE0		
0x66	OSCCAL	Oscillator Calibration Register									37
0x65	Reserved	-	-	-	-	-	-	-	-		
0x64	PRR	PRTW1	PRTM2	PRTM0	-	PRTM1	PRSP1	PRUSART0	PRADC	42	
0x63	Reserved	-	-	-	-	-	-	-	-		
0x62	Reserved	-	-	-	-	-	-	-	-		
0x61	CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	37	
0x60	WDTCR	WDFR	WDIE	WDP3	WDCE	WDIE	WDP2	WDP1	WDP0	56	
0x5F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9	
0x5E (0x5E)	SPH	-	-	-	-	(SP10) ¹	SP9	SP8	SP7	12	
0x5D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12	
0x5C (0x5C)	Reserved	-	-	-	-	-	-	-	-		
0x5B (0x5B)	Reserved	-	-	-	-	-	-	-	-		
0x5A (0x5A)	Reserved	-	-	-	-	-	-	-	-		
0x59 (0x59)	Reserved	-	-	-	-	-	-	-	-		
0x58 (0x58)	Reserved	-	-	-	-	-	-	-	-		
0x57 (0x57)	SPMCSR	SPME	(RWS0) ¹	-	(RWS1) ¹	BURSET	PWRT	PGERS	SELPRGEN	294	
0x56 (0x56)	Reserved	-	-	-	-	-	-	-	-		
0x55 (0x55)	MCUCR	-	BODS ¹	BODSE ¹	FUD	-	-	IVSEL	IVCE	45/69/93	
0x54 (0x54)	MCUSR	-	-	-	-	WDRF	BORF	EXTRF	PORF	55	
0x53 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	40	
0x52 (0x52)	Reserved	-	-	-	-	-	-	-	-		
0x51 (0x51)	Reserved	-	-	-	-	-	-	-	-		
0x50 (0x50)	ACSR	AC0	ACBG	AC0	AC1	ACIE	ACIC	ACIS1	ACIS0	248	
0x4F (0x4F)	Reserved	-	-	-	-	-	-	-	-		
0x4E (0x4E)	SPDR	SPI Data Register									176
0x4D (0x4D)	SPSR	SPIF	WCOL	-	-	-	-	-	SP2X	175	
0x4C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	174	
0x4B (0x4B)	GPDR2	General Purpose I/O Register 2									25
0x4A (0x4A)	GPDR1	General Purpose I/O Register 1									25
0x49 (0x49)	Reserved	-	-	-	-	-	-	-	-		
0x48 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B									
0x47 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A									
0x46 (0x46)	TCNT0	Timer/Counter0 (8-bit)									
0x45 (0x45)	TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00		
0x44 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00		
0x43 (0x43)	GTCCR	1SM	-	-	-	-	-	PSRASY	PSRSYNC	144/166	
0x42 (0x42)	EEARH	EEPROM Address Register High Byte ¹									21
0x41 (0x41)	EEARL	EEPROM Address Register Low Byte									21
0x40 (0x40)	EEDR	EEPROM Data Register									21
0x3F (0x3F)	ECCR	-	-	EEMF1	EEMF0	EERIE	EEMPE	EERE	EERE	21	
0x3E (0x3E)	GPDR0	General Purpose I/O Register 0									25



ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x10 (0x30)	EIFSR	-	-	-	-	-	-	INT1	INT0	73
0x1C (0x3C)	EIFR	-	-	-	-	-	-	INTF1	INTF0	73
0x18 (0x38)	PCIFR	-	-	-	-	-	PCIF2	PCIF1	PCIF0	
0x1A (0x3A)	Reserved	-	-	-	-	-	-	-	-	
0x19 (0x39)	Reserved	-	-	-	-	-	-	-	-	
0x18 (0x38)	Reserved	-	-	-	-	-	-	-	-	
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	164
0x16 (0x36)	TIFR1	-	-	JCF1	-	-	OCF1B	OCF1A	TOV1	140
0x15 (0x35)	WFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-	
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-	
0x11 (0x31)	Reserved	-	-	-	-	-	-	-	-	
0x10 (0x30)	Reserved	-	-	-	-	-	-	-	-	
0x0F (0x2F)	Reserved	-	-	-	-	-	-	-	-	
0x0E (0x2E)	Reserved	-	-	-	-	-	-	-	-	
0x0D (0x2D)	Reserved	-	-	-	-	-	-	-	-	
0x0C (0x2C)	Reserved	-	-	-	-	-	-	-	-	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	94
0x0A (0x2A)	DDRD	DD07	DD06	DD05	DD04	DD03	DD02	DD01	DD00	94
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	94
0x08 (0x28)	PORTC	-	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	93
0x07 (0x27)	DDRC	-	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	93
0x06 (0x26)	PINC	-	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	93
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	93
0x04 (0x24)	DDRB	DD07	DD06	DD05	DD04	DD03	DD02	DD01	DD00	93
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	93
0x02 (0x22)	Reserved	-	-	-	-	-	-	-	-	
0x01 (0x21)	Reserved	-	-	-	-	-	-	-	-	
0x00 (0x20)	Reserved	-	-	-	-	-	-	-	-	

- Note:
- For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
 - I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
 - Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
 - When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48A/48PA/88A/88PA/168A/168PA/328/328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
 - Only valid for ATmega88A/88PA/168A/168PA/328/328P.
 - BODS and BODSE only available for picoPower devices ATmega48PA/88PA/168PA/328P



5. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADW	Rd,K	Add Immediate to Word	$RdH, RdL \leftarrow RdH, RdL + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBW	Rd,K	Subtract Immediate from Word	$RdH, RdL \leftarrow RdH, RdL - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \wedge Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \wedge K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
XOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \text{NOT } Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \text{NOT } Rd + 1$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \wedge (\text{NOT } K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \wedge Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \wedge \text{NOT } Rd$	Z,N,V	1
SEB	Rd	Set Register	$Rd \leftarrow \text{NOT } Rd$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rr \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
BRANCH INSTRUCTIONS					
RMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{STACK}$	None	4
RETI		Interrupt Return	$PC \leftarrow \text{STACK}$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	$\text{if } (Rd - Rr) \text{ PC} \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z,N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z,N,V,C,H	1
CP	Rd,K	Compare Register with Immediate	$Rd - K$	Z,N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	$\text{if } (Rr[b] = 0) \text{ PC} \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	$\text{if } (Rr[b] = 1) \text{ PC} \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBC	P, b	Skip if Bit in I/O Register Cleared	$\text{if } (P[b] = 0) \text{ PC} \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBS	P, b	Skip if Bit in I/O Register is Set	$\text{if } (P[b] = 1) \text{ PC} \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	$\text{if } (SREG[s] = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	$\text{if } (SREG[s] = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch if Equal	$\text{if } (Z = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRNE	k	Branch if Not Equal	$\text{if } (Z = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRCS	k	Branch if Carry Set	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRCC	k	Branch if Carry Cleared	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRSH	k	Branch if Same or Higher	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRDL	k	Branch if Lower	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRMI	k	Branch if Minus	$\text{if } (N = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRPL	k	Branch if Plus	$\text{if } (N = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	$\text{if } (N \oplus V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	$\text{if } (N \oplus V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRHS	k	Branch if Half Carry Flag Set	$\text{if } (H = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	$\text{if } (H = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRIS	k	Branch if I Flag Set	$\text{if } (I = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRIC	k	Branch if I Flag Cleared	$\text{if } (I = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRVS	k	Branch if Overflow Flag is Set	$\text{if } (V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	$\text{if } (V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1/2



