ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Optimización de algoritmos empleados en la resolución de laberintos usando técnicas de control PID con el robot Pololu 3pi e incorporación de control inalámbrico por radio frecuencia"

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:

Alvarado Avila Andrea Annabell

Anchundia Peralta Máximo Rubén

GUAYAQUIL – ECUADOR

Año: 2011

A G R A D E C I M I E N T O

Agradezco en primer lugar a Dios por llenar mi vida de dicha y bendiciones, a mis padres Santy y Domingo por su amor, cariño y comprensión, a mi hermano Andrés por la compañía y el apoyo. También a toda mi familia que siempre ha estado para darme todo su apoyo.

De manera especial agradezco a aquella persona que me empuja a salir adelante siempre. Aunque no sea parte de mi vida pero siempre su recuerdo me da el ánimo necesario.

Andrea Annabell Alvarado Avila

Agradezco a Dios por todas las bendiciones recibidas y el permitirme realizar este trabajo. A mi madre, mi padre, familiares y amigos que han caminado a mi lado en los momentos difíciles. A la vida.

Máximo Rubén Anchundia Peralta

D E D I C A T O R I A

Este trabajo lo dedico a mis padres, de manera especial a mi madre Santy que con su gran amor de madre ha tratado siempre de apoyarme en todo lo que ha podido sin dudarlo.

A mi hermano y a mis tías que siempre están pendientes de lo que me pasa y me brindan su cariño incondicional.

Andrea Annabell Alvarado Avila

Este trabajo lo dedico a todas las personas incondicionales que pudieron hacer que llegue hasta esta instancia,especialmente a mi madre que siendo una mujer luchadora supo darme todo de ella para poder salir adelante brindándome su amor y compresión.

A mi abuelito Justino que aunque ya no se encuentre en esta vida, sé que desde el cielo ha de estar colmándome de bendiciones.

Máximo Rubén Anchundia Peralta

TRIBUNAL DE SUSTENTACION

Ing. Carlos Valdivieso A.

Profesor de Materia de Graduación

Ing. Hugo Villavicencio V.

Profesor Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido expuestos en tesina de Graduación nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

_____

Alvarado Avila Andrea Annabell

Anchundia Peralta Máximo Rubén

RESUMEN

La robótica en los últimos tiempos ha venido evolucionando para estar a la par con las nuevas tendencias tecnológicas. Un robot es un dispositivo, equipo o aparato capaz de recibir órdenes y de actuar para satisfacerlas, manipulando objetos y modificando su entorno en un tiempo definido, utilizando algoritmos o procedimientos bien establecidos para realizarse en un número finito de pasos, y se perfilan como grandes ayudas a la sociedad moderna para la realización de los trabajosrepetitivos, de peligro, y de gran precisión; sirviendo además en las fases del proceso educativo como medios didácticos de incalculable valor.

Es así que constantemente aparecen nuevos avances tecnológicos y uno de estos es el denominado robot Pololu 3pi

usado por personas con mayor experiencia en robótica para el desarrollo de procesos a nivel industrial para diferentes aplicaciones técnicas.

Es por ello que este proyecto desarrollará un sistema que permita el monitoreo de una determinada área de interés por parte del usuario, además el uso de una interfaz que permita al interesado realizar la adquisición de datos de un circuito a otro mediante una conexión inalámbrica como son los módulos RF.

Para lograr este fin se usarán los conocimientos de programación realizadas en el AVR Studio para la movilidad del Robot y el uso de módulos para tener la conexión inalámbrica deseada entre dos circuitos.

ÍNDICE GENERAL

RESUMEN

INDICE GENERAL

INDICE DE FIGURAS

INTRODUCCION

Capítulo 1……..……
…..1

1 DESCRIPCIÓN GENERAL DEL SISTEMA……..……...……. 1

1.1
Introducción…….……
…. 1

1.2
Antecedente…….…..
.... 3

1.3 Descripción general del
Proyecto……..... 5

1.4 Análisis y Justificación del
sistema……..... 6

1.5 Alcance del Proyecto y soluciones
similares……..…… 7

Capítulo
2……..……..……
….10

2 Análisis

Teórico………..……..…….....

10

2.1 Herramientas de software……

……..….....10

2.1.1 Compilador AVR Studio

4...…….....10

2.1.2 Simulador Proteus(Version 7.7

)….……...……..……12

2.1.3 Pic C…….…….....15

2.2 Herramientas Hardware……..…..16

2.2.1 3pi Robot Pololu…….……....16

2.2.2 Microcontrolador ATmega 328P…….. ….18

2.2.3 Microcontrolador Pic 16F88 ……..……..20

2.2.4 Pololu USB AVR programmer…….…..21

Capítulo 3……... …22

3 DISEÑO E IMPLEMENTACIÓN DEL PROYECTO……..…….....22

3.1 Detalles de Diseño Propuesto

……...….22

3.2 Diseño del

Sistema……....25

3.3 Análisis del programa AVR

Studio…… 26

3.3.1 Programa para la movilidad del robot

Pololu..... 27

33.2 Programa para la movilidad del robot usando el

Butterfly…….. 33

Capítulo

4..……

….39

4 SIMULACIÓN Y PRUEBAS

EXPERIMENTALES…….....……39

4.1 Implementación y pruebas del

proyecto…….…..…..……39

CONCLUSIONES……

…….43

RECOMENDACIONES……

……..45

ANEXO

A…….

……47

BIBLIOGRAFÍA……

……...51

ÍNDICE DE FIGURAS

Figura 1.1: RobotPololu.....1

Figura 1.2: Forma del Laberinto 2

Figura 1.3: Modulo ASK 3

Figura 1.4: RobotSpirit.....4

Figura 1.5: Robot explorador de terrenos…….8

Figura 1.6: AVR Butterfly..... ……..9

Figura 2.1: AVR Studio……....11

Figura 2.2: Simulador Proteus…….….14

Figura 2.3: Interfaz Gráfica del Pic C……...….....15

Figura 2.4: Diagrama Esquemático del Pololu…….....16

Figura 2.5: Diagrama del motor DC con puente H…….…...…17

Figura 2.6: Funcionamiento de los motores D …

……..…….18

Figura 2.7: PIC ATmega 328P
…….19

Figura 2.8: PIC
18F886……20

Figura 2.9: Pololu USB AVR
Progammer……..…….…….21

Figura 3.1: Parte Superior del Butterfly.....23

Figura 3.2:Scope del Programador.....24

Figura 3.3: Control PID Discreto.....25

Figura 3.4: Diagrama de bloques del sistema.....26

Figura 3.5 : Tabla de Configuración para el Puerto Serial 35

Figura 4.1: Partes del Programador 39

Figura 4.2: Adquisición de datos por HyperTerminal 40

Figura 4.3: Prueba en Proteus.....41

Figura 4.4: Administrador de Periféricos 42

INTRODUCCIÓN

Este proyecto de graduación tiene como objetivo destacar el uso de los sensores reflectivos para la resolución de algoritmos del Robot Pololu 3Pi bajo la programación de AVR Studio con sus respectivas librerías para la respectiva programación del mismo.

En el capítulo 1 se mencionan los objetivos y el alcance del Proyecto. También se realizó un estudio de soluciones similares en el mercado.

En el capítulo 2 se indicarán la base teórica de lo que se utilizó. Se mencionan el hardware y el software para el desarrollo de este proyecto.

En el capítulo 3 se presentará el diseño del proyecto, sus protocolos de comunicación, algoritmos de transmisión y de movilidad del robot.

En el capítulo 4 se mostrará la implementación y las respectivas pruebas desarrolladas en el robot.

Finalmente se presentan las conclusiones y recomendaciones que hemos obtenidos con la resolución del proyecto.

CAPITULO 1

1 DESCRIPCIÓN GENERAL DEL SISTEMA

1.1 Introducción

Este sistema se ha desarrollado usando una de las innovaciones de la tecnología en la robótica y consiste en un pequeño robot con una base en forma de disco con dos motores para el movimiento del mismo. Podemos apreciar en la figura 1.1, las diversas partes que lo componen al robot Pololu 3pi, el cual se moverá en diferentes sectores del laberinto y si el usuario lo desea, lo puede hacer mover mediante un control inalámbrico incorporado al mismo.

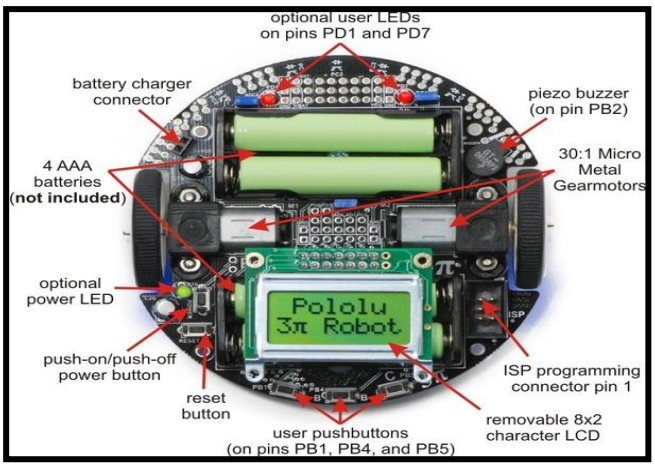

Figura 1.1 Robot Pololu 3pi

Este reporte tiene por objeto el destacar el uso del Robot 3pi resolviendo laberintos utilizando los sensores infrarrojos para la detección de las líneas que van hacer dibujadas en forma de laberinto, estos sensores vienen incluidos en nuestro Pololu 3pi, a0demás el lenguaje de programación estará bajo los esquemas del AVR Studio.

Para poder alcanzar los objetivos sin ningún problema debemos tener en cuenta que las líneas del laberinto deben de ser de color negro como lo podemos apreciar en la figura 1.2, para una mejor respuesta de los sensores, la información de los sensores y el control de giros de los motores será tratada por el **ATmega328P** que también viene incluido en el Pololu 3pi [1].

