ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

ROBOT POLOLU PROGRAMADO PARA SEGUIR REFERENCIA MOVIL Y OBEDECER COMANDOS INALAMBRICOS

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Mirna Viviana Baque Gutiérrez

Lenin Stalyn Gomez Romero

GUAYAQUIL – ECUADOR

AÑO 2011

Agradecimiento

Agradezco a Dios por las bendiciones otorgadas.

A mi familia por su apoyo incondicional.

A los profesores por ser una excelente guía, en especial al Ing. Carlos Valdivieso, y a cada una de las personas que hicieron posible la realización de esta tesina.

Viviana Baque Gutiérrez

Dedicatoria

MIS PADRES

A MIS HERMANOS

A MIS SOBRINOS

Viviana Baque Gutiérrez

Agradecimiento

A Dios quien me dio fe y fortaleza para terminar este trabajo.

A mis padres Cesar y Rosa, y mi esposa Gabriela, de quienes siempre tuve su apoyo constante e incondicional.

A mis amigos quienes me dijeron que sea constante y que lograría concluir mi proyecto de grado.

Stalyn Gómez Romero

Dedicatoria

A mis padres quienes siempre creyeron en mí y estuvieron en cada momento de estos arduos años de estudio

A mis hermanos, mi cuñada, mi esposa Gabriela León y mi hijo Lenin Gabriel quienes han sido mi motivo de inspiración para terminar este proyecto.

A mis verdaderos amigos de la universidad que fueron de grata compañía por su apoyo incondicional, y aquellas personas que no creyeron en lo que hacía porque me dieron más fortaleza para terminar

Stalyn Gómez Romero

TRIBUNAL DE SUSTENTACIÓN

ING. CARLOS VALDIVIESO A.

PROF. DEL SEMINARIO DE GRADUACIÓN

ING. HUGO VILLAVICENCIO V.

DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral".

(Reglamento de Graduación de la ESPOL)

Stalyn Gómez Romero

Viviana Baque Gutiérrez

RESUMEN

El proyecto consiste programar robot pololu para seguir referencia móvil mediante comandos inalámbricos, usando comunicación inalámbrica de Radio frecuencia.

El objetivo de este proyecto es controlar el robot pololu que sigue a un móvil. Al presionar el Joystick del KIT AVR BUTTERFLY se transmiten los comandos que son avanzar, retroceder, girar a la derecha, girar a ala izquierda y parar. Estos comandos se receptan al controlador del Robot pololu que procesa las instrucciones generando el movimiento de los motores, que son los que permiten que el robot se mueva.

El Robot Pololu y el AVR Butterfly traen incorporado una pantalla LCD que permite visualizar las instrucciones que se están enviando y recibiendo.

El desarrollo de este proyecto fue realizado en lenguaje C haciendo uso de las herramientas del del AVR Studio 4 que logra programar los micro controladores ATmega 328 y ATmega 169 que son el corazón del robot pololu y el AVR Butterfly

ÍNDICE GENERAL

Contenido **pag**

RESUMEN VIII

INDICE GENERAL..... IX

INDICE DE FIGURAS..... XII

INTRODUCCIÓN..... XIV

CAPÍTULO 1

1. DESCRIPCIÓN DEL PROYECTO..... 1

1.1 Antecedentes..... 1

1.2 Descripción del proyecto..... 2

1.3 Aplicaciones..... 3

1.4 Proyectos similares..... 3

1.4.1 Robot pololu seguidor de pared..... 3

1.4.2 Control de robot TRI TRAC radio controlado mediante joystick de Play Station 5

1.4.3 Robot controlado por mando de wii con cámara inalámbrica..... 6

1.4.4 Kit avanzado Robot SR1 S300015..... 7

CAPÍTULO 2

2. Fundamentos Teóricos 10

2.1 Descripción básica del software..... 10

2.1.1 AVR studio 4..... 12

2.1.2 Proteus 7.7..... 14

2.2 Descripción básica del hardware..... 14

2.2.1 Robot Pololu 3pi 15

2.2.2 Información general..... 16

2.2.3 Atmega 328 17

2.3 AVR Butterfly 18

2.3.1 Elementos que contiene el AVR Butterfly 20

2.3. 2 Joystic 21

2.3.4 Pantalla LCD 22

2.3.5 Conexión de la pantalla LCD STK502 al ATmega169 21

2.3.6 Atmaga 169 23

2.4 Modulo Hm_Tr 25

2.4.1 características 25

2.4.2 Aplicación..... 26

CAPÍTULO 3

3 Diseño del Proyecto..... 27

3.1 Prueba inicial..... 27

3.2 Descripción del proyecto final..... 28

3.2.1 Diagrama de bloques..... 29

3.2.2 Diagrama de flujo del transmisor..... 30

3.2.3 Código AVR Butterfly para el Transmisor..... 31

3.2.4 Diagrama de flujo del Receptor..... 47

3.2.5 Código del receptor..... 49

CAPÍTULO 4

4.1 Simulación y pruebas del proyecto 56

4.2 Pruebas de Funcionamiento..... 56

4.3 Pruebas de simulación del transmisor y Receptor..... 56

4.4 Resultados de la simulación..... 59

ÍNDICE DE FIGURAS

Tabla de contenido pag

Figura 1.4.1 Robot Pololu 3pi..... 3

FIgura 1.4.2 Control de robot TRI TRAC radio controlado mediante joystick de Play Station)2 5

Figura 1.4.3 Robot controlado por mando de wii con cámara inalámbrica..... 6

Figura 1.4.4 Kit avananzado robot SR1 S300015..... 9

Figura 2.1a Página de inicio del programa AVR STUDIO 4..... 11

Figura 2.1 b Programa Proteus..... 11

Figura 2.1.1 selección del lenguaje..... 13

Figura 2.1.2 ventana de inicio de Proteus..... 14

Figura 2.2.1 a Robot Pololu 3pi..... 15

Figura 2.2.1 b caracteristicas del Pololu..... 17

Figura 2.3AVR Butterfly 19

Figura 2.3.2 Diagrama del Joystick 21

Figura 2.3.3 pantalla LCD 22

Figura modulo Hm_Tr 25

Figura 3.2.1 Diagrama de Bloques del proyecto..... 29

Figura 3.2.2 Diagrama de Transmisor..... 30

Figura3.7 Diagrama de Flujo Receptor 49

Figura 4.3 a diagrama avance de pololu 56

Figura 4.3 b diagrama pololu retrocede..... 57

Figura 4.3 c simulación del receptor..... 57

Fig. 4.2.1: Motores ejecutando desplazamiento 58

Figura 4.3 e diagrama del pololu funcionando 59

INTRODUCCION

El objetivo del proyecto es implementar el control del robot Pololu 3π por medio del KIT AVR BUTTERFLY mediante comunicación inalámbrica RF, el robot pololu recibirá los comandos que serán enviados al presionar el joystick del AVR Butterfly para procesar las instrucciones que fueron recibidas y así ejecutar el movimiento de los motores del pololu y lograr su desplazamiento para que siga al móvil.

Los capítulos se encuentran constituidos de la siguiente manera:

En el primer capítulo, se refiere una descripción general del proyecto, las partes y funciones del mismo, aplicaciones en la industria y proyectos similares como el Robot TRIC-TRAC controlado mediante joystick de play station el cual es un robot todo terreno con orugas que puede ser comandado por radio control o de forma autónoma.

En el segundo capítulo, se detallan las herramientas del hardware como son el robot POLOLU y EL AVR BUTTERFLY así como la parte del software como es el AVR STUDIO 4, sus compiladores, el simulador PROTEUS y las librerías para el correcto funcionamiento del robot pololu.

El tercer capítulo contiene información sobre el diseño e implementación del proyecto, diagrama de bloque general del sistema, diagramas de flujos del transmisor y del receptor así como los algoritmos del programa del microcontrolador.

En el cuarto y último capítulo se muestran las pruebas de simulación, pruebas de funcionamiento realizadas con el robot pololu y el AVR butterfly. Los análisis de resultados, validaciones pruebas.

CAPÍTULO 1

1 Descripción del Proyecto.

1.1 Antecedentes

El avance tecnológico cada día es más vertiginoso en toda área de la electrónica, y somos testigos de que día tras día surgen nuevos dispositivos electrónicos que son más veloces, más pequeños y con un menor costo. Las características de estos nuevos y modernos dispositivos tecnológicos son diseñados más compactos y eficientes, dando lugar a una buena aplicación, Que sería de mucha ayuda para las personas más aun en el área de la producción.

La principal aplicación de los robots tiene lugar en la industria, donde se necesitan repetir tareas, seguir cierto recorrido donde las personas no pueden llegar y estos robots llegarían a realizar ese trabajo con el solo hecho ser controlado mediantes comandos inalámbricos, por internet, etc. Un robot está programado para realizar los mismos movimientos con una buena precisión, por lo que es perfecto para aplicaciones industriales.

Aunque todos los campos de la industria son susceptibles de emplear diferentes tipos de robots para determinadas operaciones. El robot pololu 3pi es un dispositivo con características que la tecnología de este momento demanda. El Pololu 3pi es un pequeño robot autónomo de alto rendimiento, diseñado para sobresalir en competencias en línea recta o para resolver laberintos.

1.2 Descripción del proyecto

Este proyecto tiene como función principal programar el robot pololu 3pi que seguirá una referencia móvil que será controlada mediante comandos inalámbricos, para esto se necesita un circuito receptor que recibirá los comandos que serán enviados a través del Butterfly que es el que hará las veces de transmisor.

EL sistema de control del proyecto será diseñado para que funcionen con los micro controladores ATMEGA 328 y el ATMEGA 169 usando la interfaz del AVR Studio 4 que es un software que ofrece dos tipos de compiladores que son: ATMEL AVR assembler para lenguaje assembler y AVR GCC para lenguaje C. Este sistema se realizará con un programa adecuado de control que se describirá en el presente trabajo.

El Robot Pololu y AVR Butterfly traen integrado un display que permite mostrar los mensajes que indican lo que están realizando en el momento, es

decir en el caso de butterfly indica que comandos se está enviando al momento de presionar el joystick y el Pololu indica que comandos se están recibiendo para la ejecución. Ya sean avanzar, retroceder girar a la derecha, girar a la izquierda o parar al pololu.

1.3 Aplicaciones

Una de las principales aplicaciones de este proyecto es la de explorar lugares de difícil accesos sin importar los obstáculos ya que este los evita aunque se le podría incorporar una cámara para poder observar el camino recorrido.

1.4 Proyectos similares

1.4.1 Pololu 3pi seguidor de pared.

Figura 1.4.1 de Robot Pololu 3pi

El robot 3pi es completa plataforma móvil diseñada para sobresalir en la línea siguiente y concursos de resolución de laberintos.

Este proyecto demuestra una posible configuración donde dos sensores de distancia Sharp analógicos están conectados con un mínimo de soldadura. Muchos otros sensores se podrían utilizar para distanciar el pololu 3pi: un receptor de radio-control podría hacer un giro con de precisión, un telémetro sonar podría mejorar el alcance de la de detección del pololu 3pi, y un sensor de distancia digital podría ofrecer una detección rápida obstáculo frente. Pero en este caso se trabaja con los sensores Sharp

Los dos sensores de distancia permiten explorar una habitación siguiendo la pared y evitando los obstáculos que encuentra en el recorrido. Este trata de mantener una distancia prudencial de la pared ni ningún otro obstáculo que ve en su lado izquierdo y se convierte en el lugar a la derecha si es que llega a detectar un obstáculo en frente de él girara y seguirá por otro recorrido. Como medida de seguridad adicional, cada 15 segundos, el robot trata de una copia de seguridad en caso de que sin saberlo ha atascado mientras explora. La figura anterior muestra el pololu 3pi seguidor de pared (wall follower)

1.4.2 Control de robot TRI TRAC radio controlado mediante joystick de Play Station 2

TRI-TRAC es un robot todo terreno con orugas que puede ser usado mediante radio control o mediante control autónomo. Debido a su oruga el robot puede trabajar tanto en interior como exterior. El kit incluye:

- Parte mecánica con motores y oruga.
- Circuito Next Step con procesador ATCOM.
- Circuito de control de motores.
- Mando a distancia.