ATmega48A/48PA/88A/88PA/168A/168PA/328/328P

Mnemonic	Operands	Description	Operation	Flags	#Clocks
BRE	k	Branch if Interrupt Enabled	$\#(I - 1) \text{ then } PC = PC + k + 1$	None	1/2
BRD	k	Branch if Interrupt Disabled	$\#(I - 0) \text{ then } PC = PC + k + 1$	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
BS	P,b	Set Bit in I/O Register	$IO(P,b) = 1$	None	2
CB	P,b	Clear Bit in I/O Register	$IO(P,b) = 0$	None	2
LSL	Rd	Logical Shift Left	$Rd(n+1) = Rd(n), Rd(0) = 0$	Z,C,N,V	1
LSR	Rd	Logical Shift Right	$Rd(n) = Rd(n+1), Rd(7) = 0$	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	$Rd(0) = C, Rd(n+1) = Rd(n), C = Rd(7)$	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	$Rd(7) = C, Rd(n) = Rd(n+1), C = Rd(0)$	Z,C,N,V	1
RRR	Rd	Arithmetic Shift Right	$Rd(n) = Rd(n+1), n=0..6$	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	$Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)$	None	1
SFET	s	Flag Set	$SREG(s) = 1$	SREG(s)	1
SCLR	s	Flag Clear	$SREG(s) = 0$	SREG(s)	1
ST	Rr, b	Bit Store from Register to T	$T = Rr(b)$	T	1
LD	Rd, b	Bit Load from T to Register	$Rd(b) = T$	None	1
SC		Set Carry	$C = 1$	C	1
CC		Clear Carry	$C = 0$	C	1
SN		Set Negative Flag	$N = 1$	N	1
CN		Clear Negative Flag	$N = 0$	N	1
SZ		Set Zero Flag	$Z = 1$	Z	1
CZ		Clear Zero Flag	$Z = 0$	Z	1
SEI		Global Interrupt Enable	$I = 1$	I	1
CLI		Global Interrupt Disable	$I = 0$	I	1
SES		Set Signed Test Flag	$S = 1$	S	1
CLS		Clear Signed Test Flag	$S = 0$	S	1
SEV		Set Two's Complement Overflow	$V = 1$	V	1
CLV		Clear Two's Complement Overflow	$V = 0$	V	1
SET		Set T in SREG	$T = 1$	T	1
CLT		Clear T in SREG	$T = 0$	T	1
SEH		Set Half Carry Flag in SREG	$H = 1$	H	1
CLH		Clear Half Carry Flag in SREG	$H = 0$	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	$Rd = Rr$	None	1
MOVW	Rd, Rr	Copy Register Word	$Rd = 1, Rr = Rr + 1, Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd = K$	None	1
LDD	Rd, X	Load Indirect	$Rd = (X)$	None	2
LDD	Rd, X+	Load Indirect and Post-Inc.	$Rd = (X), X = X + 1$	None	2
LDD	Rd, -X	Load Indirect and Pre-Dec.	$X = X - 1, Rd = (X)$	None	2
LDD	Rd, Y	Load Indirect	$Rd = (Y)$	None	2
LDD	Rd, Y+	Load Indirect and Post-Inc.	$Rd = (Y), Y = Y + 1$	None	2
LDD	Rd, -Y	Load Indirect and Pre-Dec.	$Y = Y - 1, Rd = (Y)$	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	$Rd = (Y + q)$	None	2
LDD	Rd, Z	Load Indirect	$Rd = (Z)$	None	2
LDD	Rd, Z+	Load Indirect and Post-Inc.	$Rd = (Z), Z = Z + 1$	None	2
LDD	Rd, -Z	Load Indirect and Pre-Dec.	$Z = Z - 1, Rd = (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd = (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd = (k)$	None	2
ST	X, Rr	Store Indirect	$(X) = Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	$(X) = Rr, X = X + 1$	None	2
ST	X-, Rr	Store Indirect and Pre-Dec.	$X = X - 1, (X) = Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) = Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) = Rr, Y = Y + 1$	None	2
ST	Y-, Rr	Store Indirect and Pre-Dec.	$Y = Y - 1, (Y) = Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y + q) = Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) = Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	$(Z) = Rr, Z = Z + 1$	None	2
ST	Z-, Rr	Store Indirect and Pre-Dec.	$Z = Z - 1, (Z) = Rr$	None	2
STD	Z+q, Rr	Store Indirect with Displacement	$(Z + q) = Rr$	None	2
STS	k, Rr	Store Direct to SRAM	$(k) = Rr$	None	2
LPM		Load Program Memory	$Rd = (Z)$	None	3
LPM	Rd, Z	Load Program Memory	$Rd = (Z)$	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc.	$Rd = (Z), Z = Z + 1$	None	3
SPM		Store Program Memory	$(Z) = Rr, Rr$	None	3
IN	Rd, P	In Port	$Rd = P$	None	1
OUT	P, Rr	Out Port	$P = Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK = Rr$	None	2



Mnemonics	Operands	Description	Operation	Flags	#Clocks
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Note: 1. These instructions are only available in ATmega168PA and ATmega328P.