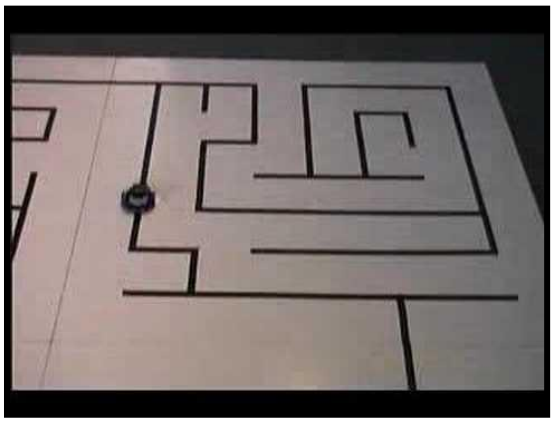

Figura 1.2 Forma del laberinto

Otra característica que va a tener el robot es que puede ser manejado a distancia, en nuestro proyecto utilizaremos los módulos ASK que se observan en la figura 1.3. La forma de trasmitir será por modulación, desplazamiento de amplitud. De esta manera los datos se trasmiten por amplitud, su voltaje varía de 3 a 12v, los módulos a utilizarse serán el TLP433 y el RLP433. Su rango de frecuencia será de 433Mhz.

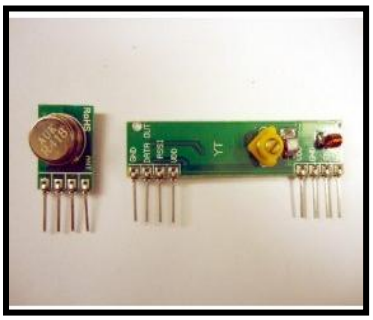

Figura 1.3 Módulos ASK

1.1 Antecedentes

A lo largo de la historia el hombre se ha fascinado por seres artificiales que le acompañen en su morada, seres que realicen sus tareas respectivas, tareas pesadas o difíciles de realizar por un ser humano, y que no decirse sobre la resolución de problemas entre una de ellas podemos enfocarnos en el de resolución de laberintos.

Hay varios tipos de ejemplos en la página del Pololu a manera de guías para mostrar cómo pueden ser usados los diversos tipos de componentes del 3Pi, así como también la manera de ejecutar comportamientos más complejos como la resolución de laberintos la cual nos sirve como parte de nuestro proyecto.

Este robot utiliza el software GNU C/C++ que trabaja perfectamente con el 3pi, Atmel´s AVR Studio provee un cómodo ambiente de desarrollo y el extenso set de librerías del Pololu que hace verdaderamente fácil la interfaz con todo tipo de hardware[2].

En la actualidad existen modelos similares al que se desarrollará en este proyecto como por ejemplo el robot "Spirit" que se aprecia en la figura 1.4, que desde enero del 2004 empezó a enviar fotografías de la superficie de Marte, cuyo objetivo es el control inalámbrico, recolección y análisis de datos para determinar si en Marte alguna vez hubo agua, este fue nuestra motivación para diseñar nuestro robot con capacidad de control inalámbrico de los movimientos.

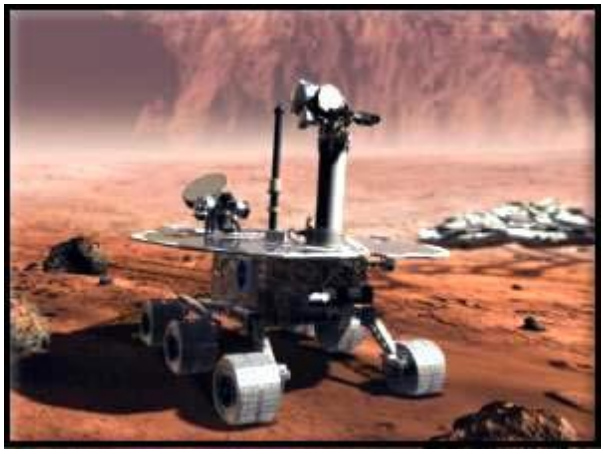

Figura 1.4 Robot Spirit

1.2 Descripción General del Proyecto

La aplicación de este proyecto consiste en la utilización del Pololu 3pi robot para la resolución de laberintos, esto se lograría a partir de una optimización de algoritmos partiendo de un algoritmo que tendrá lugar a que el robot se haga un arreglo de rutas a recorrer donde tendrá memorizado las posibles alternativas que tiene para ir de la entrada a la salida y por último colocaremos el robot en la posición inicial y este podrá llegar a la meta sin tener problemas de pérdida de secuencias.

Usaremos técnicas de controlador PID para mejorar la respuesta dinámica del robot para una referencia de posición sobre el efecto final, así como también para eliminar el error en estado estacionario de dicha posición.

Para la programación del robot usaremos el **ATmega328** que viene incluido en el robot y para llegar a éste hicimos uso del AVR Studio que es un paquete de desarrollo integrado de la empresa Atmel's con (IDE) que trabaja en armonía con el compilador WinAVR's. El AVR Studio nos permitirá cargar las diferentes librerías para los tipos de sensores y elementos que conforman el Pololu. El lenguaje a utilizar es C el cual facilita la programación del sistema, el desarrollo de los códigos a emplear serán mostrados en los capítulos posteriores[1].

También haremos uso de los módulos RF para poder trabajar de manera inalámbrica enviándole señales al robot y comandar a este de manera remota por medio de un enlace. Los datos de los sensores del robot llegan al trasmisor por un enlace y los comandos para los movimientos llegan al robot por otro enlace.

1.3 Análisis y justificación del sistema

Para la realización de este proyecto hay que considerar el hecho de que se partió de un sistema robótico que cuenta con sensores, micromotores, una pequeña pantalla LCD que permite la interacción con datos y variables que pueden ser manipuladas. El robot Pololu 3pi cumplió con la arquitectura básica deseada para desde allí poder

ampliar o destacar el uso de sensores en este trabajo, los sensores utilizados son de reflexión que nos sirven para el desarrollo de la aplicación.

Hemos empleado el ambiente de desarrollo AVR Studio, ya que este nos da la oportunidad de no tan solo programar en assembler, sino de integrar un compilador para C. Ambos son gratuitos y se pueden obtener accediendo al website de ATMEL. La característica principal del **AVR STUDIO** es su fácil manejo, además de esto tiene librerías propias para poder trabajar con los sensores del Pololu de forma directa, en cambio si utilizamos otra herramienta de compilación para programar tendríamos que definir nuevas librerías y configurar los puertos, también permite visualizar rápidamente lo que ocurre con los registros y poder modificar sin ningún problema los registros de memoria del microcontrolador.

Los sensores infrarrojos utilizados son 5 que vienen incluidos en el robot Pololu 3pi, se usan estas clases de sensores porque son de gran utilidad cuando se trata de la detección de color negro, están distribuidos de una manera tal que pueden sin ningún problema escoger las diferentes rutas que se encuentran en el laberinto [1].

Además de esto hemos utilizado los módulos con radiofrecuencia de 433 MHz (TLP y RLP) con modulación ASK ya que trabajaremos con amplitud, la utilización de estos módulos es debido a su bajo costo ya que con un buen algoritmo esto tendría una eficiencia que llegaría a un 80% en comparación a los módulos FSK que presentan mejor envío de mensajes pero el consumo de energía es mayor.

1.4 Alcance del proyecto y soluciones similares

Se considera que un proyecto como este podría tener algunas aplicaciones como por ejemplo en el campo de la industria, desactivadores de explosivos, exploración de terrenos no aptos para el hombre como se observa en la figura1.5, debido a que no se lo ha orientado a una aplicación específica. Tan solo cambiar un sensor o hacer pequeñas innovaciones se lo puede aplicar para otro campo aparte de la que se ha especificado en este proyecto.

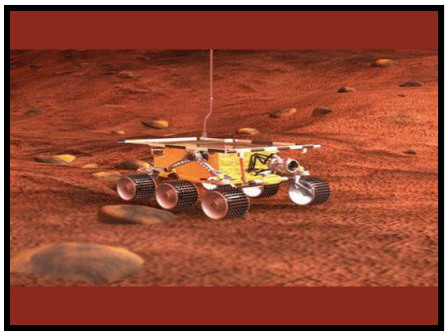

Figura 1.5 Robot explorador de terrenos

Un uso importante que le podemos dar a esta aplicación es cambiar los sensores infrarrojos por sensores detectores de bombas ya que con esto se dotaría a las fuentes de seguridad frente a la amenaza terrorista que cada día pone en peligro a cientos de vidas en el mundo, el robot se encargaría de buscar hasta encontrar su objetivo, y enviar los datos mediante comunicación inalámbrica de donde queda localizado el propósito.

También podemos utilizar las técnicas de control PID para manejar velocidad de un robot móvil y a su vez setear una velocidad de trabajo y el control PID se encargaría de controlar esta velocidad del motor, estas técnicas son muy usadas en la industria para el control de las velocidades de los motores para los procesos que necesitan cierta velocidad indiferente de las perturbaciones que afecten al sistema por ejemplo en envasado, banda transportadora.[3].

La placa de desarrollo Butterfly aparte de tener un microcontrolador de gama alta, tiene un tamaño muy adecuado para diferentes aplicaciones y a su vez tiene un sin número de módulos para diferentes estudios, entre una de las características que encontramos fue una pantalla de cristal líquido y un joystick que podrían ser perfectos para la manipulación de cualquier otro módulo pues podríamos hacer una comunicación inalámbrica o por computadora, entre algunas de las opciones que tendríamos mediante la

computadora es enviar la temperatura del ambiente y a su vez desde ésta podría enviarse comandos de control para cambiar la temperatura del ambiente [7], Ver figura1.6.