Figura 1.4.2 control de robot Tric trac

1.4.3 Robot controlado por mando de wii con cámara inalámbrica

Este proyecto está basado en Arduino. Posee una cámara inalámbrica con alcance de 30 metros montada sobre un sistema Pan&Tilt que es usado en mini cámaras en el que va montado un sonar de ultrasonidos y un láser en cruz. Su diseño permite girar hasta 360º para tener una visión periférica completa.

Ahora mismo, el control se realiza con un mando nunchuk de wii, El Nunchuk es una expansión para el mando inalámbrico de Wii. Con el acelerómetro para el movimiento de las orugas, el joystick para el control del Pan&Tilt y los botones para seleccionar entre modos y activar el láser.

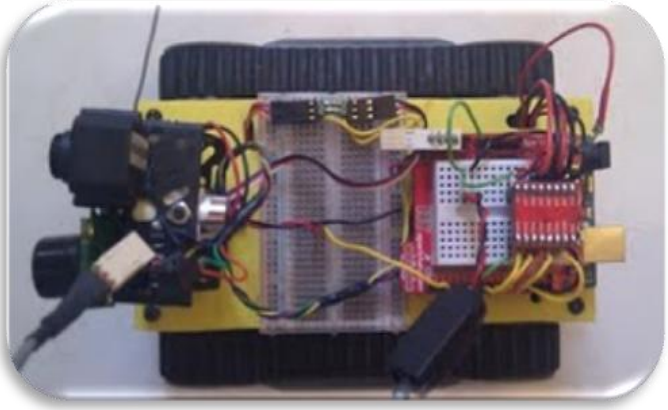


Figura 1.4.3 robot controlado por comando wii

1.4.4 KIT AVANZADO ROBOT SR1 S300015

El kit avanzado del robot SR1 es una versión más completa que incluye todos los componentes de la versión básica, más un sensor brújula digital, un transceptor de datos vía radio, un sensor de líneas, un mando a distancia por infrarrojos y un placa de experimentación y prototipo. Con estos componentes adicionales ya se pueden realizar labores y programas más avanzados incluyendo el control remoto vía radio y la transmisión de datos

Inalámbrica entre el PC y el robot o entre varios robots. Con la ayuda de la brújula se pueden realizar programas de navegación basados en grados reales, mientras que el sensor de líneas permite que el robot siga una línea dibujada en el suelo. Con todos los accesorios extras, que ahora se incluyen, se pueden ejecutar los programas avanzados del disco y que son explicados en detalle en el manual. Al igual que la versión básica, el Robot SR1 viene acompañado por un completo juego de programas y ejemplos que le enseñan poco a poco a programar y sacar partido al SR1.

El SR1 es un robot multifuncional de desarrollo y aprendizaje dirigido a aquellos entusiastas y aficionados a la robótica que quieren aprender y profundizar en la construcción real de robots móviles de experimentación. Tanto si es un recién llegado al mundo de la robótica, como si es un

aficionado experto, encontrará que el robot SR1 es la plataforma idónea donde hacer todo tipos de proyectos desde un simple guiado por colisión, hasta un avanzado robot radio controlado con sistema de telemetría y capaz de enviar audio, vídeo y datos de forma inalámbrica de la misma forma que lo haría un robot como los que se mandan a explorar el espacio.

El robot SR1 cuenta con un chasis lo suficientemente robusto para proteger todos los componentes mecánicos y electrónicos del robots mientras se desplaza en cualquier entorno interior. El chasis admite ampliaciones como plataformas de carga, techos con sensores, motores dc, ruedas de sumo, etc. Desde el punto de vista de la electrónica, se ha buscado un compromiso entre versatilidad de funciones y facilidad de programación que le permita.

Disponer de gran cantidad de sensores, además de poder incluir accesorios extras como cámaras, servos, etc. y todo ello controlable y programable desde cualquier PC sin necesidad de otro software que el proporcionado.

El robot SR1 está diseñado para desenvolverse de forma autónoma y segura en cualquier parte como el hogar, escuela u oficina siendo capaz de eludir obstáculos y trampas.

El correcto funcionamiento para que este robot se mueva es gracias a la gran cantidad de sensores que este tiene como son:

Sensores de contacto

Sensores de luz

Sensores de inclinación

Sensores de distancia

Sensores seguidores de línea

Figura 1.4.4 Robot SR1

CAPÍTULO 2

2 Fundamentos Teóricos.

En este capítulo se muestra la información de los dispositivos y componentes utilizados para la implementación del proyecto empezando con su parte más esencial el pololu 3pi y el avr butterfly, asi como el software para la programación y simulación de los mismos como es el AVR studio4 y el Proteus.

2.1 Descripción básica del software

Para realizar la aplicación que gestionará el pololu y el butterfly con los datos que serán enviados y recibidos a través de ellos usaremos el programa AVR studio 4 usando el lenguaje más acorde ya sea lenguaje ensamblador o c. Fig 2.1 a Estos compiladores son los que nos ayudaran a entender el código base del robot Pololu 3pi y permitirá implementar una variedad al mismo.

El objetivo del programa es que se pueda realizar ciertas tareas como las siguientes:

- Opciones para poder configurar los comandos que serán enviados por medio del butterfly, usando el joystick.
- Opciones para poder seleccionar los comandos de comunicación que sean necesarios para seguir el móvil.

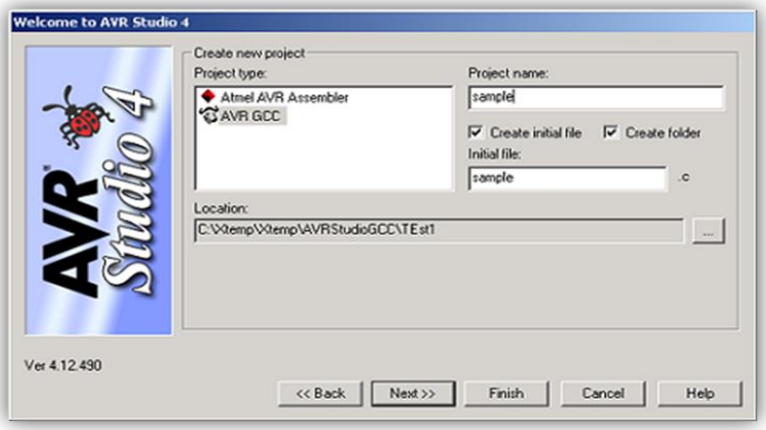


Figura 2.1a Página de inicio del programa AVR STUDIO 4

Para poder realizar las conexiones y ver la simulación del proyecto es necesario trabajar con el software de simulación proteus 7.7 según la Fig 2.1 b que muestra la ventana de inicio del programa.

Figura 2.1 b Programa Proteus 7.7

2.1.1 AVR studio 4.

Es un entorno de desarrollo IDE ensamblador y programador de software para el desarrollo de aplicaciones de Atmel AVR de 8 bits en Windown NT, Windows 2000, Windows XP, Windows Vista y Windows 7.

El IDE soporta todas las herramientas de Atmel que apoyan a la arquitectura AVR 8 bit.

AVR studio incorpora un depurador que permite el control de ejecución con fuente y nivel de instrucción, paso a paso y puntos de interrupción, el registro, la memoria y E/S puntos y configuración y gestión, y apoyo a la programación completa para los programadores independientes además permite crear archivos assembler (asm) y archivos .C véase la Fig 2.1.3.

Características Principales

Integrado ensamblador y simulador

- Se integra con el compilador GCC plug-in
- AVR RTOS plug-in de apoyo
- Soporta AT90PWM1 y ATtiny 40.
- Herramientas de CLI al día con el apoyo de TPI
- Ayuda en línea.

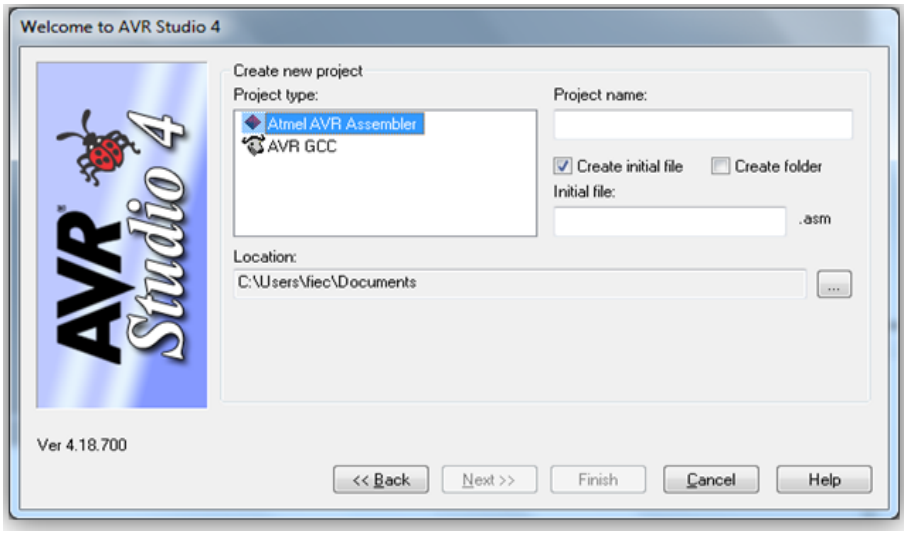


Figura 2.1.1 selección del lenguaje

AVR Studio cuenta con algunas formas para poder programar los microcontroladores de la familia ATMEL, para la realización de este trabajo se utilizaran los siguientes.

La programación ISP, a la cual se accede mediante la opción AVRISP, permite grabar el microcontrolador tanto del Robot Pololu 3pi asi también como del AVR Butterfly. Se hace uso del Pololu USB AVRProgrammer el cual se conecta al puerto ISP de los módulos a través de un cable de 6 líneas.

Se conecta el modulo a programar al PC se verifica la conexión luego comienza el proceso de grabación y luego de verificación del micro controlador.

2.1.2 Proteus 7.7

Es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción.

Sus reconocidas prestaciones lo han convertido en el más popular simulador software para micro controladores PIC. Fig 2.1.4

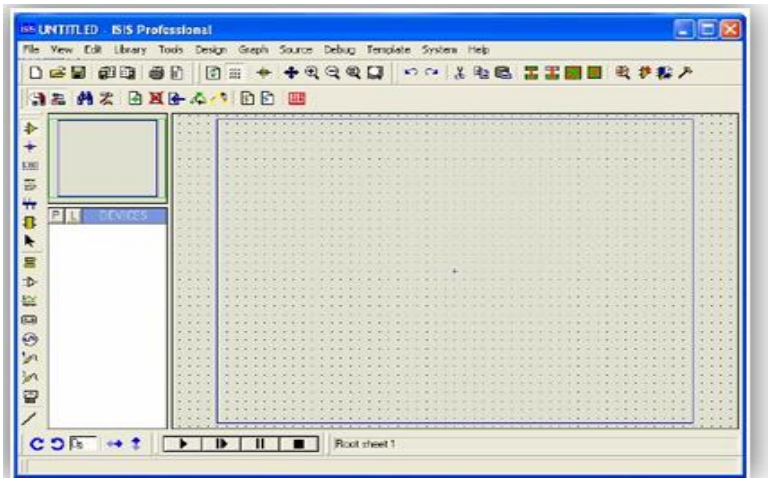


Figura 2.1.2 ventana de inicio de Proteus

2.2 Descripción básica del Hardware

En lo que hace referencia a la parte física, el hardware es la implementación del robot pololu y el butterfly para que siga referencia móvil. En este capítulo se muestra información de los componentes utilizados para la implementación del proyecto.

