**CAPITULO 1**

**DESCRIPCION GENERAL DEL SISTEMA**

**1 Descripción del proyecto**

En esta tesis se aborda el problema del control de robots mediante la utilización de un joystick en interfaz de radiofrecuencia y se presenta la técnica usada normalmente para tratar este problema.

El desarrollo se realiza para un robot conocido en el ámbito académico, como es el Pololu 3pi de Atmel acompañado del kit AVR Butterfly. En este caso, se aprovechan las facilidades que existen para el control de dicho robot, de forma que se llega a un modelo capaz de moverse a lo largo y ancho de la pista controlando su dirección de movimiento por medio de un joystick por radiofrecuencia.

### 1.1 Estrategia Implementada

Como estrategia implementada en este proyecto los elementos que utilizaremos son el robot pololu 3pi por ser una plataforma completa de alto rendimiento móvil, además por tener algunos componentes interactuando entre ellos con los que se puede trabajar, tales como sensores, LCD, timbre, botones de usuario, todos estos conectados a un microcontrolador ATMega328 C-programable. Capable of speeds exceeding 3 feet per second, 3pi is a great first robot for ambitious beginners and a perfect second robot for those looking to move up from non-programmable or slower beginner robots. Este robot es capaz de alcanzar velocidades superiores a 3 pies por Segundo.

Otro elemento utilizado es el KIT AVR Butterfly que es una tarjeta que consta con un microcontrolador Atmega169PV. Este KIT está formado por algunos componentes los más destacados son una LCD con 120 segmentos, un Four-direction joystick with centre push for user input Joystick de cuatro direcciones, y un termistor NTC para medir temperatura del cual utilizaremos el joystick para controlar el movimiento del robot pololu a través de radiofrecuencia.

**1.2 Alcance y limitaciones del proyecto**

Mediante pruebas realizadas a nuestro proyecto hemos encontrado que tiene sus alcances y limitaciones entre las cuales podemos denotar que:

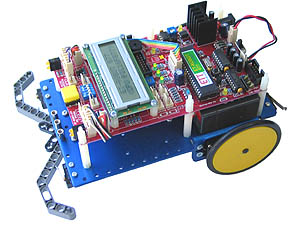
**1.2.1 ALCANCES**

* Su sistema único de potencia le permite moverse a velocidades de hasta 100 cm/segundo.
* Es de tamaño compacto mide 95 mm de diámetro y pesa 83 g sin baterías el cual facilita su funcionalidad.
* El robot viene completamente ensamblado es decir no hay necesidad de adaptar las ruedas, los motores, los sensores, etc.
* Es de fácil codificación ya que trabaja con uno de los más usados lenguajes de programación que es el “Lenguaje C”.

**1.2.2 LIMITACIONES**

* Es muy sensible en su manipulación ya que podemos inducirle estática y podemos dañarlo en su totalidad.
* Contiene elementos muy delicados que con cualquier golpe pueden ser averiados parcial o completamente.

**1.3 Proyectos Similares**

**1.3.1 Robot Futurlec 877**

**Figura 1-1: Robot Futurlec 877**

El nuevo robot Futurlec 877 es un robot emocionante, fácil de usar y totalmente programable. Based on the very popular PIC16F877 microcontroller from Microchip, this robot can be programmed to perform a number of instructions. Basado en el popular microcontrolador PIC16F877 de Microchip, este robot puede ser programado para realizar una serie de instrucciones. A series of microswitches are used for collision detection, together with a couple of dc motors to control robot movement. Una serie de micros interruptores se utilizan para la detección de colisiones, junto con un par de motores de corriente continua para controlar el movimiento del robot. A row of optical sensors are mounted on the base for tracking operations. Una fila de sensores ópticos están montados en la base para el seguimiento de las operaciones. Together with a ready to run LCD installed on the top for feedback and information display. Junto con una lista para funcionar LCD instalado en la parte superior para mostrar información y la información.

Se lo puede encontrar en tiendas electrónicas en USA a un costo de $129.