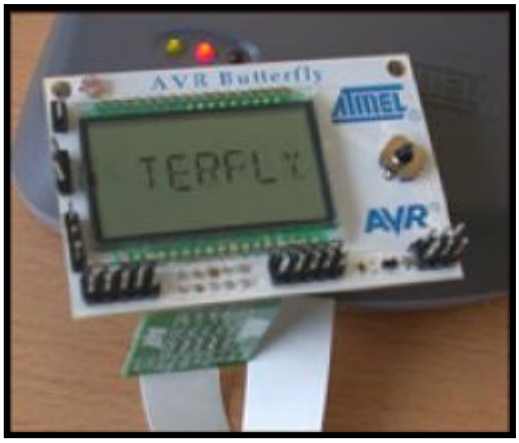

Figura 1.6 AVR Butterfly

CAPÍTULO 2

2 ANÁLISIS TEÓRICO

Este proyecto se fundamenta en la optimización de algoritmos empleados en la resolución de laberintos utilizando técnicas de control PID con el robot Pololu 3pi e incorporación de control inalámbrico por radio frecuencia. Las herramientas para la aplicación del proyecto se dividen en software y hardware las cuales describiremos más adelante.

2.1 Herramientas de Software para el desarrollo del proyecto

2.1.1 Compilador AVR Studio 4

El compilador AVR Studio 4 es un sistema integrado (IDE) de entorno de desarrollo para la depuración de AVR software. El AVR incluye un ensamblador y simulador de ASM y en combinación con WinAVR, el cual es compatible con C/C++.

Con el fin de contar con una plataforma AVR de desarrollo completa se requiere de lo siguiente:

- GCC: que es la colección de compiladores de GNU, configurado y compilados para nuestros objetivos de AVR.
- Paquete BinUtils, con las opciones específicas de AVR habilitadas.
- Biblioteca AVR LibC.
- Programador (ISP Programmer)

WinAVR. Es un conjunto de herramientas de archivo ejecutable de código abierto de desarrollo de software para la serie de Atmel AVR de microprocesadores RISC en la plataforma Windows. Esto incluye el compilador GNU GCC para C y C++.

AVR Libc es una librería de acceso a las funciones específicas, cuyo objetivo es proporcionar una biblioteca de C de alta calidad para su uso con GCC en los microcontroladores ATmel AVR,ver figura 2.1. [2]

.

Figura 2.1 AVR Studio

Características del Compilador AVR Studio 4

Podemos citar entre otras que las más importantes características para el compilador son:

- Escribir programas en C en AVR Studio.
- Compilar en un archivo hexadecimal. Utilizando el compilador AVR-GCC (que se integra en AVR Studio).
- Simular el objetivo AVR chip y depurar el código en AVR Studio.
- Programa el chip real utilizando el AVRISP mkII dispositivo USB, que se une a nuestro chip de destino con un cable especial de 6 pines.
- Una vez programado, el chip se ejecuta el programa en su circuito.

2.1.2 Proteus (Versión 7.7)

Proteus Virtual System Modelling (VSM) combina el modo mixto de simulación de circuitos SPICE, componentes animados y modelos de microprocesador para facilitar la co-simulación de diseños de microcontroladores, ver figura 2.2.

Esto es posible porque se puede interactuar con el diseño de indicadores de la pantalla utilizando como LED y pantalla LCD y actuadores, tales como interruptores y botones. La simulación se lleva a cabo en tiempo real.

Proteus VSM incluye una serie de instrumentos virtuales que incluye un osciloscopio , analizador de lógica, generador de funciones , generador de patrones , contador de números y de terminales virtuales , así como voltímetros y amperímetros sencillos. Además, ofrecemos dedicada Maestro / Esclavo / Monitor de analizadores de protocolo de modo de SPI y I2C simplemente los cables en las líneas de serie y monitor o interactuar con los datos en tiempo real durante la simulación. Una valiosa y barata forma realmente para obtener el software de comunicación de la derecha antes de la creación de prototipos de hardware.

Si desea tomar medidas detalladas en los gráficos, o el análisis de otros tipos, como la frecuencia, la distorsión, el ruido o los análisis de barrido de los circuitos analógicos, puede comprar la opción de simulación avanzada . Esta opción también incluye “Conformidad Análisis” una poderosa herramienta única y de Calidad de Software Assurance.

Aunque Proteus VSM es único en su capacidad para ejecutar cerca de simulaciones en tiempo real de sistemas completos de micro-controlador, su verdadero poder proviene de su capacidad para realizar estas simulaciones en modo paso a paso. Esto funciona igual que el depurador software favorito, excepto que a medida que paso el código, se puede observar el código, el efecto en el diseño de entrada, incluyendo todos los componentes electrónicos externos al microcontrolador [8].

Figura 2.2 Simulador Proteus 7.7

Características del Proteus (Versión 7.7)

Entre estas características podemos mencionar:

- Contexto de interfaz de usuario.
- Bibliotecas modelos
- Esquema de entradas.
- Diseño jerárquico.
- Co-simulación de software del microcontrolador.
- Depuración de código.
- Diagnóstico de mensajería.

2.1.3 PIC C

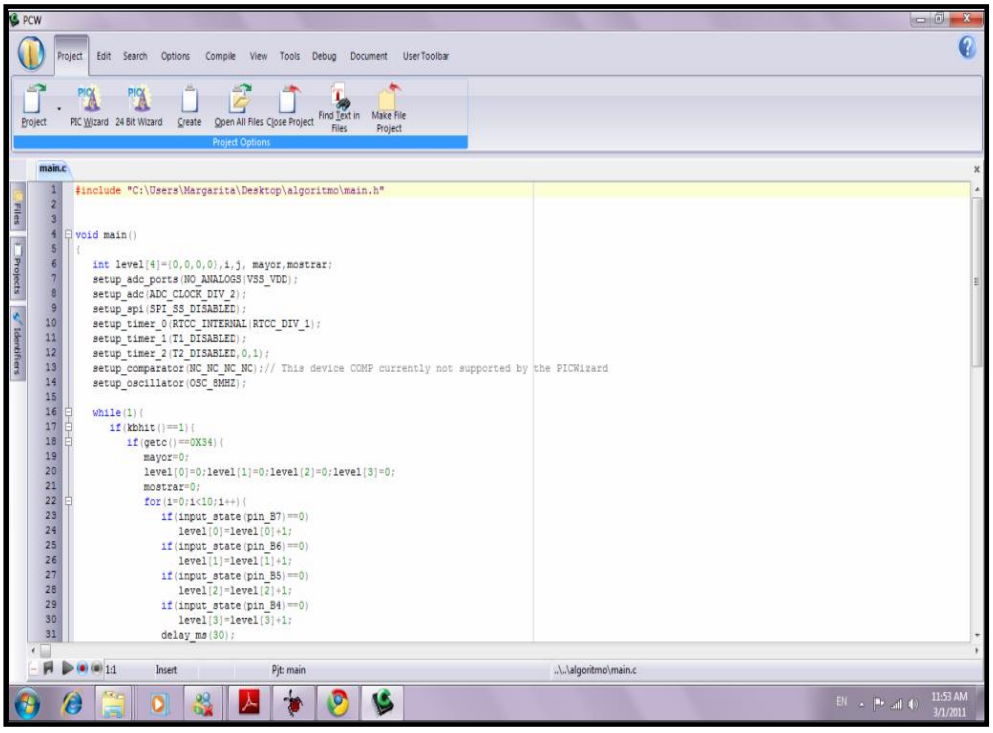

Figura 2.3 Interfaz Gráfica del PIC C

Optimiza compiladores de CCS C contienen operadores estándar C y construido en las bibliotecas de funciones que son específicas a los registros de PIC, proporcionando a los desarrolladores una herramienta poderosa para acceder a las funciones del dispositivo de hardware desde el nivel del lenguaje C. Norma preprocesadores C, los operadores y las declaraciones se pueden combinar con las directivas específicas de hardware y CCS proporciona funciones integradas y las bibliotecas de ejemplo para desarrollar rápidamente aplicaciones que incorpora tecnologías de vanguardia tales como táctil capacitiva, inalámbricas y por cable de comunicación, de movimiento y control de motor y la gestión de la energía, ver figura2.3 [9].

2.2 Herramientas de Hardware para el desarrollo del proyecto

2.2.1 3pi Robot Pololu

El robot Pololu 3pi es un completo y de alto rendimiento de la plataforma móvil con dos motores con engranajes de metal micro, cinco sensores de reflectancia ver figura 2.4, un carácter de 8 x 2 LCD, un timbre y tres botones de usuario, todos conectados a un microcontrolador ATMega328 C-programable. Capaz de alcanzar velocidades superiores a 3 pies por segundo, 3pi es un gran robot primera para los principiantes ambiciosos y un robot perfecto

segundos para aquellos que buscan pasar de los robots para principiantes no programables.

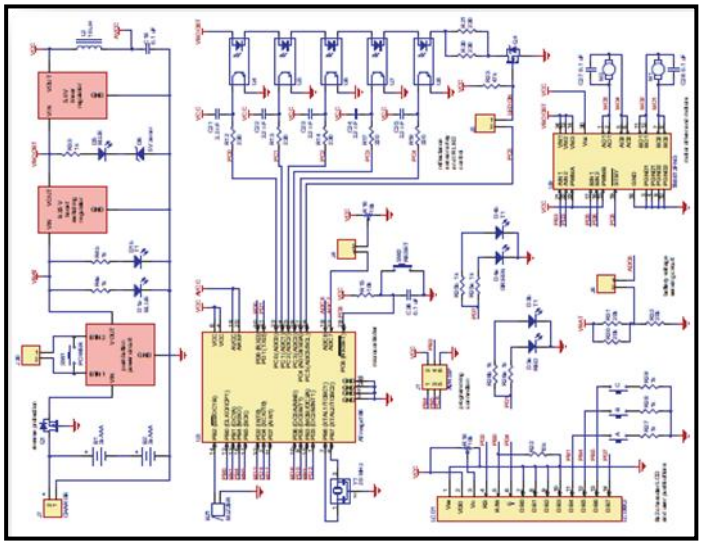

Figura 2.4 Diagrama Esquemático del pololu 3 pi

El robot 3pi está diseñado para sobresalir en la línea siguiente y concursos de resolución de laberintos y es una gran plataforma para las personas con experiencia en programación C para aprender robótica. En su corazón es un microcontrolador Atmel ATMega328 funcionando a 20 MHz con 32 KB de memoria Flash de programa y memoria de datos de 2 KB, suficiente espacio para ejecutar programas complicados. Un extra de 512 bytes de memoria flash persistentes se proporciona en el microcontrolador para el registro de datos o aplicaciones de aprendizaje a largo plazo. Usted puede utilizar AVR Studio o WinAVR para el desarrollo. El 3pi también es compatible con el entorno de desarrollo popular Arduino.