GP2Y0A02YK

Long Distance Measuring Sensor

■ Features

1. Less influence on the colors of reflected objects and their reflectivity, due to optical triangle measuring method
2. Distance output type
(Detection range: 20 to 150cm)
3. An external control circuit is not necessary
Output can be connected directly to a microcomputer

■ Applications

1. For detection of human body and various types of objects in home appliances, OA equipment, etc

■ Absolute Maximum Ratings (T_a=25°C)

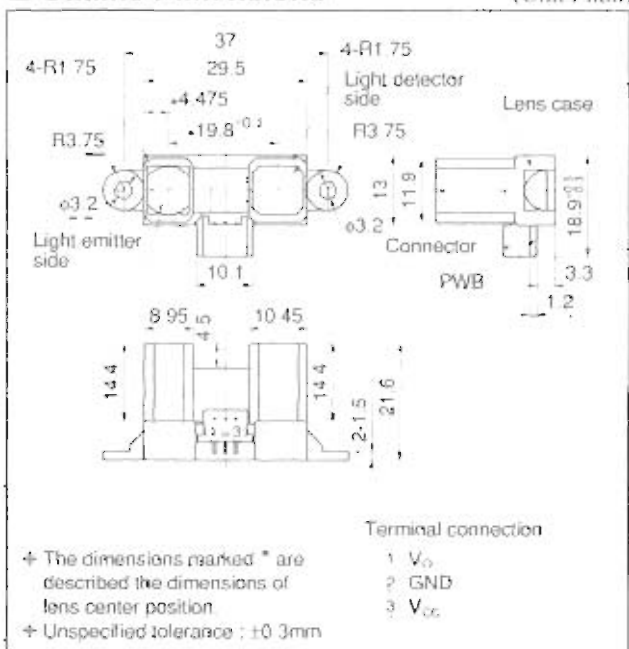
Parameter	Symbol	Rating	Unit
Supply voltage	V _{CC}	-0.3 to +7	V
*1 Output terminal voltage	V _O	-0.3 to V _{CC} +0.3	V
Operating temperature	T _{op}	-10 to +60	°C
Storage temperature	T _{stg}	-40 to +70	°C

*1 Open collector output

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating Supply voltage	V _{CC}	4.5 to 5.5	V

■ Outline Dimensions (Unit : mm)



Electro-optical Characteristics

($T_a=25^\circ\text{C}$, $V_{CC}=5\text{V}$)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range	ΔL	¹⁾ ²⁾	20	-	150	cm
Output terminal voltage	V_O	²⁾ $L=150\text{cm}$	0.25	0.4	0.55	V
Difference of output voltage	ΔV_O	²⁾ Output change at $L=150\text{cm}$ to 20cm	1.8	2.05	2.3	V
Average dissipation current	I_{CC}	-	-	3.3	50	mA

Note: 1) Distance to reflective object

2) Using reflective object White paper (Made by Kodak Co Ltd. gray cards R-27. white face, reflective ratio,90%)