El pololu y el butterfly son los principales dispositivos pero dentro de estos se encuentran algunos integrados, los microcontroladores que se programaran son el ATMEGA168 o el ATMEGA328 ambos de la serie AVR.

2.2.1Robot Pololu 3pi

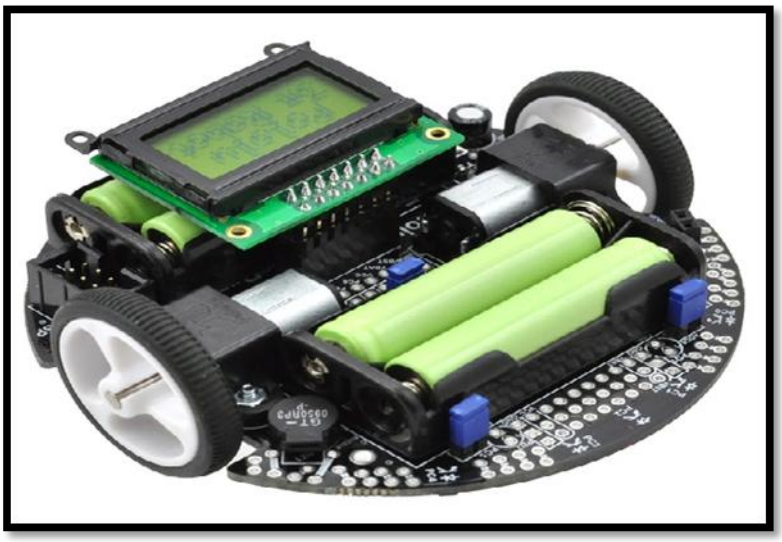


Figura 2.2.1 a Robot pololu 3pi

El robot Pololu 3pi es una plataforma completa, de alto rendimiento móvil con dos motores con engranajes de metal micro, cinco sensores de reflectancia, un carácter de 8 x 2 LCD, un timbre, y tres botones de usuario, todos conectados a un micro controlador ATMega328 C-programable. Capaz de alcanzar velocidades superiores a 3 pies por segundo. Fig 2.2.1

2.2.2 Información general

El robot 3pi está diseñado para sobresalir en línea y concursos de resolución de laberintos. Tiene un tamaño pequeño (9,5 cm/3.7” de diámetro, 83 g72.9 oz sin baterías) y tienes 4 pilas AAA, mientras que un sistema de poder único ejecuta los motores a una velocidad constante independiente de 9,25 V de la carga del nivel de batería. La tensión regulada del 3pi permite alcanzar velocidades de hasta 100 cm /segundo, mientras que lo precisa en vueltas y giros que no varían con el voltaje de la batería.

El robot pololu 3pi es una gran plataforma para personas con experiencia en programación en lenguaje C. Su corazón es un microcontrolador ATmel ATmega 328P funcionando a 20 MHz y con 32 KB de memoria Flash de programa, 2 KB de RAM, y 1KB de memoria persistente. El GNU C/C ++ funciona a la perfección con la 3pi, Atmel AVR studio proporciona un entorno de desarrollo cómodo, y un amplio conjunto de bibliotecas proporcionadas por pololu le hace una brisa para interactuar con todo el hardware integrado. El 3pi también es compatible con la plataforma de desarrollo Arduino polular.

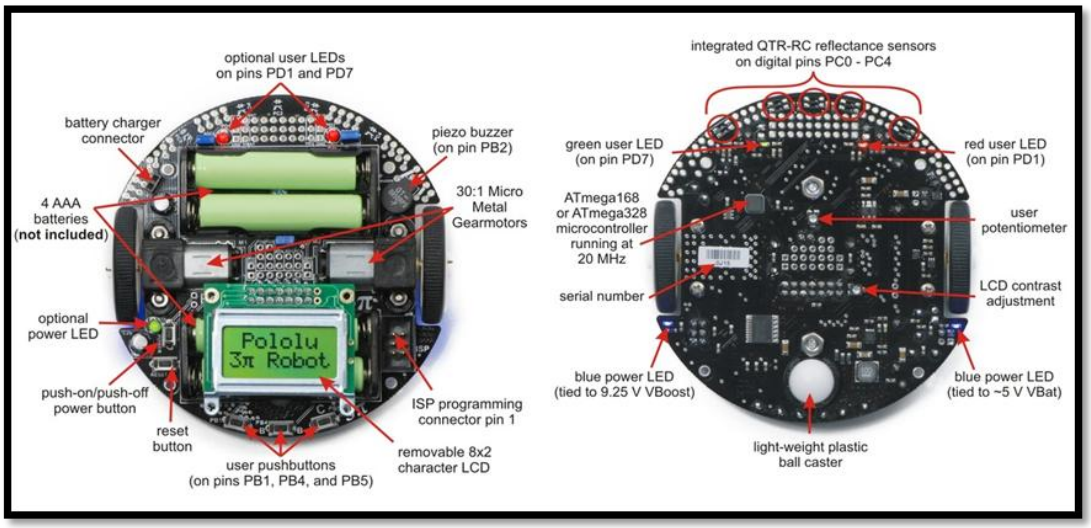


Figura 2.2.1 b características del Pololu

2.2.3 Atmega 328

El ATmega328P es un micro controlador de baja potencia CMOS de 8 bits basado en el AVR mejorado la arquitectura RISC. Mediante la ejecución de instrucciones de gran alcance en un solo ciclo de reloj, el ATmega 328P logra tasas de transferencia cerca de un MIPS por MHz que permite al diseñador del sistema a optimizar el consumo de energía en comparación con la velocidad de procesamiento.

Las características del ATMEGA 328 son las siguientes:

- 2KB de memoria flash ISP con la lectura y escritura
- Memoria eeprom 1KB
- 2KB SRAM

- 23 registros de propósito general
- Líneas de entradas /salidas
- 32 registro de propósito general de trabajo
- Tres temporizadores flexibles contadores con comparadores
- Interrupciones internas y externas
- 6 canales de 10 bits
- Convertidor A/D
- 5 modos seleccionables de software de ahorro de energía.

Mediante la ejecución de instrucciones de gran alcance en un solo clic de reloj, el dispositivo logra tasas de transferencia de cerca de 1 MIPS por MHZ, equilibrando el consumo de energía y velocidad de procesamiento.

2.3 AVR Butterfly

Los kits de AVR Butterfly están diseñados para demostrar los beneficios y las principales características de los microcontroladores AVR. AVR Butterfly es un módulo de soporte que puede ser utilizado en numerosas aplicaciones. Fig 2.3

El AVR Butterfly contiene un microcontrolador ATMega169, el cual va a realizar el comando de las diferentes funciones de las que es capaz éste kit.

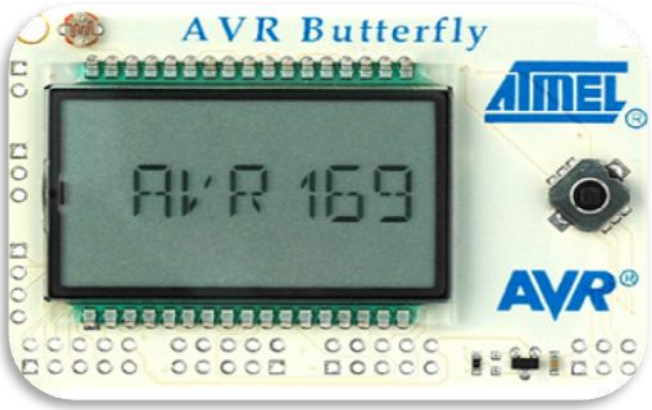


Figura 2.3AVR Butterfly

Caracteristica del Butterfly

- Diseño de bajo poder
- El tipo de paquete MLF
- Controlador de LCD
- Memorias
- Flash, EEPROM, SRAM, DATAFLASH externos
- Interfaces de comunicación
- UART, SPI, USI
- Convertidor analógico a digital (ADC)
- Temporizadores / contadores

- Reloj en tiempo real (RTC)
- Modulación por impulsos (PWM)

2.3.1 Elementos que contiene el AVR butterfy.

Los siguientes recursos están disponibles en el kit del butterfly Atmega 169

- LCD en la pantalla de vidrio con 120 segmentos, para demostrar la ATMEGA 169 controlador LCD.
- Joystick de 4 direcciones con empuje el centro, como la entrada del usuario
- Elemento piezoeléctrico, para reproducir sonidos
- 32KHZ Xtal para la RTC
- 4 Mbit DATAFLASH, para el almacenamiento de datos
- RS-232-convertidor de nivel, para comunicarse con las unidades fuera de borda
- Coeficiente de temperatura negativo (NTC) termistor, para medir la temperatura
- Resistencia depende de la luz (LDR) para medir la intensidad de la luz 3v pila de botón (600mAh) para proporcionar energía de funcionamiento
- Emulación JTAG, para la interfaz de comunicación adicional

El ATMEGA 169 en el juego de controles de los periféricos externos, y también se puede utilizar para hacer la lectura de voltaje de 0 a 5 voltios.

El kit se puede reprogramar una serie de maneras diferentes, incluyendo programación en serie a través del puerto JTAG. La mayoría de usuarios prefieren utilizar el gestor de arranque precargado con el estudio de AVR para descargar nuevo código.

El AVR Butterfly viene con una aplicación reprogramada. En esta sección se pasará a través de los fundamentos de esta solicitud.

2.3.2 Joystick

El AVR Butterbvly tiene un joystick en miniatura para operar la entrada de usuario. Maneja en cinco direcciones, incluyendo push arriba, abajo derecha izquierda y centro. La línea común de todas las direcciones es GND. Esto significa que pull-p interna debe estar habilitado en el ATMEGA 169 a detectar a partir de la entrada de la palanca de mando, véase la figura 2.2.6

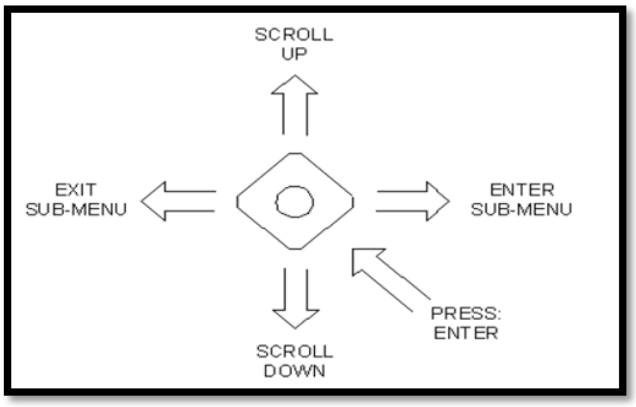


Figura Diagrama del Joystick 2.3.2

2.3.3 Pantalla LCD

La pantalla LCD del AVR Butterfly es la misma que la utilizada en la disposición STK502 de Atmel. Las conexiones entre el ATmega 169 y la pantalla LCD también son las mismas.

STK502 es un módulo superior diseñado para añadir soporte ATmega169 a la placa de desarrollo STK500 de Atmel Corporation.