Mover un motor con control de velocidad y dirección, es una característica que tienen los motores, para cambiar de dirección de rotación se debería cambiar la polaridad del voltaje; entonces como no es lógico realizar el cambio de conexión de pilas debemos utilizar el llamado puente H, como veremos en la figura 2.5.

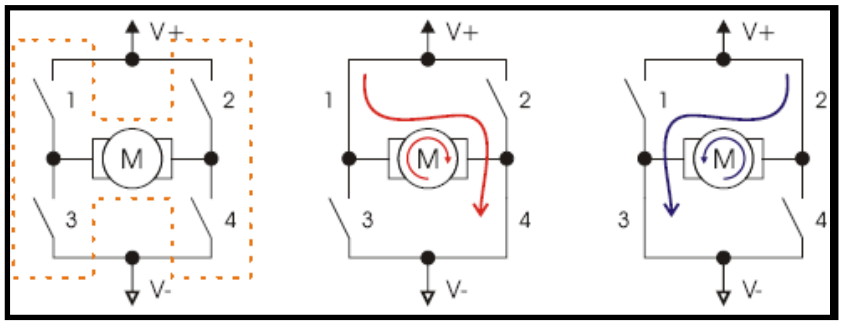

Figura 2.5 Diagrama del motor DC con puente H

Los cuatro puntos de corte de corriente permiten el cambio de sentido. Se usan puentes H para ambos motores en el Pololu 3pi mediante el chip TB6612FNG conectando las salidas de los puertos del microcontrolador correspondientes a los pines PD5 y PD6 para el motor M1 y para el motor M2 se utilizan los pines de control en PD3 y PB3. Podemos ver su funcionamiento en la siguiente tabla 2.6 : [1]

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)

Tabla2.6 Funcionamiento de los motores DC

2.2.2 Microcontrolador ATmega 328P

Este microcontrolador está incorporado en el Pololu 3pi el mismo que será programado para controlar los motores del Pololu 3pi. A continuación se describen las características más relevantes de este dispositivo.

Dentro de su arquitectura interna tenemos; Memoria Flash no volátil de 32 kb, Memoria Sram de 2KB, Memoria EEPROM de 1Kb, posee Port D, C, B, los USART y A/C convertidor.

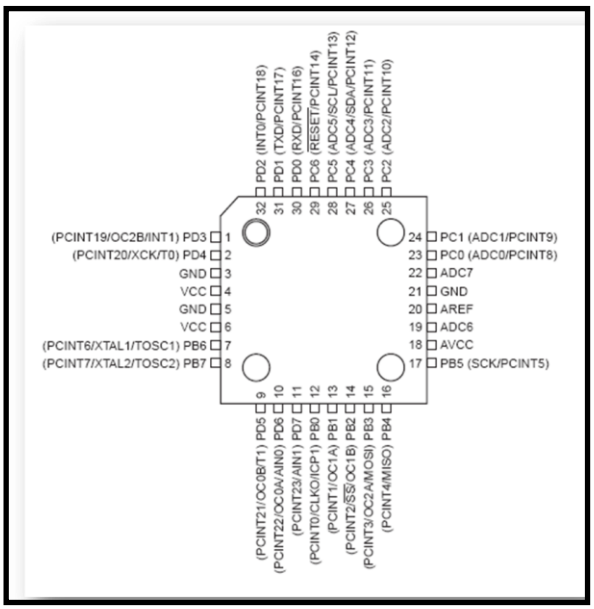

Figura 2.7. PIC ATmega 328P

Características principales del microcontrolador ATmega 328P

Tiene una arquitectura RISC, contiene 32 x 8bits Registros de Propósito General de Trabajo, tiene el ALU la cual le permite realizar las operaciones aritméticas y de comparación, las operaciones se realizan a un ciclo por reloj, alta resistencias no-volátiles segmentos de memoria en lo que respecta al espacio de memoria Flash, esta se divide en dos secciones una de arranque del programa y la otra será la del requerimiento del programa

En el momento de llamar a las interrupciones y subrutinas, la dirección de retorno del contador de programa (PC) es almacenada en la Pila (Stack).Además contiene 64 direcciones de la CPU para funciones periféricas que son espacio de memoria de entrada y salidas I / O.
En la figura 2.7 se muestra el esquema de los pines del ATmega 328P del Pololu 3pi. [10].

2.2.3 Microcontrolador PIC 16F886

Este microcontrolador se encuentra en el Circuito Moda.
En su arquitectura interna consta de una arquitectura RISC, posee 8 niveles de profundidad en la pila, un rango de frecuencia de 8MHz a 31KHz, convertidor A/D con 10 bit de resolución, temporizadores TMR0, TMR1 y TMR2, módulo de PWM, permite RS-485, RS-232 la cual se está utilizando como comunicación en el proyecto, tiene los Puerto A, B y C como lo podemos apreciar en la figura 2.9. [11].

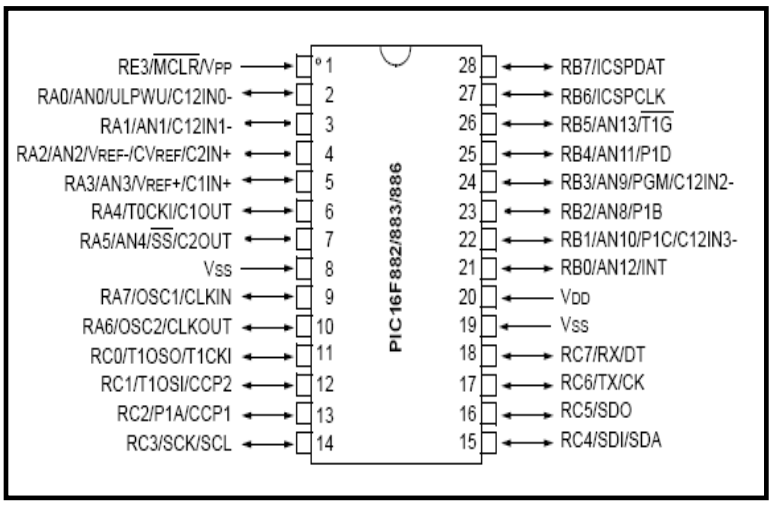

Figura 2.8 PIC 16F886

2.2.4 Pololu USB AVR Programmer

El programador emula un AVRISP v2 en un puerto serie virtual, por lo que es compatible con el software estándar de programación AVR. Tienen dos características adicionales que ayudan a generar y depurar proyectos de aplicación para el monitoreo de señales y los niveles de tensión.

Principio de funcionamiento del Pololu USB AVR Programador.

El programador AVR USB se conecta al puerto USB de su ordenador a través de un cable mini-B y se comunica con el software de programación, tales como AVR Studio o AVRDUDE, a través de un puerto COM virtual utilizando el protocolo AVRISPV2/STK500. El programador se conecta al dispositivo de destino a través de un cable de programación de 6 pines ISP, ver figura 2.10. [2].

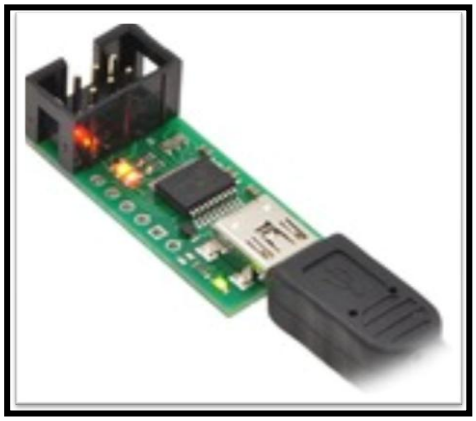

Figura 2.9 Pololu USB AVR Programmer

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL PROYECTO

3.1 Detalles del diseño propuesto

Este capítulo muestra el diseño y la implementación del software y hardware para alcanzar cada uno de los objetivos propuesto. En primera instancia se describirá de una manera muy general el diseño del hardware utilizado para luego mostrar la parte medular del proyecto que es la implementación del software.

La parte del hardware está compuesta por el Robot Pololu 3pi, que consta de varios elementos, incluidos sensores y una pequeña pantalla LCD, para poder trabajar en este dispositivo se hizo uso de la programación y técnicas de control PID discreto. Como elementos adicionales usamos un trasmisor y un receptor para la comunicación inalámbrica entre el Pololu y el mando que tiene el usuario el cual tiene una pantalla de cristal y un Joystick, que nos va a permitir controlar el movimiento que deseamos que realice el Pololu mediante una comunicación analógica.

El Kit AVR Butterfly ver figura 3.1, tiene los siguientes periféricos: un controlador LCD, varias interfaces de comunicación entre ellas: UART, SPI, USI. A su vez tiene diferentes tipos de programación, tiene incluido el convertidor analógico-digital, reloj en tiempo real y modulación por ancho de pulsos (PWM).