3) Distance measuring range of the optical sensor system

Fig.1 Internal Block Diagram

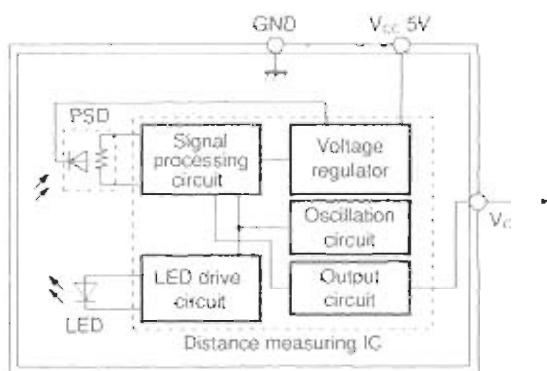


Fig.2 Timing Chart

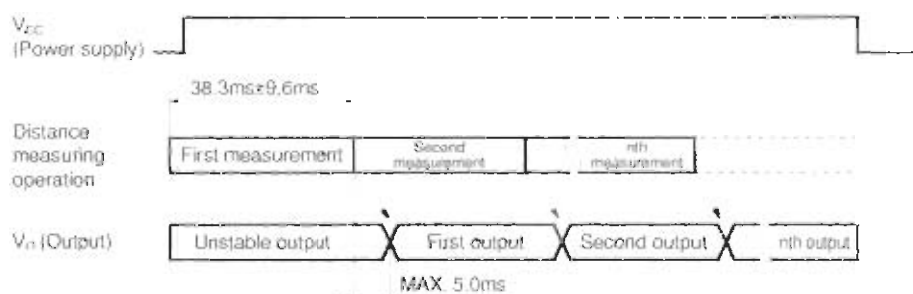
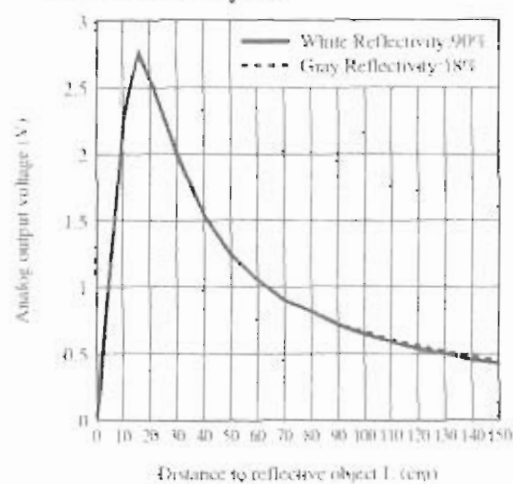


Fig.3 Analog Output Voltage vs. Distance to Reflective Object



NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.
- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.
- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:
 - (i) The devices in this publication are designed for use in general electronic equipment designs such as:
 - Personal computers
 - Office automation equipment
 - Telecommunication equipment [terminal]
 - Test and measurement equipment
 - Industrial control
 - Audio visual equipment
 - Consumer electronics
 - (ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
 - Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
 - Traffic signals
 - Gas leakage sensor breakers
 - Alarm equipment
 - Various safety devices, etc.
 - (iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
 - Space applications
 - Telecommunication equipment [trunk lines]
 - Nuclear power control equipment
 - Medical and other life support equipment (e.g., scuba).
- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.
- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.
- This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.
- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

BIBLIOGRAFÍA

[1] POLOLU 3PI, Página principal donde se describe al pololu 3pi: partes, funciones, precios y guías que ayudaran a programar el robot pololu 3pi.

<http://www.pololu.com>

Fecha de consulta: 25/04/11.

[2] POLOLU 3PI, Guía de usuario del robot pololu el cual da pautas para verificar el pololu así como el de instalar software y drives que se usen para la programación del robot.

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

Fecha de consulta: 25/04/11.

[3] Descripción y especificaciones breves del módulo Orangután,
<http://www.pololu.com/catalog/product/1227/specs>

Fecha de consulta: 25/04/11.

[4] Recursos dados para la utilización de las librerías, programadores para el pololu 3 pi.

<http://www.pololu.com/catalog/product/1227/resources>

Fecha de consulta: 27/04/11.

[5] Pardue Joe, Smiley Micros.com, *C programming for Microcontrollers, Featuring ATMEL's AVR Butterfly and the Free WinAVR Compiler*, edición 2005.

<http://www.smileymicros.com/>

Fecha de consulta: 27/04/11.

[6] Hoja de especificaciones del microcontrolador ATMEGA328P.

http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf

Fecha de consulta: 02/05/11.

[7] POLOLU 3PI, Detector de obstáculos y graficación de la posición de los obstáculos,

http://www.roso-control.com/Espanol/EDU/MICRO/70_Proyectos/24_XBot/Paper.pdf

Fecha de consulta: 27/04/11.