STK502 incluye una pantalla LCD. Cuenta con seis dígitos de 14 segmentos, y algunos segmentos adicionales. En general, la pantalla es compatible con 120 segmentos. La pantalla está diseñada para la tensión de funcionamiento de 3V. Fig 2.2.7

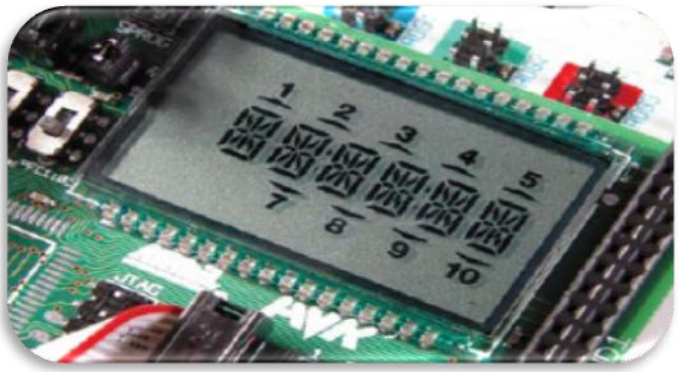


Figura 2.3.4 pantalla LCD

2.3.4 Conexión de la pantalla LCD STK502 al ATmega169

El segmento de pines de ATmega 129 se encuentran en PORTA, PORTC, PORTD y PORTG. Por razones de simplicidad en el uso de todos ellos son

unidos en la cabecera de la etiqueta “pasadores de segmento de ATmega 169”. La cabecera a su lado, la etiqueta “STK502 pines LCD” tiene todos los segmentos pines para la pantalla LCD en el STK502.

Al utilizar el cable de 34 derivaciones que viene con el STK502-kit, los dos pines de conexión se pueden conectar, permitiendo que el ATmega 169 para controlar la pantalla LCD.

2.3.5 Atmega 169

El ATmega 169 es un microcontrolador de baja potencia CMOS de 8 bits basado en el AVR mejorado la arquitectura RISC. Mediante la ejecución de instrucciones de gran alcance en un solo ciclo de reloj, el ATmega 169 logra tasas de transferencia cerca de 1 MIPS por MHz que permite al diseñador del sistema optimizar el consumo de energía en comparación con la velocidad de procesamiento.

El nucleo AVR combina un amplio conjunto de instrucciones con 32 registros de propósito general de trabajo.

Todos los 32 registros están conectados directamente a la unidad lógica aritmética (ALU), lo que permite dos registros independientes que se alcanzará en una sola instrucción ejecutada en un ciclo de reloj. La arquitectura resultante es un código más eficiente mientras que alcanza

rendimientos de hasta 10 veces más rápido que los convencionales microncontroladores CISC.

El ATmega 169 proporciona las siguientes características:

- 16k bytes de sistema programable
- Flash con lectura y escritura mientras que las capacidades, 512 bytes de EEPROM, SRAM bytes 1K.
- 54 registros de propósito general
- 32 registros de propósito general de trabajo
- Controlador de LCD con la resistencia de step-up de tensión
- Una serie UART programable, serie universal
- Sistema de interrupción
- Interfaz con el inicio de condición del detector
- El Powerdown modo guarda el contenido del registro

2.4 MODULO HM –TR

Modulo transparente de datos inalámbricos de enlace que se desarrolla por la microelectrónica, dedicada a las aplicaciones que necesita la transmisión de datos inalámbrica.

Cuenta con alta velocidad de datos, ya la distancia de transmisión. El protocolo de comunicación es auto controlado y completamente transparente para la interfaz de usuario. El módulo puede ser incorporado a su diseño actual, de modo que la comunicación inalámbrica se pueden configurar fácilmente.

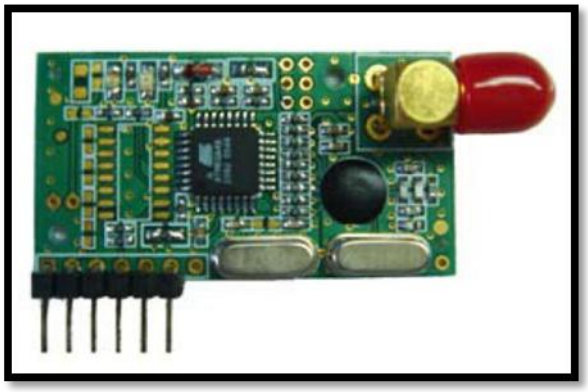


Figura 2.4 modulo Hm_Tr

2.4.1 Características

1. FSK tecnología, el modo dúplex medio, robusto a las interferencias

2. Banda ISM, sin necesidad de solicitar licencia de uso de frecuencias

3. Frecuencia de operación puede ser configurado y puede ser utilizado en aplicaciones FDMA

4. Desviación de frecuencia de transmisión y ancho de banda del receptor puede ser seleccionado.

5. Traducción Protocolo es dueño de sí mismo, fácil de usar.

6. Velocidad de datos se puede seleccionar de una amplia gama.

7. Proporcionar pines permiten controlar ciclo de trabajo para satisfacer los requisitos de aplicación diferentes

8. Alta sensibilidad, rango de transmisión de largo.

9. UART interfaz estándar, TTL o RS-232 seleccionable por el nivel de la lógica

10. Muy, confiables de pequeño tamaño, fácil montaje.

2.4.2 Aplicación

- Control remoto, sistema de medición a distancia
- Inalámbrico de medición
- Control de acceso
- Identidad de la discriminación
- Recopilación de datos

CAPÍTULO 3

3 Diseño Del Proyecto

En este capítulo se ponen de manifiesto las etapas de diseño, implementación para la elaboración de este proyecto, sus diagramas de bloques, algoritmos y los códigos que serán cargados en los micro controladores a usarse.

Para comenzar a trabajar con el Robot Pololu y el AVR Butterfly se realizaron algunas pruebas antes de la programación de nuestro proyecto.

3.1 Prueba Inicial

Para la realización de este proyecto se seleccionó el Robot Pololu 3pi que trae incorporado un microcontrolador ATMEGA 328 que es el que se programa para que ejecute el programa a realizarse.

Como primer pasó antes de comenzar a trabajar con el pololu se realizo las pruebas demo que vienen incorporadas a este robot, como es el seguir una línea usando las librerías respectivas, de igual manera se realizó con el AVR butterfly, y esto nos permitió familiarizarnos con el programa AVR.

Al cargar el programa demo en el Pololu este hace que el robot muestre en la pantalla LCD que este posee, el nombre de tal robot como es Pololu 3 π Robot. Además se puede escuchar una música que es atraves del buzzer del robot.

Se pudo realizar con el butterfly las pruebas como es prender la pantalla LCD y ver los comandos que tiene el Joystick.

3.2 Descripción del proyecto final

3.2.1 Diagrama de bloques

El diagrama de bloques del sistema se encuentra establecido por la etapa del transmisor y la etapa del receptor. Ver fig 3.4.1

La etapa de transmisión está constituida por el Kit AVR Butterfly que envía los datos correspondientes mediante el Joystick que es por el cual se seleccionan los comandos .La señal ingresa al módulo de transmisión de RF, que se encarga de modular la señal para ser transmitida.

En la etapa de recepción se encuentra el módulo receptor que demodula la señal para ser receptadas por medio del Robot Pololu que luego comenzara a moverse dependiendo de los comandos que sean presionados por el Joystick.

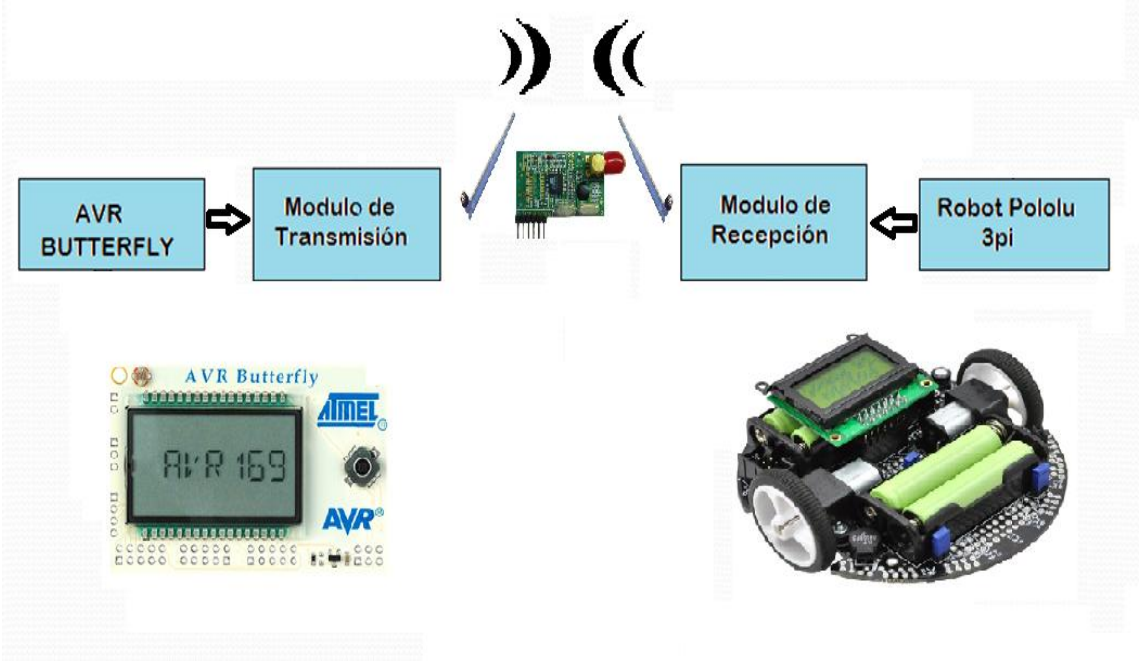


Fig 3.2.1 Diagrama de Bloques del proyecto

3.2.2 Diagrama de Flujo del Transmisor

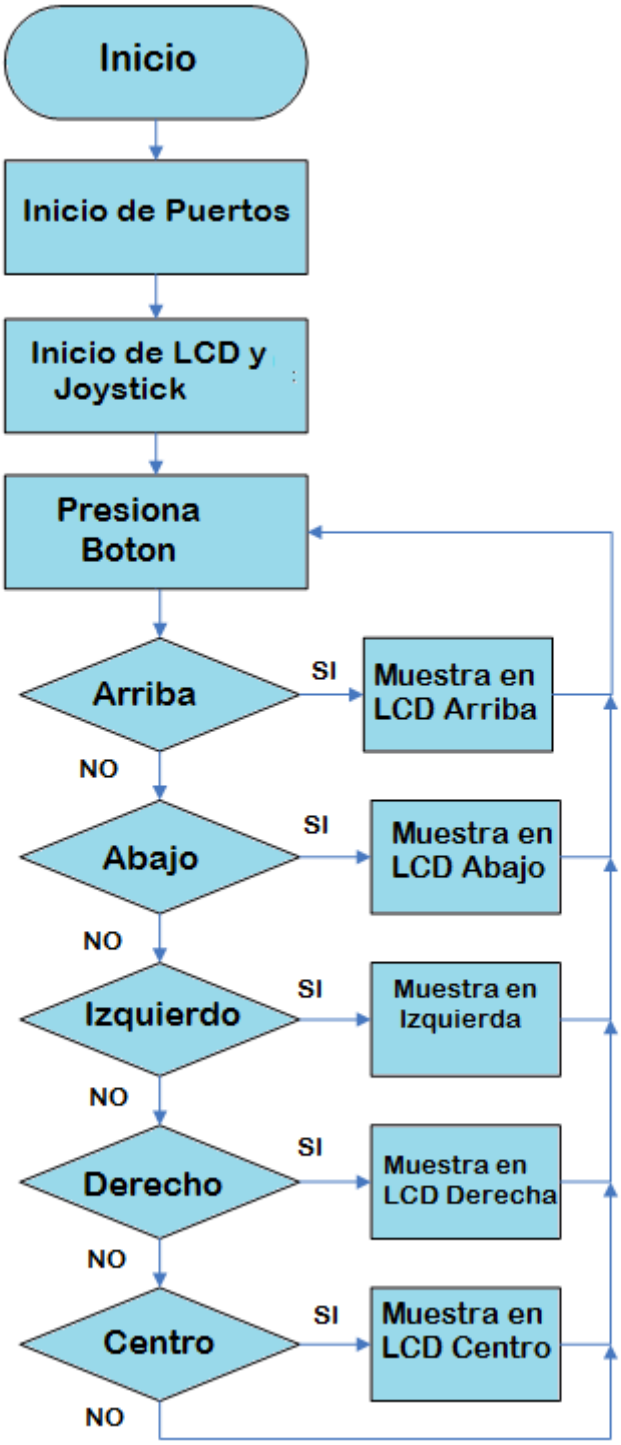


Figura 3.2.2 Diagrama de Transmisor

Con este algoritmo se muestra lo que realiza el transmisor primero se inicializa, luego se setean los puertos después de esto se da inicio de la LCD y del joystick, el programa espera a que presionen botón, si este se presiona, entonces se envía trama esta trama contiene un carácter que da una instrucción para que el robot pololu lo recepte.