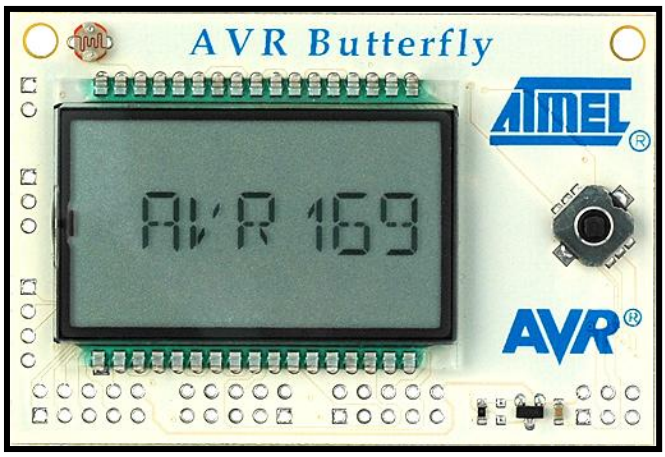

Figura 3.1 Parte Superior del Butterfly

También se ha hecho uso del programador que es de gran ayuda para poder grabar en el Pololu los cambios hechos en el software, además tiene algunas funciones como lo son: convertidor de USB a serial con niveles de voltaje TTL, esto quiere decir que se puede conectar directamente al Pololu, conjuntamente tiene soportes para diferentes sistemas operativos entre ellos tenemos: Windows XP, Windows Vista, Windows 7 y Linux. Este dispositivo también puede actuar como dos canales del osciloscopio severamente limitada por la frecuencia de muestreo que es apenas de 20KHz como se observa en la figura 3.2.

Figura 3.2 Scope del Programador

Otro punto clave en la realización del proyecto es el software a utilizar llamado AVR Studio el cual ya se ha detallado anteriormente, además de la utilización de las librerías del Pololu que son de importante ayuda en este proyecto en el cual vamos a resaltar el uso de dos que nos parecieron más importantes. Entre ellas tenemos la librería del **PID discreto** en el cual como entrada en la planta está la lectura de los 5 sensores reflectivos el cual si nota una variación en la línea recta en la que se dirige tendrá como resultado el cambio en la velocidad de los motores ya sea en el motor 1 o en el motor 2 dependiendo del signo del error y esto depende a la perturbación que se provoque sobre nuestro sistema. Otra librería es la **maze-solve** el cual nos ayudará en la solución del laberinto en la que aplica la teoría de arboles para la resolución de rutas correctas y eliminar rutas erróneas ver figura 3.3.

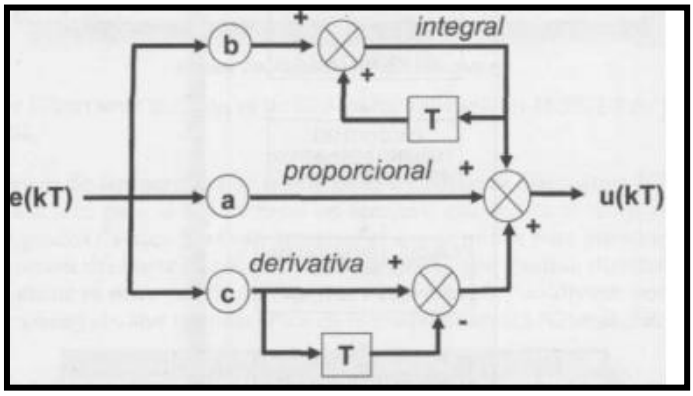

Figura 3.3 Control PID discreto

Para poder hacer uso de las funcionalidades del Robot Pololu 3Pi se tuvo que hacer una instalación extra a parte del AVR Studio y del GCC, estas herramientas fueron facilitadas de la pagina www.pololu.com, con ellas se pudo manejar con mayor rapidez las funcionalidades del robot, a su vez tener un sin número de ejemplos desde encendido y apagado de los leds del robot hasta un control PID para los controles el cual facilita el aprendizaje de las librerías.

3.2 Diseño del Sistema

El diseño del sistema está basado en el siguiente diagrama de bloque, en el que se muestra las señales de entrada del sistema así como las señales de salida. Las señales de entrada están representados por los sensores reflectivos que son 5 en total como también por el AVR Butterfly que mediante los módulos de radiofrecuencia es una entrada

serial que nos sirve para la manipulación del movimiento del Pololu 3Pi.

Las señales de salida serán las que se encuentra en los motores 1 y 2, que permitirán la movilidad del Robot Pololu 3Pi,ver figura3.4.

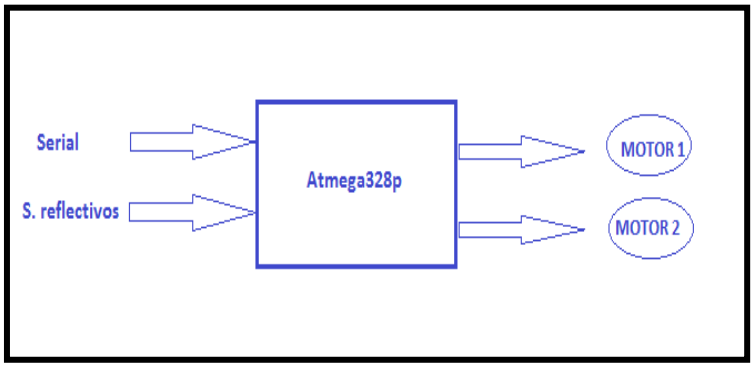

Figura 3.4 Diagrama de bloques del Sistema

3.3 Análisis del programa en AVR Studio

Para el diseño del programa que permitirá la movilidad del Robot Pololu 3Pi se utilizó el programa AVR Studio y las librerías proporcionadas por el Pololu, cargados luego en el robot para realizar el respectivo procesamiento de los datos adquiridos por cada sensor reflectivos del mismo además señales que vienen desde el módulo de radiofrecuencia.

Para este análisis lo hemos divido en dos partes: en la primera se presentará el diseño de la movilidad del robot mediante el algoritmo

guardado en el microcontrolador y la segunda parte mostraremos la movilidad del robot por medio de comunicación inalámbrica.

3.3.1 Programa para la movilidad del Robot Pololu

A continuación se describe en detalle el código que genera el movimiento del Robot Pololu 3pi en el AVR Studio.

Esta es toda la programación desarrollada en el AVR Studio, aquí se detallan todas las librerías usadas, además las líneas de código que nos parece más sobresalientes para mostrar.

```
/*
 * Codigo para el motor
 * Cambiamos la duty del pwm
 */

#include <pololu/3pi.h>

//Dependiendo de la direccion movemos los motores....
void turn(char dir)
{
    switch(dir)
    {
    case 'L':
        // Cambio a la izquierda.
        set_motors(-80,80);
        delay_ms(200);//es para un giro de 90
        break;
    case 'R':
        // Cambio a la derecha.
        set_motors(80,-80);
        delay_ms(200);// es para un giro de 90
        break;
    case 'B':
        // Girar.
        set_motors(80,-80);
        delay_ms(400);//es 400 para el giro de 180 grados
        break;
    case 'S':
        //No hacer ningun movimiento
        break;
    }
}
```

La primera línea del archivo, como cualquier otro archivo C que se escribe para el 3pi, contiene un include que le da acceso a las

funciones de la Biblioteca Pololu AVR. Dentro de la función turn () se utiliza las funciones de biblioteca delay_ms () y set_motors () para realizar giros a la izquierda, derecha o un giro de 180°. Las velocidades del motor y los tiempos de las vueltas son parámetros que deben ser ajustados por el 3pi.

Para acceder a esta función desde otros archivos de C, es necesario un archivo de cabecera que se llama turn.h . El archivo de cabecera sólo contiene una sola línea:

void turn(char dir);

Otra cosa que se debe tener en cuenta que aparte de llamar a la función debemos tener en cuenta al archivo cabecera turn.h y de esta forma podemos utilizar la función turn.

La función **follow_segment(),** hará que el Pololu se dirija en línea recta gracias a los 5 sensores reflectivos que se ubican en la parte inferior del robot. En esta función la entrada de la planta serían los sensores reflectivos.

unsigned int sensors[5];
unsigned int position = read_line(sensors,IR_EMITTERS_ON);

En estas dos lineas de código el primer paso que hacemos es la declaración de las variables que son los sensores y en la siguiente hacemos la respectiva lectura de los sensores reflectivos. Un punto importante que debemos tener en cuenta que la variable position es la más importante ya que está nos permitirá obtener el estado de entrada de la planta.

int proportional = ((int)position) - 2000;

En está linea observamos un valor de 2000 que significa el set point de nuestro controlador PID el cual si nuestra lectura es de 2000, el error resultante tendrá un valor de 0, como conclusión tendríamos que nuestra planta está en el set point deseado y nuestro controlador no hace ningún cambio en la velocidad de los motores 1 y 2. Pero también podemos tener un valor tanto positivo como negativo en nuestra lectura esto causaría una variación de velocidad tanto como e

n

```
// el derivativo en forma discreta q es una resta
// entre la anterior lecctura y la actual
int derivative = proportional - last_proportional;
integral += proportional;

// Guardamos la antigua posicion
last_proportional = proportional;

// el control PID con su repectiva constantes
// Y dependiendo del signo resultante movemos en motor uno y dos
int power_difference = proportional/20 + integral/10000 + derivative*3/2;
```

otor 1 o en el motor 2.

Estás líneas son parte de un controlador PID discreto, el cual permite sacar su parte derivativa e integral del mismo, además de esto encontramos la aplicación ya del microcontrolador con sus respectivos valores del controlador PID y se aplican las constantes proporcional, integral y derivativa.

```
        const int max = 75; // the maximum speed
        if(power_difference > max)
            power_difference = max;
        if(power_difference < -max)
            power_difference = -max;

        if(power_difference < 0)
            set_motors(max+power_difference,max);
        else
            set_motors(max,max-power_difference);

        // We use the inner three sensors (1, 2, and 3) for
        // determining whether there is a line straight ahead, and the
        // sensors 0 and 4 for detecting lines going to the left and
        // right.

        if(sensors[1] < 100 && sensors[2] < 100 && sensors[3] < 100)
        {
            // si es q no se ve ninguna interseccion
            return;
        }
        else if(sensors[0] > 200 || sensors[4] > 200)
        {
            // encontro una interseccion
            return;
        }

    }
}
```

Con estas líneas de código trabajaremos velocidad del motor, esto nos ayuda a que el Pololu no sufra un desborde en su velocidad y que pueda ser que las lecturas fallen de nuestros sensores, así mismo en la variable max deberíamos hacer los respectivos cambios para la velocidad y respuesta de la planta esto debería ir acompañado con los hechos en las constantes del controlador PID porque con el cambio respectivo en estas variables uno puede tener un mejor overshoot y un mejor tiempo de respuesta de la planta.

```
#include <pololu/3pi.h>
#include "follow-segment.h"
#include "turn.h"

//Array q almacena los tipos de giro
char path[100] = "";
unsigned char path_length = 0; // the length of the path

// Mostramos la ruta actual q esta dando en la lcd.
void display_path()
{

    path[path_length] = 0;

    clear();
    print(path);

    if(path_length > 8)
    {
        lcd_goto_xy(0,1);
        print(path+8);
    }
}
```

Observamos que hemos declarado un arreglo tipo char con una dimensión de 100 el cual nos sirve para almacenar una gran cantidad de rutas del laberinto, la variable path_length nos sirve para mostrar en que posición nos encontramos en el arreglo. La función display_path() nos ayuda a visualizar en la LCD los distintos niveles de la lectura de cada sensor esto nos permitirá poner en una buena ubicación al sensor.

```
char select_turn(unsigned char found_left, unsigned char found_straight,
 unsigned char found_right)
{
    // esto retorna la direccion de q debe dar
    if(found_left)
        return 'L';
    else if(found_straight)
        return 'S';
    else if(found_right)
        return 'R';
    else
        return 'B';
}
```

Está función nos retorna el tipo de giro que debemos dar, el cual en siguientes procedimientos este valor será procesado para hacer el respectivo giro del motor.

```
void simplify_path()
{
    // solamente simplifica el arrglo si la segunda es B
    if(path_length < 3 || path[path_length-2] != 'B')
        return;

    int total_angle = 0;
    int i;
    for(i=1;i<=3;i++)
    {
        switch(path[path_length-i])
        {
        case 'R':
            total_angle += 90;
            break;
        case 'L':
            total_angle += 270;
            break;
        case 'B':
            total_angle += 180;
            break;
        }
    }
```

Esta función nos dará como resultado la anulación de todas las rutas erróneas que el robot encuentre y así ir simplificando el camino a su llegada. Como ejemplo tenemos el camino RBR esto nos da como resultado estar en el mismo lugar de partida, el robot tendrá como opción seguir en línea recta, como detalle adicional detallamos lo que significa cada letra: R es girar derecha; L girar a la izquierda; B nos

dice que el robot dará un giro de 180 grados, esto quiere decir que no hay camino.

```
// muestra la solucion
while(1)
{
     // Apagamos los motores.
     set_motors(0,0);
     play(">>a32");

     // esperamos q presione la tecla b pa mostrar la solucion
     while(!button_is_pressed(BUTTON_B))
     {
          if(get_ms() % 2000 < 1000)
          {
               clear();
               print("Solved!");
               lcd_goto_xy(0,1);
               print("Press B");
          }
          else
               display_path();
          delay_ms(30);
     }
     while(button_is_pressed(BUTTON_B));

     delay_ms(1000);
```

Estas líneas de código en primer lugar nos muestran en el LCD que debemos presionar el botón B para presentar la solución que está grabado en el arreglo.

```
int i;
for(i=0;i<path_length;i++)
{
     //PID
     follow_segment();

     // velocidad q va
     set_motors(50,50);
     delay_ms(50);
     set_motors(40,40);
     delay_ms(200);

     turn(path[i]);
}
follow_segment();
```

En este for recorremos desde 0 que es la posición inicial del arreglo hasta path_length que es la última posición, también aplicamos técnicas de control PID para mantenernos siempre en la línea negra y

que los sensores siempre tengan una buena lectura, al momento que salimos del follow_segment es porque encontramos una intersección entonces seteamos cierta velocidad en los motores y hacemos el respectivo cambio con turn, si nos fijamos dentro de turn ingresamos el tipo de giro que almacenamos en la variable path.

3.3.2 Programa para la movilidad del Robot usando el Butterfly

Para llegar a estas instancias hemos tomado como base el código fuente proporcionado por la página www.avr.com que fue de gran ayuda en la utilización de las librerías. El código fuente proporcionado es el que viene cargado de fábrica en el Butterfly y con unas pequeñas mejoras se ha logrado codificar con éxito el programa de esta tesis, a continuación se detallará las líneas de código que se adjuntaron para llegar al objetivo de este proyecto.

```
*****************************************************************************/
void Initialization(void)
{
    char tst;           // dummy

    OSCCAL_calibration();       // Calibración del Registro OSCAL

    CLKPR = (1<<CLKPCE);        // Setea el Clock Preescalador

    // Setea el Preescalor para convertir los 8Mhz en 1Mhz
    CLKPR = (1<<CLKPS1) | (1<<CLKPS0);

    // Deshabilita el comparador analógico
    ACSR = (1<<ACD);

    // Deshabilita la entrada digital PF0-2 (power save)
    DIDR0 = (7<<ADC0D);

    // Habilita las pullup en el puerto B
    PORTB = (15<<PB0);        //
    // Habilita las pullup en el puerto E
    PORTE = (15<<PE4);

    sbiBF(DDRB, 5);               // setea OC1A como salida
    sbiBF(PORTB, 5);              // setea OC1A con un nivel alto de voltaje

    Button_Init();              // Inicializa los cambias de pines para el joystick


    USART_Init(51);             // Baud rate = 2400bps


    DF_CS_inactive;             // Deshabilita DataFlash

    LCD_Init();                 // Inicializa la LCD
}
```

Esta función de inicialización es muy importante para la configuración de los registros del Butterfly (Atmega169), en las primeras líneas de código encontramos la función OSCAL_calibration que nos permite calibrar el registro OSCAL, también tenemos la configuración de los registros para poder llevar el oscilador de 8Mhz a 1Mhz esto es necesario para la comunicación serial ya que la configuración de los Baudios la vamos hacer en base a 1Mhz.

Para la respectiva configuración de los Baudios utilizamos la siguiente tabla3.5:

Baud Rate (bps)	f_{osc} = 1.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%
4800	12	0.2%	25	0.2%
9600	6	-7.0%	12	0.2%
14.4k	3	8.5%	8	-3.5%
19.2k	2	8.5%	6	-7.0%
28.8k	1	8.5%	3	8.5%
38.4k	1	-18.6%	2	8.5%
57.6k	0	8.5%	1	8.5%
76.8k	–	–	1	-18.6%
115.2k	–	–	0	8.5%
230.4k	–	–	–	–
250k	–	–	–	–
Max. (1)	62.5 kbps		125 kbps	

1. UBRR = 0, Error = 0.0%

Figura 3.5 Tabla de Configuración del Puerto Serial

La tasa de máxima de los módulos ASK es de 4800 Baudios hemos escogido la tasa de trabajo de 2400 Baudios por la razón que los módulos ASK tienden a trabajar en mejores condiciones si escogemos la menor tasa de trabajo ya que con la máxima están propensos al error.

Con la tasa de trabajo de 2400 Baudios a 1 Mhz el error es de 0.2% propios del microcontrolador, para configurar este valor tenemos que agregarlo a un registro de 16 bits el cual lo separaremos en un nivel

alto y uno bajo. El nivel alto tendrá que llevar el valor de 51 para el nivel bajo tendremos el 25, el registro a utilizar es el UBRR.

```
void USART_Init(unsigned int baudrate)
{
   // Set baud rate
   UBRRH = (unsigned char)(baudrate>>8);
   UBRRL = (unsigned char)baudrate;

   // Enable 2x speed
   UCSRA = (1<<U2X);

   // Enable receiver. Transmitter stays disabled since
   // no data gets send in this application (only receive in vcard).
   UCSRB = (1<<RXEN)|(1<<TXEN)|(0<<RXCIE)|(0<<UDRIE);

   // Async. mode, 8N1
   UCSRC = (0<<UMSEL)|(0<<UPM0)|(0<<USBS)|(3<<UCSZ0)|(0<<UCPOL);
}
```

Como se puede apreciar en la función hacemos el respectivo desplazamiento del valor de 51 para convertirlo en 25, otra parte muy importante es TXEN que lo seteamos con uno para decir que vamos a trasmitir datos. Este código lo podemos verificar en el propio datasheet del microcontralor que se lo puede encontrar la configuración en lenguaje assembler si es que no estamos trabajando en lenguaje C.

```
Initialization();
//sei(); // mt __enable_interrupt();

for (;;)                    // Main loop
{
    if( !(PINB & (1<<PINB7)) ) {
        LCD_puts_f("PINB7", 1);
        Usart_Tx(dato1);
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');
        Usart_Tx('a');

        Delay(20);
    }
```

Este fragmento de código es una demostración de cómo se trasmiten los datos vía serial y es procesado cada uno de los botones del Joystick, como se puede apreciar en este código el dato es enviado ocho veces por la razón que en los primeros datos receptados tenemos una tasa de error del 70%, mientras que los últimos datos llegan correctamente, esto se debe a la influencia de ruido en el ambiente pues estos módulos utilizan una modulación basada en la a

```
if(button_is_pressed(BUTTON_C)){
    clear();
    print("Serial");
    char buffer1[2]="  ";
    serial_set_baud_rate(2400);
    while(1){
        buffer1[0]=0;
        buffer1[1]=0;

        while(serial_receive_blocking(buffer1,2, 1000));

        clear();
```

ud de la onda.