Estos caracteres son los siguientes

u arriba entonces enciende motores del pololu hacia adelante

d abajo entonces enciende motores del pololu hacia atrás

R derecha , enciende los motores del pololu a la derecha

L izquierda, enciende los motores hacia la izquierda

S para, pone los motores en stop

3.2.3 Código AVR Butterfly para el Transmisor

```
//Declaración de constantes
#define  Centro    0
#define  Arriba    1
#define  Abajo     2
#define  Izquierda 3
#define  Derecha   4
#define  Otros     5

//Librerias a usar
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
```

```
#include <avr/delay.h>
#include <inttypes.h>

//Librerias a usar que no pertenecen al entorno
#include "mydefs.h"
#include "LCD_functions.h"
#include "LCD_driver.h"
#include "button.h"
#include "usart.h"


//Prototipo de funciones
int Obtener_Boton(void);
//Programa principal
int main(void)
{
                    //Declaración de variable
          int input;


//Se habilita las interrupciones globales
sei();
//Se muestra un mensaje a través del LCD
   PGM_P statetext = PSTR("AVR BUTTERFLY");
// Disable Analog Comparator (power save)
 ACSR = (1<<ACD);
 // Disable Digital input on PF0-2 (power save)
   DIDR0 = (7<<ADC0D);
```

```
 // Enable pullups
PORTB = (15<<PB0);
PORTE = (15<<PE4);

// Initialize pin change interrupt on joystick
Button_Init();
// initialize the LCD
LCD_Init();
// set Clock Prescaler Change Enable
CLKPR = (1<<CLKPCE);
 // set prescaler = 8, Inter RC 8Mhz / 8 = 1Mhz
CLKPR = (0<<CLKPS1) | (1<<CLKPS0);

//Configuración del USART a 207= 9600 baudios
USART_Init(207);
if (statetext)
{
  LCD_puts_f(statetext, 1);
   LCD_Colon(0);
 statetext = NULL;
}
//Lazo infinito
while (1)
 {
if (statetext)
```

```
        {
        LCD_puts_f(statetext, 1);
        LCD_Colon(0);
        statetext = NULL;
        }
        //Se espera a que sea presionado un botón
         input = Obtener_Boton();
         //Usart_Tx('1');
//Se realiza una determinada acción según el botón presionado
        switch (input)
        {
        case Centro:
         //Usart_Tx('0');
                statetext = PSTR("CENTRO");
            Usart_Tx('U');
          break;

        case Derecha:
  //Usart_Tx('0');
                statetext = PSTR("DERECHA");
     Usart_Tx('R');
   break;

        case Izquierda:
 //Usart_Tx('0');
```

```
                    //Usart_Tx('0');
                    statetext = PSTR("IZQUIERDA");
                Usart_Tx('I');
                    break;

            case Arriba:
 //Usart_Tx('0');
                    statetext = PSTR("ARRIBA");
    Usart_Tx('C');
    break;

  case Abajo:
          //Usart_Tx('0');
                    statetext = PSTR("ABAJO");
  Usart_Tx('a');
                    break;

 default:
  break;
          }
   }
   return 0;
}

/*Función que retorna un valor entero correspondiente al botón
```

```
presionado en el JoyStick */
int Obtener_Boton(void)
{
int Temp1;
        //PB4-->O  Centro
        //PB6-->A  Arriba
        //PB7-->B  Abajo
        //PE2-->C  Izquierda
        //PE3-->D  Derecha
        //Centro
Temp1=(PINB) & 0b00010000;
if(Temp1==0b00000000)
{
        sei();
        return Centro;
}
        //Arriba
Temp1=PINB & 0b01000000;
if(Temp1==0b00000000)
{
        sei();
        return Arriba;
}
        //Abajo
Temp1=PINB & 0b10000000;
```

```
        if(Temp1==0b00000000)
        {
                sei();
                return Abajo;
        }
                //Izquierda
        Temp1=PINE & 0b00000100;
        if(Temp1==0b00000000)
        {
                sei();
                return Izquierda;
        }
                //Derecha
        Temp1=PINE & 0b00001000;
        if(Temp1==0b00000000)
        {
                sei();
                return Derecha;
        }
        sei();
        return Otros;
}
        //Codigo del  usart
```

```
        #include <avr/io.h>
        #include "usart.h"
/* 25. May 2005 - adapted to avrlibc iom168.h version 1.17 */
void USART_Init(unsigned int baudrate) // UDRR Value not baudrate
        {
   UBRR0H = (unsigned char)(baudrate>>8);
        UBRR0L = (unsigned char)baudrate;
   UCSR0A = (1<<U2X0);
UCSR0B=(1<<RXEN0)|(1<<TXEN0)|(0<<RXCIE0)|(0<<UDRIE0;
UCSR0C=(0<<UMSEL0)|(0<<UPM00)|(0<<USBS0)|(3<<UCSZ00)|(0<<UCPOL0);
        }
void Usart_Tx(unsigned char data)
{
   while (!(UCSR0A & (1<<UDRE0)));
   UDR0 = data;
}
char Usart_Rx(void)
{
  while (!(UCSR0A & (1<<RXC0)));
  return UDR0;
}
        //Driver LCD
        #define REDUCED
        // Include files.
```

```
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <inttypes.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#ifndef REDUCED
// mt - only for AUTO:
#include "main.h"
#else
#include "mydefs.h"
#endif
// mt - for gButtonTimeout
#include "button.h"
#include "LCD_driver.h"
#ifndef BOOL
#define BOOL    char
#define FALSE   0
#define TRUE    (!FALSE)
#endif

// Variable from "button.c" to prevent button-bouncing
extern unsigned char gButtonTimeout;
volatile char gAutoPressJoystick = FALSE;
// Used to indicate when the LCD interrupt handler should update the LCD
// mt jw char gLCD_Update_Required = FALSE;
volatile char gLCD_Update_Required = FALSE;
```

```
// LCD display buffer (for double buffering).
volatile char LCD_Data[LCD_REGISTER_COUNT];

// Buffer that contains the text to be displayed
volatile char gTextBuffer[TEXTBUFFER_SIZE];

// Only six letters can be shown on the LCD.
volatile signed char gScroll;
volatile char gScrollMode;
////Start-up delay before scrolling a string over the LCD
char gLCD_Start_Scroll_Timer = 0;

// The gFlashTimer is used to determine the on/off
volatile char gFlashTimer = 0;
// Turns on/off the colons on the LCD
char gColon = 0;
// Look-up table used when converting ASCII to
// LCD display data (segment control)
unsigned int LCD_character_table[] PROGMEM =

/*************************************************************/
*   Function name : LCD_Init
*   Returns :       None
*   Parameters :    None
*   Purpose :       Initialize LCD_displayData buffer.
```

```
*                Set up the LCD (timing, contrast, etc.)
************************************************************/
void LCD_Init (void)
{
   LCD_AllSegments(FALSE);                  // Clear segment buffer.
   LCD_CONTRAST_LEVEL(LCD_INITIAL_CONTRAST);  //Set the LCD contrast
   // Select asynchronous clock source, enable all COM pins and enable all
   LCDCRB = (1<<LCDCS) | (3<<LCDMUX0) | (7<<LCDPM0);
   // Set LCD prescaler to give a framerate of 32,0 Hz
   LCDFRR = (0<<LCDPS0) | (7<<LCDCD0);
   LCDCRA = (1<<LCDEN) | (1<<LCDAB);          // Enable LCD and set low
power waveform
   //Enable LCD start of frame interrupt
   LCDCRA |= (1<<LCDIE);
   gLCD_Update_Required = FALSE;
}
/*****************************************************
*   Function name : LCD_WriteDigit(char c, char digit)
*   Returns :       None
*   Parameters :    Inputs
*   Purpose :       Stores LCD control data in the LCD_displayData buffer.
*                (The LCD_displayData is latched in the LCD_SOF interrupt.)
*****************************************************/
void LCD_WriteDigit(char c, char digit)
{
   unsigned int seg = 0x0000;               // Holds the segment pattern
```

```
char mask, nibble;
volatile char *ptr;
char i;
if (digit > 5)                          // Skip if digit is illegal
   return;
//Lookup character table for segmet data
if ((c >= '*') && (c <= 'z'))  // c is a letter
{
   if (c >= 'a')                        // Convert to upper case
      c &= ~0x20;                       // if necessarry
   c -= '*';       //mt seg = LCD_character_table[c];
seg = (unsigned int) pgm_read_word(&LCD_character_table[(uint8_t)c]);
      }
// Adjust mask according to LCD segment mapping
if (digit & 0x01)
   mask = 0x0F;            // Digit 1, 3, 5
else
   mask = 0xF0;            // Digit 0, 2, 4
ptr = LCD_Data + (digit >> 1);  // digit = {0,0,1,1,2,2}

for (i = 0; i < 4; i++)
{
   nibble = seg & 0x000F;
   seg >>= 4;
   if (digit & 0x01)
      nibble <<= 4;
```

```
        *ptr = (*ptr & mask) | nibble;
        ptr += 5;
    }
}
        /*************************************************
        *   Function name : LCD_AllSegments(unsigned char input)
        *   Returns :       None
        *   Parameters :    show -  [TRUE;FALSE]
        *   Purpose :       shows or hide all all LCD segments on the LCD
        *************************************************/
        void LCD_AllSegments(char show)
        {
           unsigned char i;
    if (show)
        show = 0xFF;
            // Set/clear all bits in all LCD registers
    for (i=0; i < LCD_REGISTER_COUNT; i++)
        *(LCD_Data + i) = show;
}
        /*************************************************
        *   LCD Interrupt Routine
        *   Returns :       None
        *   Parameters :    None
        *   Purpose: Latch the LCD_displayData and Set
        LCD_status.updateComplete
        *************************************************/
```

```
SIGNAL(SIG_LCD)
{
    static char LCD_timer = LCD_TIMER_SEED;
    char c;
  char c_flash;
   char flash;
   char EOL;
   unsigned char i;
    static char timeout_count;
  static char auto_joystick_count;
c_flash=0; // mt
/******** Button timeout for the button.c, START **********/
    if(!gButtonTimeout)
    {
timeout_count++;
  if(timeout_count > 3)
     {
         gButtonTimeout = TRUE;
timeout_count = 0;
   }
 }
/********** Button timeout for the button.c, END **********/
    if(gAutoPressJoystick == AUTO)
    {
       auto_joystick_count++;
```

```
    if(auto_joystick_count > 16)
    {
gAutoPressJoystick = TRUE;
auto_joystick_count = 15;
  }
}
    else
        auto_joystick_count = 0;
    LCD_timer--;
// Decreased every LCD frame
  if (gScrollMode)
    {
   // If we are in scroll mode, and the timer has expired,
   // we will update the LCD
   if (LCD_timer == 0)
   {
      if (gLCD_Start_Scroll_Timer == 0)
      {
        gLCD_Update_Required = TRUE;
}
else
    gLCD_Start_Scroll_Timer--;
     }
    }
    else
```

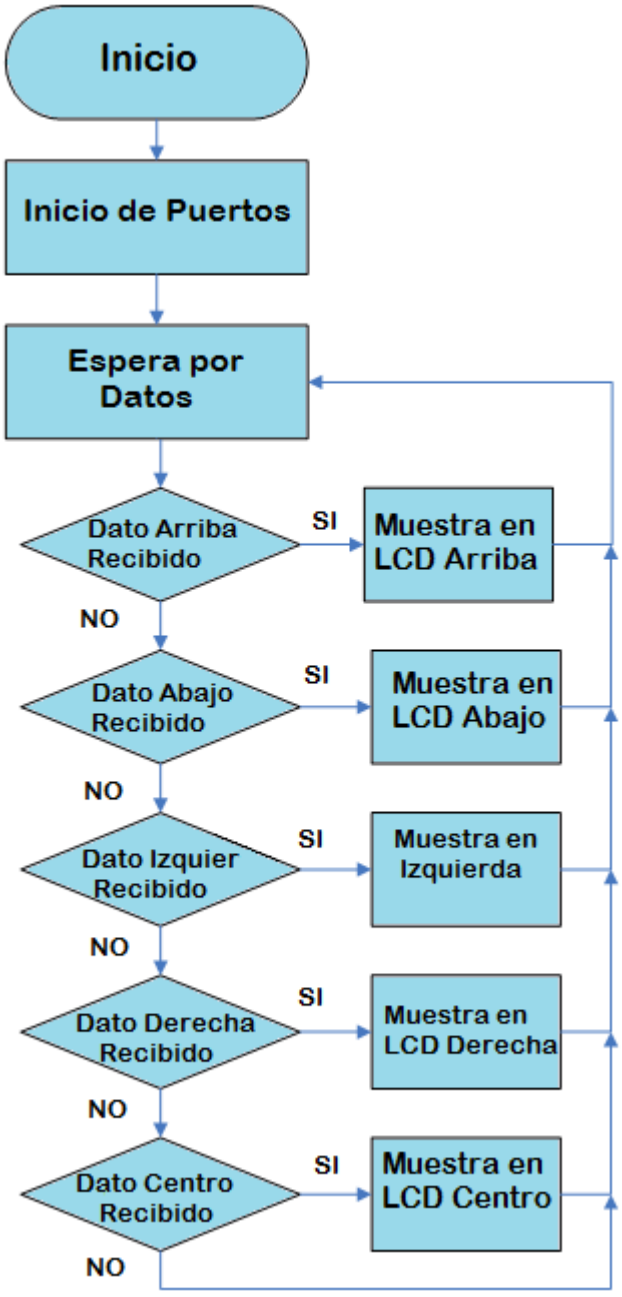
```
        {   // if not scrolling,
            // disble LCD start of frame interrupt
    //        cbi(LCDCRA, LCDIE);   //DEBUG
        gScroll = 0;
        }
        EOL = FALSE;
        if (gLCD_Update_Required == TRUE)
        {
     // Duty cycle of flashing characters
            if (gFlashTimer < (LCD_FLASH_SEED >> 1))
    flash = 0;
else
    flash = 1;
     // Repeat for the six LCD characters
for (i = 0; i < 6; i++)
{
    if ((gScroll+i) >= 0 && (!EOL))
    {

      // We have some visible characters
        c = gTextBuffer[i + gScroll];
        c_flash = c & 0x80 ? 1 : 0;
        c = c & 0x7F;
        if (c == '\0')
            EOL = i+1;      // End of character data
```

```
    }
    else
        c = ' ';
    // Check if this character is flashing
    if (c_flash && flash)
        LCD_WriteDigit(' ', i);
    else
        LCD_WriteDigit(c, i);
}
// Copy the segment buffer to the real segments
for (i = 0; i < LCD_REGISTER_COUNT; i++)
    *(pLCDREG + i) = *(LCD_Data+i);
// Handle colon
if (gColon)
    *(pLCDREG + 8) = 0x01;
else
    *(pLCDREG + 8) = 0x00;
// If the text scrolled off the display,
// we have to start over again.
if (EOL == 1)
    gScroll = -6;
else
    gScroll++;
// No need to update anymore
gLCD_Update_Required = FALSE;
```

```
    }
    // LCD_timer is used when scrolling text
    if (LCD_timer == 0)
    {
/*      if ((gScroll <= 0) || EOL)
          LCD_timer = LCD_TIMER_SEED/2;
       else*/
          LCD_timer = LCD_TIMER_SEED;
    }
    // gFlashTimer is used when flashing characters
    if (gFlashTimer == LCD_FLASH_SEED)
       gFlashTimer= 0;
    else
       gFlashTimer++;
}
```

3.2.4 Diagrama de Flujo del Receptor



3.7 Diagrama de Flujo Receptor

3.2.5 Código Receptor

```
//Declaración de constantes
#define F_CPU 20000000UL
#define BAUD 4800
#define MYUBRR F_CPU/16/BAUD-1
#define PIND_MASK ((1<<PIND0)|(1<<PIND1))

//Librerias a usar
#include <avr/pgmspace.h>
#include <pololu/orangutan.h>

//Prototipo de procedimientos
void USART_Init( unsigned int );
unsigned char ReceiveByte (void);

//Programa principal
int main ()
{
//Declaración de variable
int i, j=0;

//Seteo de puertos
DDRD = 0xFE;
PORTD |= PIND_MASK;
DDRB = 0x08; // set PORTD for output
```

```
PORTB = 0x00; // set LEDs off
//Configuración de USART a 2400 baudios
USART_Init(521);
//Lazo infinito en espera de caracter recibido
        while(1)
        {
        //Se muestra en pantalla un mensaje que indica que
        //se está a la espera de recibir un caracter
        clear();
        print("ESPERO");
        lcd_goto_xy(0, 1);
        print("TU ORDEN");
        delay_ms(500);

        //Se toma la decisión en función del caracter
        //recibido. El arranque del motor se lo realiza
        //en dos tiempos para no realizar una demanda súbita
        //de potencia

        //ADELANTE
        i = ReceiveByte();
        if(i!=j)
        {
                if(i == 0X5E)
                {
```

```
            clear();
            print("FORWARD");
            set_motors(50,50);
            delay_ms(1000);
            delay_ms(1000);
            set_motors(100,100);
}

            //ATRÁS
else if (i == 0X4F )// 'a'
{
            clear();
            print("BACK");
            set_motors(-50,-50);
            delay_ms(1000);
            set_motors(-100,-100);
            delay_ms(1000);
}
//IZQUIERDA
else if (i == 0X5B)//'I'
{
            clear();
            print("LEFT");
            set_motors(50,-50);
            delay_ms(500);
```

```
            set_motors(100,-100);
            delay_ms(500);
}
//DERECHA
else if (i == 0XAB)
{
            clear();
            print("RIGHT");
            set_motors(-50, 50);
            delay_ms(500);
            set_motors(-100, 100);
            delay_ms(500);
}

//ALTO
else if (i == 'U')
{
            clear();
            print("Stop");
            set_motors(0,0);
            delay_ms(250);
}
j=i;
}
```

```
                }
        }

//Implementación de procedimiento/*
Se configuran los registros para la transmisión por USART
de manera que el dato recibido indica el BAUDRATE al cual
se va a trabajar
*/
void USART_Init(unsigned int baudrate)
{
  // Set baud rate
 UBRR0H = (unsigned char)(baudrate>>8);
 UBRR0L = (unsigned char)baudrate;
        //UCSR0A = (0<<U2X0);
 // Enable receiver and transmitter
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);
  // Async. mode, 8N1
  UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}

//Implementación de función
/*
Función que espera a la recepción de un caracter y luego
retorna el dato recibido al programa principal
*/
```

```
unsigned char ReceiveByte (void)
{
 /* Wait for incomming data */
 while (!(UCSR0A & (1 << RXC0)));

  /* Return the data */
 return UDR0;
}
```

```
//Codigo del usart
void USART_Init(unsigned int baudrate);

void Usart_Tx(char);
char Usart_Rx(void);
```

CAPÍTULO 4

4.1 SIMULACIÓN Y PRUEBAS DEL PROYECTO

En este capítulo se muestran las pruebas de la implementación y las simulaciones del proyecto, sus análisis sus resultados.

4.2 Pruebas de Funcionamiento.

Para el desarrollo del proyecto primero se realizaron pruebas con el Butterflay que viene en el proteus y también se realizaron pruebas con el Pololu seguidor de línea.