En esta parte del código esperamos que se presione la opción C para poner a funcionar la recepción de datos de manera serial el cual solo utilizaremos el puerto de recepción, como nos daremos cuenta se creó una variable llamada Buffer1 el cual es un arreglo de char que es muy necesario en la recepción de datos seriales, después se hace la

respectiva configuración de Baudios del Pololu o velocidad de transferencia, luego de estos pasos ingresamos en un while true el cual siempre estará inicializando la variable buffer1 y recepción de datos y a la vez almacenamos en la variable buffer1.

```
if(buffer1[0]==buffer1[1]){

                    clear();
          lcd_goto_xy(0, 1);
          switch(buffer1[1])
                    {
                    case 'b':
                         print("Frente");
                         set_motors(40,40);
                         break;
                    case 'a':
                         print("Retro ");
                         set_motors(-40,-40);
                         break;
                    case 'e':
                         print("Derec ");
                         set_motors(30,-30);
                         break;
                    case 'd':
                         print("Izqui ");
                         set_motors(-30,30);;
                         break;
                    case 'c':
                         print("Stop  ");
                         set_motors(0,0);
                         break;
                    default:
                         print("Nada  ");
```

Aquí en esta línea se hace la comparación de los datos guardados en la variable buffer1 si estas variables son iguales procesamos el dato en un switch y verificamos si este cumple algunos de los casos especificados en esta función y con respecto a esto hacemos el respectivo movimiento de los motores.

CAPÍTULO 4

4 SIMULACION Y PRUEBAS EXPERIMENTALES.

4.1 Implementación y pruebas del Proyecto

El diseño básicamente de nuestro Robot Pololu 3pi cuenta con todos los sensores proporcionados del mismo para la realización de nuestro proyecto los cuales se encuentran en la posición adecuada para hacer la correcta lectura del color programado de las líneas de nuestro laberinto.

En la implementación del código del robot, el modo de comunicación para las pruebas se las realizo por medio USB para el ingreso del firmware en el microcontrolador, en la pruebas se utilizo este hardware para probar la comunicación RS232 antes de aplicarlas con los módulos de RF, ver figura 4.1.

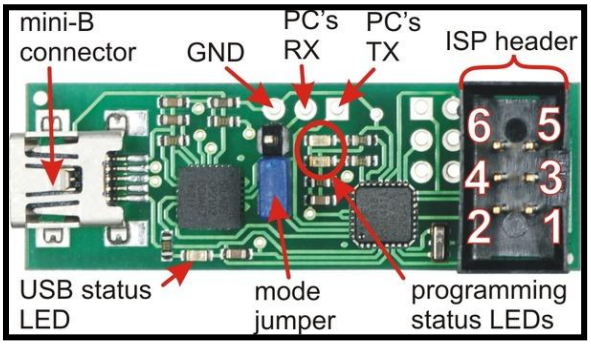

Figura 4.1 Partes del Programador

Después de cargar el programa en el microcontrolador se chequeo la respectiva conexión de PD0 y de PD1 para el correcto funcionamiento del mismo.

En la parte de la adquisición de datos se tomo muy en cuenta la frecuencia a trabajar ya que en la primeras pruebas con los módulos de RF se obtuvo errores por la modulación ASK debido a que la tasa de error es muy alta, ver figura 4.2.

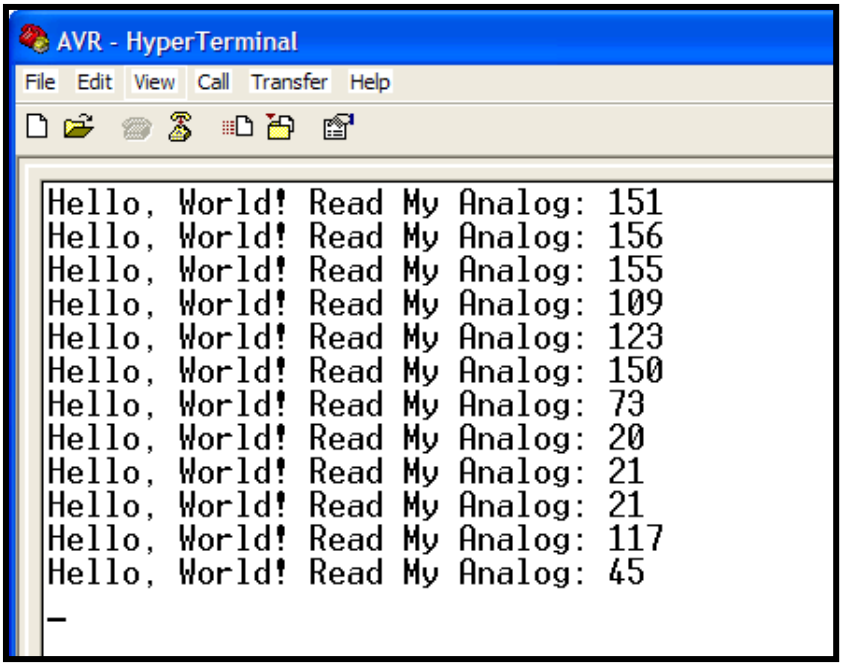

Figura 4.2 Adquisición de datos por hyper Terminal

En la pantalla lcd se pudo apreciar los errores a la hora de hacer las pruebas en el laberinto cuando hubo un cambio de velocidad en el controlador PID, pues se mostraba las rutas almacenadas en el arreglo y en el momento de ser eliminadas algunas rutas erróneas este procedimiento no se realizaba.

El simulador de proteus fue de ayuda a la hora de probar el lenguaje de programación y configuración del microcontrolador, ver figura 4.3.

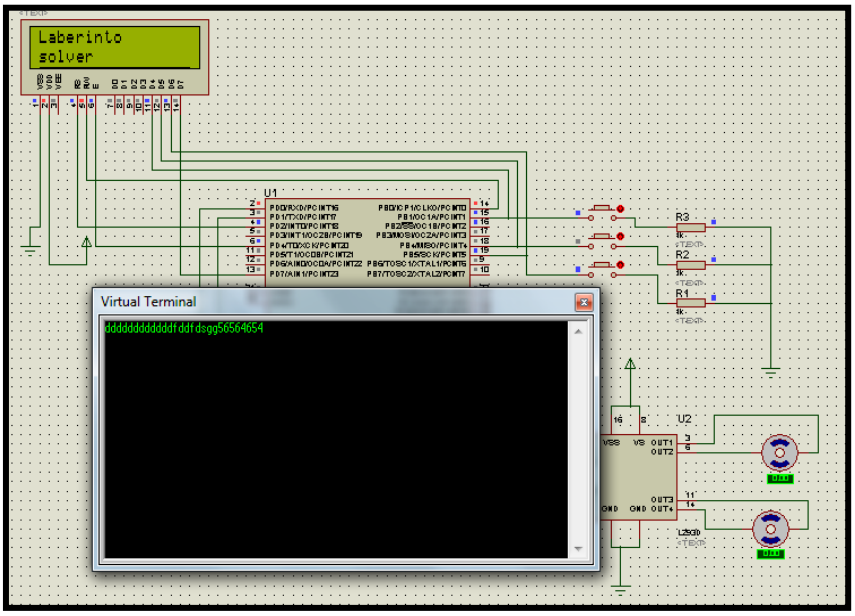

Figura 4.3 Prueba en Proteus

Para probar los módulos de RF se tuvo que usar los puertos de la computadora, para el cual usamos el programador del Pololu el que crea un puerto vitual en la misma. Se pudo apreciar los errores en la comunicación, en el que se tuvo que hacer comparaciones de datos para la correcta lectura de los datos enviados por el modulo receptor de RF ver figura 4.4.

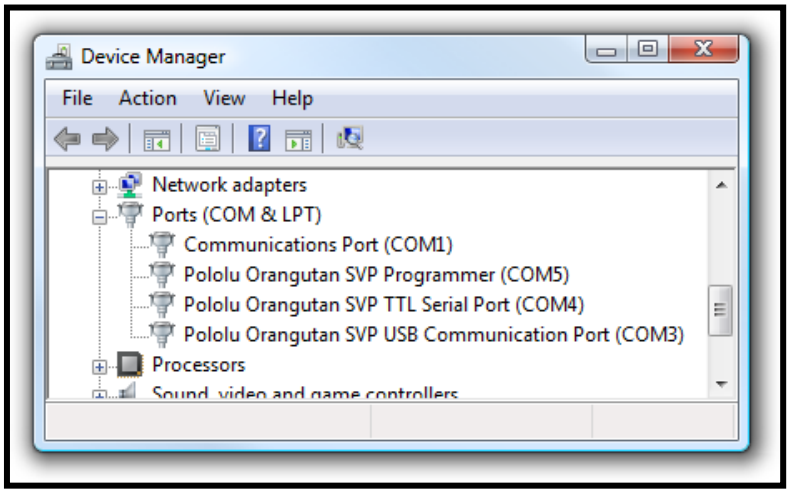

Figura 4.4 Administrador de Periféricos

CONCLUSIONES

1) Se obtuvo mejor respuesta en la resolución del laberinto gracias al manejo de las técnicas de control PID, porque esta técnica de control fue utilizada para que el robot Pololu no se desvíe de su ruta y siempre se mantenga en línea recta y las diferentes perturbaciones que se encuentren no afecten en la resolución del laberinto.

2) Después de varias pruebas realizadas se puede concluir que para la comunicación inalámbrica donde se usaron módulos de ASK tuvimos que trabajar con la tasa de transferencia más baja que nos provocará menos errores en el envío de datos ya que esta modulación empleada es muy propensa a tener fallas por el ruido ambiental por lo que está basada en la amplitud de la onda.

3) Se puede concluir que en el desarrollo de este proyecto nos dimos cuenta que la herramienta utilizada para la programación fue nuestra mejor opción, el AVR Studio tiene las herramientas necesarias para poder trabajar con gran facilidad el algoritmo para la resolución de laberintos

4) El Programa PROTEUS nos permite una excelente visualización del comportamiento del PIC programado dentro del circuito utilizado; a través de la simulación que se puede realizar en esta

herramienta muy útil, por sus múltiples funciones que por medio de las cuales realizamos un exhaustivo análisis del circuito implementado.

RECOMENDACIONES

1) Verificar si las baterías están a su máxima carga ya que ni no se tiene por lo menos el 50% de la carga el Pololu se apagará automáticamente el otro problema que se presentaría es que los sensores no detectan la misma cantidad de luz reflejada y los motores no se mueven con la misma velocidad.