4.3 Pruebas de simulación del Transmisor y Receptor

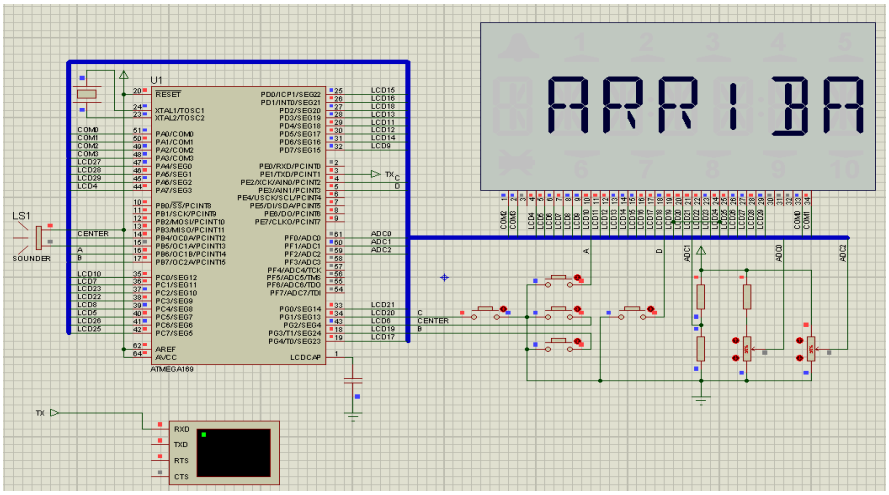


Figura 4.3 simulación pololu avanza

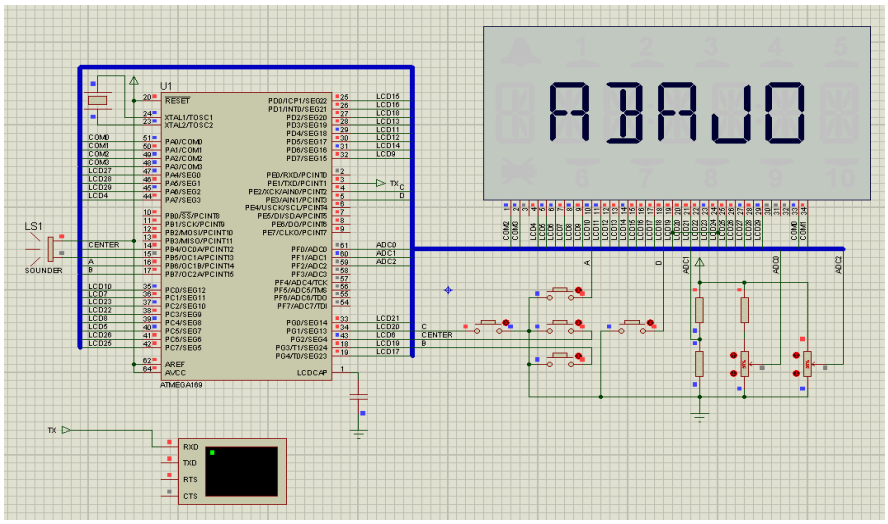


Figura 4.3 b simulación hace que el pololu retroceda

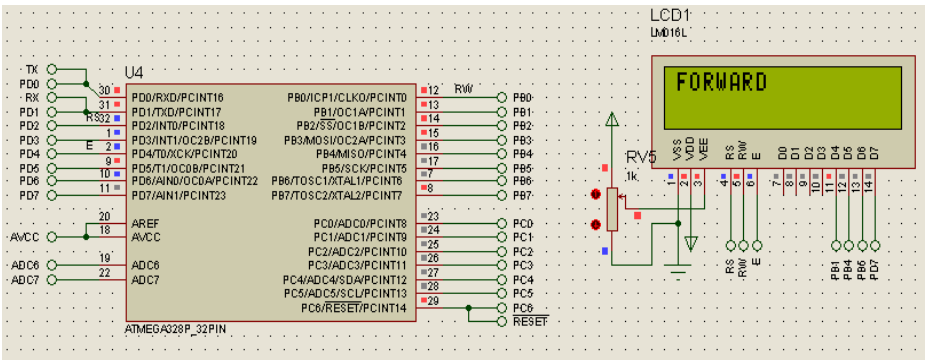


Fig 4.3 c simulacion del receptor

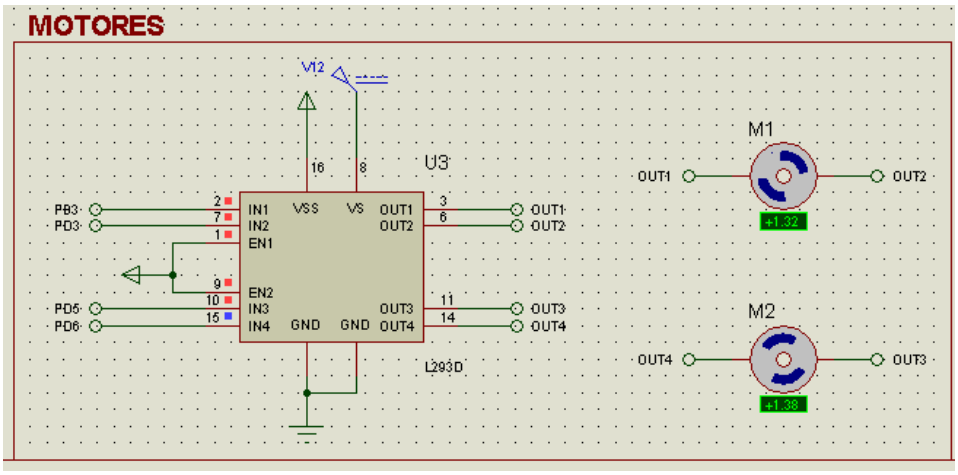


Fig. 4.2.1: Motores ejecutando el desplazamiento

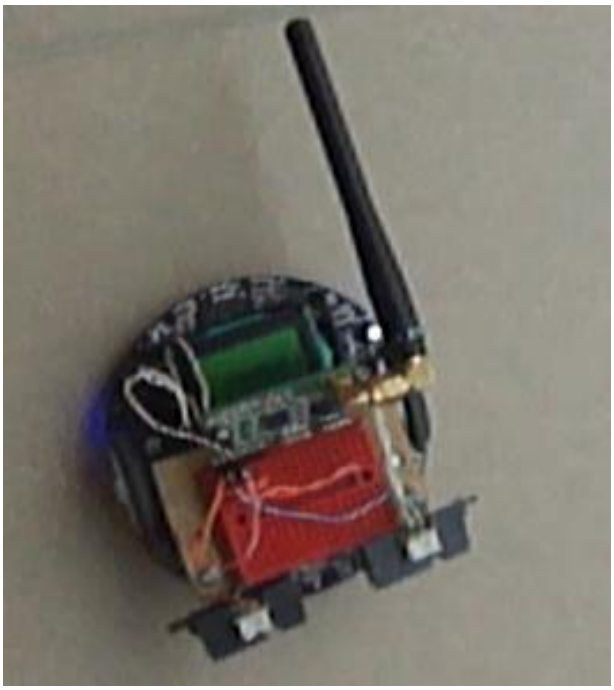


Figura 4.3 e Robot pololu 3pi

4.4 Resultado de la simulación

De la simulación se pudo obtener una representación del funcionamiento del proyecto, se pudo observar al enviarse los datos del trasmisor y los datos que recibía el receptor mostrando en el LCD los mensajes correspondientes, también se pudo visualizar el movimiento de los motores del pololu que este hace que se desplace, llegando a obtener lo que se deseaba los resultados fueron iguales que los de la simulación.

CONCLUSIONES

1. Al realizar el proyecto nos hemos familiarizado con los micro controladores de la familia ATMEL conociendo las características para su correcto funcionamiento de igual forma se pudo trabajar con las herramientas que ofrece el AVR studio 4, para programar este tipo de micro controladores conociendo las beneficios y limitaciones del robot pololu 3pi y el Butterfly.
2. El Robot Pololu es un dispositivo muy útil en el campo de la Robótica ya que este puede realizar recorridos evitando obstáculos siguiendo a un móvil o no y puede ser controlado inalámbricamente o vía remota.
3. El Kit AVR Butterfly es una poderosa herramienta de aprendizaje, es práctico, eficaz y muy amigable; que con el desarrollo del proyecto se va descubriendo progresivamente las características del micro controlador ATmega169.

RECOMENDACIONES

1. Es necesario revisar las hojas de especificaciones antes de trabajar con los dispositivos y en el caso de el Butterfly y el Robot Pololu revisar su user guide ya que ahí dan las recomendaciones para trabajar con ellos.

2. Es recomendable que las baterías para que el pololu trabaje este bien cargadas ya que este puede no permitir grabar bien el programa o más aun que el programador se queme.

3. Fijar bien las frecuencias de trabajo para que el buttetfly como para el pololu pueda transmitir, ya que si ambas no tienen igual frecuencias no transmite y entonces el robot no se mueve

ANEXOS

ANEXOS 1

ESPECIFICACIONES Y DIAGRAMAS DEL MICROCONTROLADOR ATMEGA 328

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
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - 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 I^2C 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
- I/O and Packages
  - 23 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 for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

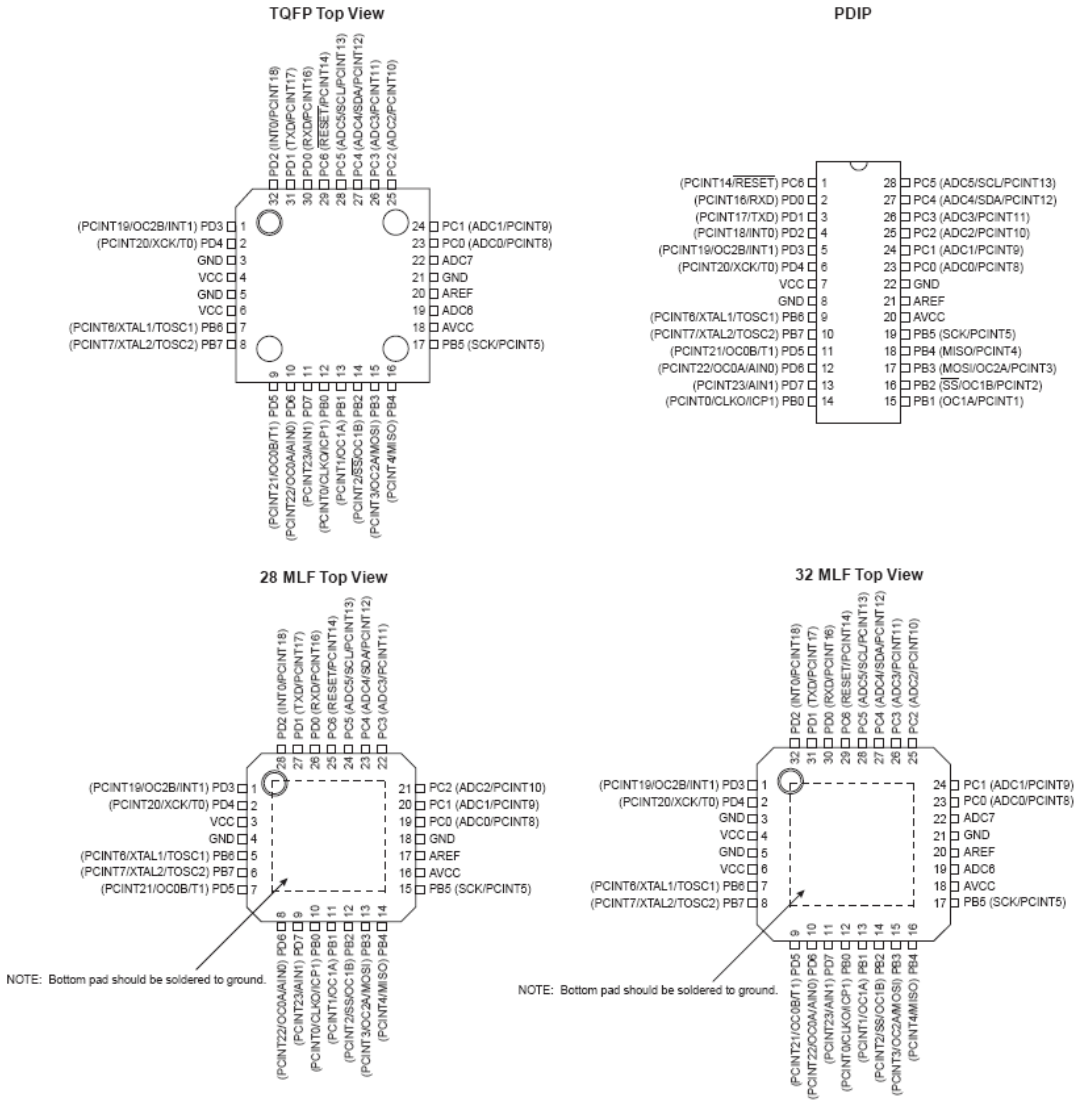
8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09

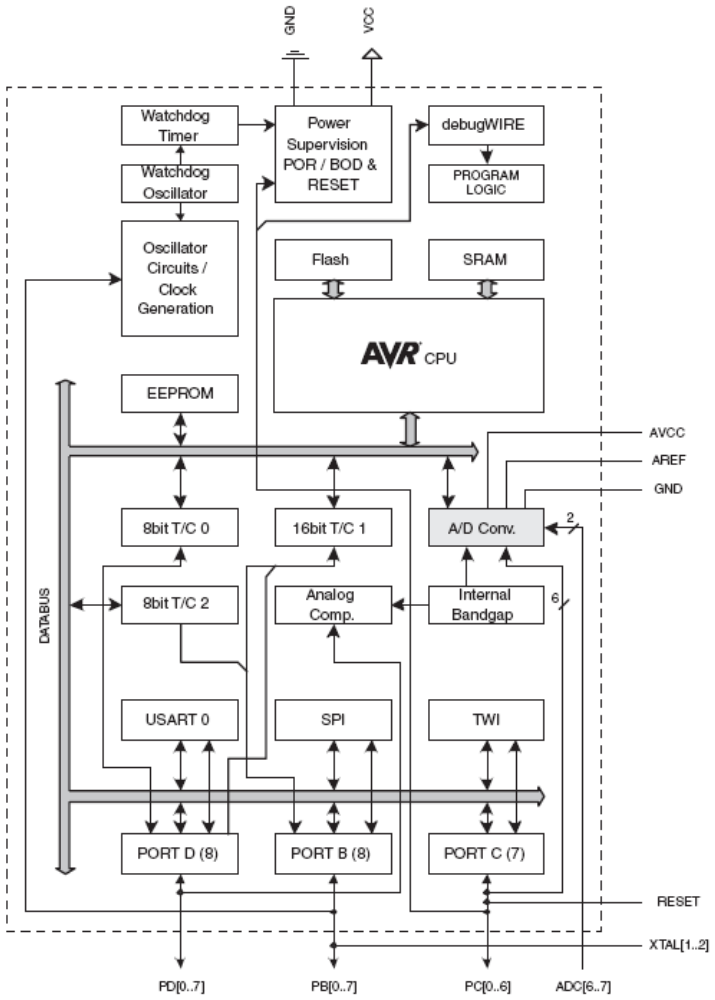
1. Pin Configurations

Figure 1-1. Pinout ATmega48PA/88PA/168PA/328P



2.1 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

Diagrama de Bloques del microcnontrolador Atmega 328

ANEXOS 2

ESPECIFICACIONES Y DIAGRAMAS DEL MICROCONTROLADOR ATMEGA 169

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 16K bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - 4 x 25 Segment LCD Driver
  - 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
  - Four PWM Channels
  - 8-channel, 10-bit ADC
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Universal Serial Interface with Start Condition Detector
  - 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
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Speed Grade:
  - ATmega169V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega169: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
- Temperature range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode:
    1 MHz, 1.8V: 350µA
    32 kHz, 1.8V: 20µA (including Oscillator)
    32 kHz, 1.8V: 40µA (including Oscillator and LCD)
  - Power-down Mode:
    0.1µA at 1.8V