2) Al usar el Pololu en la resolución del laberinto tener un ambiente claro donde la oscuridad no vaya a afectar el desarrollo del mismo, ya que en un lugar muy oscuro los sensores reflectivos no brindan una mayor eficacia.

3) La velocidad tiene una relación directa con la longitud de las líneas del laberinto entonces si deseamos elevar la velocidad por ende tendríamos que incrementar la longitud debido a que si el robot tiene mayor rapidez y su longitud es demasiado corta su respuesta a los giros estará más propensa a los errores.

4) Manejar de manera adecuada en la programación los sensores para que esté pueda detectar sin ningún problema el color negro y blanco para que a la hora de escoger cualquier ruta no se atasque y lea el correcto camino para que no se quede dando vueltas en un mismo lugar.

5) Cada vez que hagamos pruebas con el robot siempre debemos recordar que las llantas recogen el polvo y suciedad en el curso, esto traerá como consecuencia problemas en las curvas y en su rapidez, y cuando empiece un aumento significativo en su velocidad el rendimiento se hace dependiente de la tracción de los neumáticos. Para solucionar este inconveniente debemos frotar las llantas con un poco de alcohol y una toalla de papel.

6) Para mejorar la comunicación inalámbrica es mejor usar modulación FSK pues está no se ve afectada por el ruido ambiental por la razón de que está depende de la frecuencia y no de la amplitud como se aplica en la modulación ASK.

ANEXO A

Pruebas del Hardware del programador del Robot Pololu 3Pi y uso del programa SIOW para la comunicación serial.

En la parte de hardware que fue necesaria para este proyecto fue la utilización del USB serial que tiene el programador Orangután y fue necesaria en las pruebas de comunicación ASK, el cual en nuestra computadora fue el puerto COM2 ver figura A1.

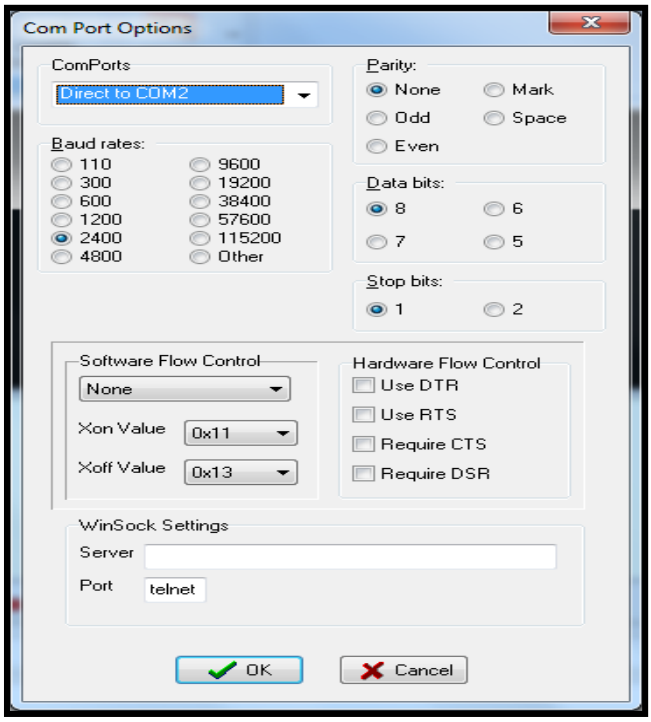

Figura A1: Elección de puerto

Para poder saber si los datos estaban siendo trasmitidos y receptados en el programador hicimos un corto en el Tx y Rx del programador y enviamos un dato x esperando que la prueba resulte y de cómo resultado una recepción correcta del dato enviado. Después se hizo

las respectivas pruebas con el trasmisor y receptor ASK para saber la tasa de error que se tenía al enviar y recibir el dato, ver figura A2 .

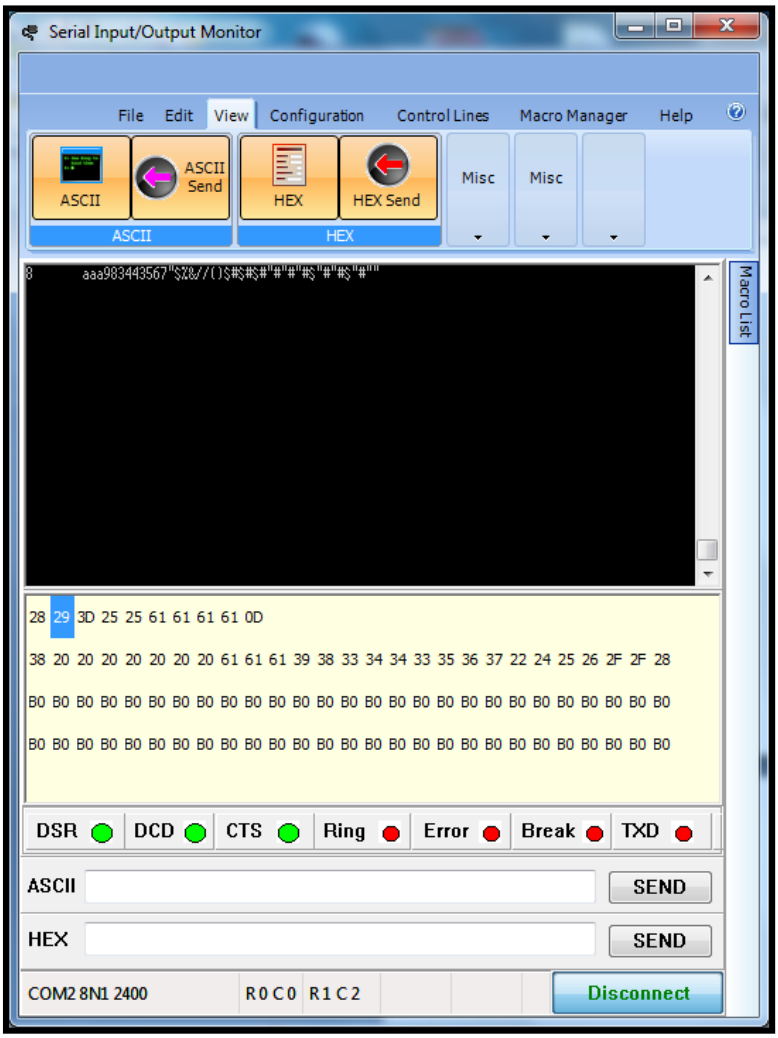

Figura A2: Tasa de errores

Para la conexión del Butterfly fue necesario sólo un conector serial para que pueda ser conectado a la computadora, el cual usamos

puerto serial y el bootloader del Butterfly, no fue necesario el uso del programador Orangután, ver figura A3.

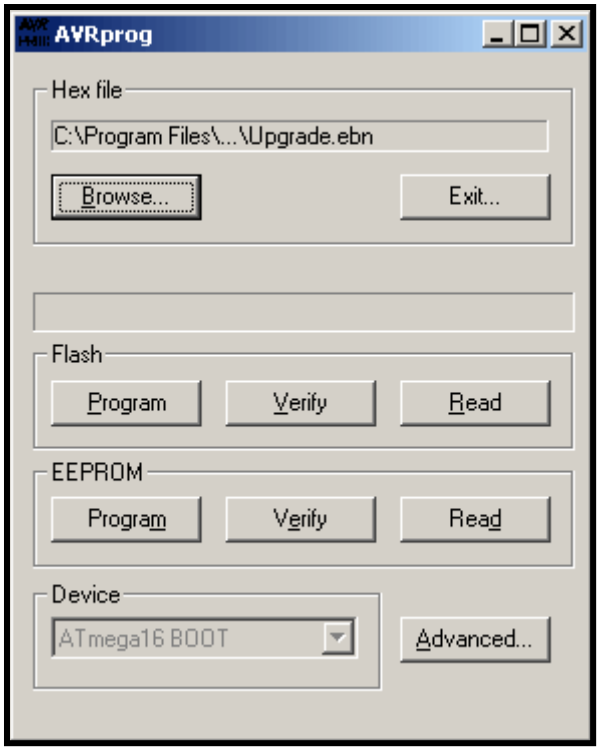

Figura A3: AVRprog

En esta parte se uso el programa AVRprog para hacer uso del bootloader del butterfly, la razón de todo esto es que esta herramienta es hecha para aplicaciones con fines didácticos y es por esta razón que se encuentra el bootloader, el cual reserva un espacio de memoria para hacer la programación que se ve a cargar en los respectivos registros.

Cuando va a usar el bootloader del Butterfly uno debe tener en cuenta que se debe tener presionado el botón central del joystick para que el hardware sea detectado por el el programa AVRprog, una vez que es detectado el el butterfly se abre una programa, buscamos el archivo HEX dando click en browse y al final damos click en program.

Una vez cargado el programa debemos quitar la pila para que corra el programa cargado en el butterfly.

B I B L I O G R A F I A

[1]. Pololu Robotics & Electronics , Página HTML,
http://www.pololu.com

[2]. Compilador AVRstudio
http://www.atmel.avrstudio.com/

[3]. Conceptos de PID discreto
http://www.dia.uned.es/~fmorilla/MaterialDidactico/Aspectos%20practicos.pdf

[4]. Proyectos con el butterfly
http://gandalf.arubi.uni-kl.de/avr_projects/

[5]. Detalle de resolución de laberintos
http://www.pololu.com/docs/0J21/8.a

[6]. Características del Butterfly
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3146

[7]. Detalles de los módulos RF
http://www.ucontrol.com.ar/wiki/index.php?title=Comunicaci%C3%B3n_inal%C3%A1mbrica_entre_PICs

[8]. Compilador Proteus
http://cieerch.blogspot.com/2010/08/manual-compilador-ccs-pic.html

[9]. Pic C
http://electronicapic.iespana.es/manual/programacion_pic_con_c.pdf

[10]. Atmega 328p

http://www.atmel.com/dyn/products/product_card.asp?part_id=328p

[11]. Atmega Pic 16f886

http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026562