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

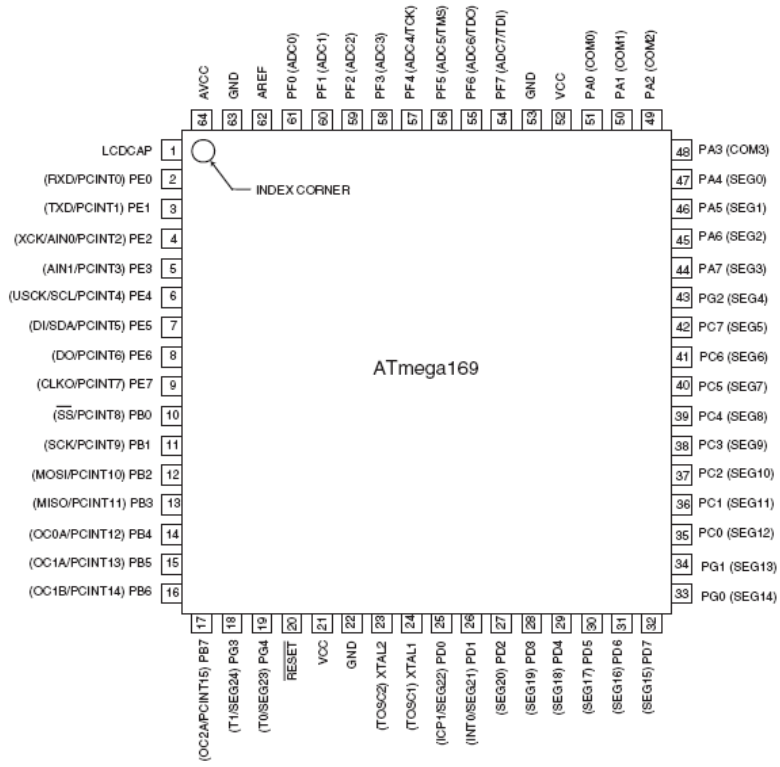
ATmega169V
ATmega169

Notice:
Not recommended in new designs.

2514P–AVR–07/06

Pin Configurations

Figure 1. Pinout ATmega169



Note: The large center pad underneath the QFN/MLF packages is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.

Disclaimer

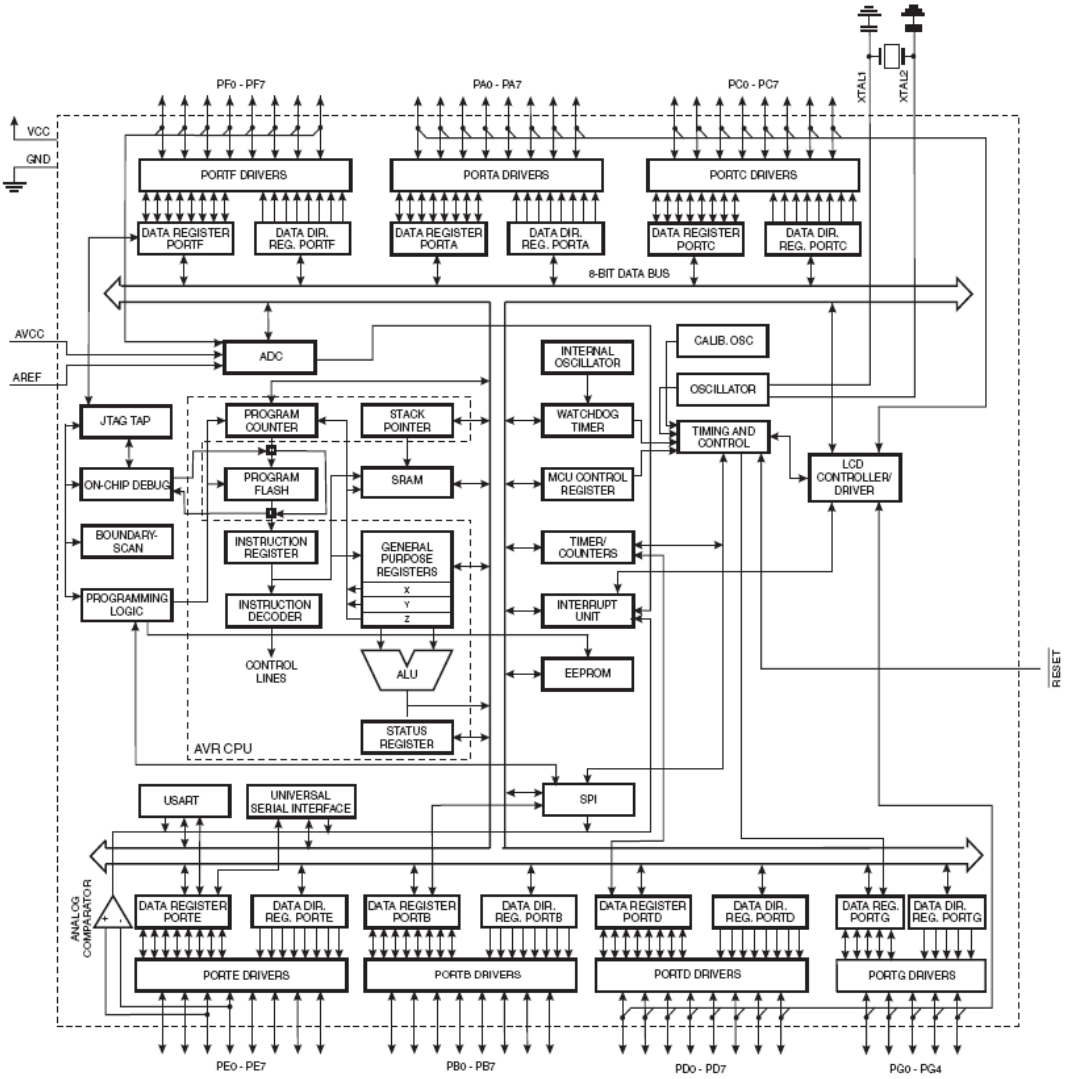
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega169 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 ATmega169 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

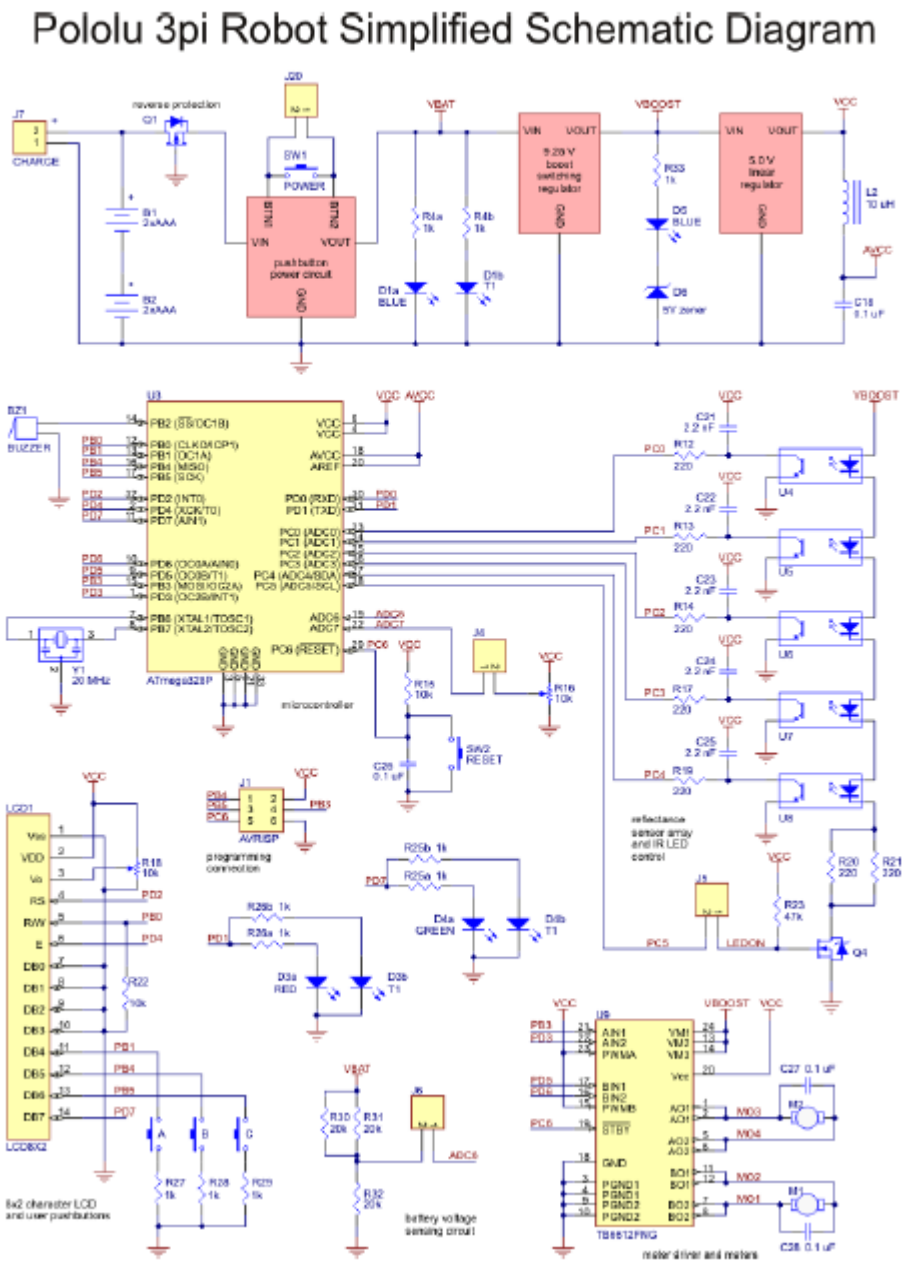
Figure 2. Block Diagram



ANEXOS 3

ESQUEMA DEL CIRCUITO SIMPLIFICADO DEL ROBOT POLOLU 3PI

5.e 3pi. Esquema del circuito simplificado.



Referencias Bibliográficas

[1] Osar Gonzales , Hoja del Robot Pololu:Robotica

http://www.pololu.com/

Fecha de consulta 20/04/11

[2] Pololu Corporation, Guía de usuario del Robot Pololu

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

Fecha de Consulta 20/04/11

[3] Súper Robótica, Proyectos similares

http://www.superrobotica.com/S320107.htm

Fecha de Consulta 22/04/11

[4] Atmel Corporation , Hoja de Datos del microcontrolador Atmega 169

http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

Fecha de consulta 5/04/11

[5] Atmel Corporation , Hoja de Datos del microcontrolador Atmega 328

http://www.atmel.com/dyn/products/product_card.asp?part_id=3012

Fecha de consulta 5/04/11.

[6] Atmel Corporation, AVR Butterfly, user guide

http://www.atmel.com/dyn/resources/prod_documents/doc4271.pdf

Fecha de consulta 25/ 04/11.

[7] HOPE MICROELECTRONICS CO LTD, modulo HM-TR
thttp://www.roboeq.com/PDF/0501018.pdf

Fecha de consulta 03/05/11

[8] Recursos dados para la utilización de las librerías, programadores para el AVR butterfly así como para el pololu 3 pi,
http://www.pololu.com/catalog/product/1227/resources

Fecha de consulta: 20/01/11.

[9] 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: 10/01/11.