**1.3.2 Boe-Bot Robot Kit - Versión USB**

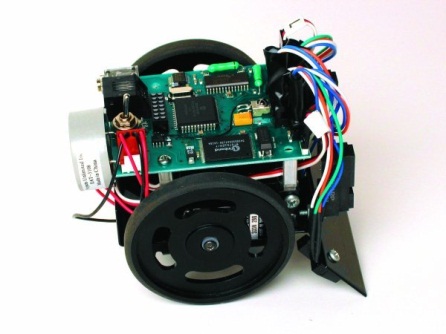


**Figura 1-2: Robot Boe-Bot Kit**

El robot Boe-Bot se basa en un chasis de aluminio de alta calidad de cepillado que proporciona una plataforma robusta para los servomotores de rotación continua y USB BASIC Stamp la Junta de Educación. Many mounting holes and slots may be used to add custom robotic equipment or off-the-shelf Parallax add-ons. Muchos agujeros y las ranuras de montaje se pueden utilizar para agregar equipos a medida robótica o fuera de la plataforma-Parallax complementos. The rear wheel is a slider ball held in place with a cotter pin. La rueda trasera es una bola deslizante en su lugar con un pasador. Drive wheels are molded to fit precisely on the servo spline and held in place with a small screw. Tracción en las ruedas están moldeadas para encajar precisamente en la tira servo y sujeta con un tornillo pequeño.

What really makes the Boe-Bot unique is the BASIC Stamp microcontroller's flexibility of programming when coupled with breadboard circuit construction.Lo que realmente hace que el Boe-Bot única es la flexibilidad del microcontrolador BASIC Stamp de programación cuando se combina con la construcción protoboard circuito.

Se lo puede encontrar en tiendas electrónicas en USA a un costo de $159.99.

**1.3.3 Kit de Robótica de la CAC**

**Figura 1-3: Kit de Robótica de la CAC**

El Kit de desarrollo de la robótica ofrece una introducción al mundo de los robots para los principiantes y aficionados avanzados robots. The development kit includes the powerful PCW Integrated Development Environment with compiler support for Microchip's PIC ® PIC10, PIC12 and PIC16 families and an ICD-U64 in-circuit programmer/debugger that supports C aware real time debugging. El kit de desarrollo incluye el poderoso PCW entorno de desarrollo integrado con el apoyo del compilador para los microcontroladores PIC ® PIC10, Pic12 y las familias PIC16 y un circuito programador-U64 en la CIE / depurador que admita C consciente de depuración en tiempo real. The prototyping board has a variety of devices for every robot to be able to interact with its environment. La placa de prototipo tiene una variedad de dispositivos para cada robot para poder interactuar con su entorno. The devices allow the robot to see, sense magnetic fields, speak, accept external commands, and move; making it fit for robot sumo competition. Los dispositivos permiten al robot para ver, los campos magnéticos sentido, hablar, aceptar comandos externos, y se mueven, por lo que es apto para la competencia del robot de sumo.

La brújula electrónica y un convertidor de texto a voz son únicos para el Kit de Robótica de la CAC. The compass allows the robot to move freely and still know its heading and location. La brújula permite que el robot se mueva libremente y aun así saber de su partida y la ubicación. The text-to-speech converter provides a more personal way to interact with people. El convertidor de texto a voz ofrece una forma más personal para interactuar con la gente.

Se lo puede encontrar en tiendas electrónicas en USA a un costo de $209.

Following along in [Robotics with the Boe-Bot ,](http://translate.googleusercontent.com/translate_c?hl=en&ie=UTF-8&sl=en&tl=es&u=http://www.parallax.com/tabid/134/List/1/ProductID/303/Default.aspx&rurl=translate.google.com.ec&usg=ALkJrhj6Mb0ULHI3bCq2X5Rf1ZgE0aYi9w) users quickly learn about embedded projects, from wiring and components to programming and mechanical dependencies.

**CAPITULO 2**

**FUNDAMENTACION TEORICA**

**2 Descripción**

En el presente capítulo se exponen las herramientas de Hardware y Software que se usan para el correcto funcionamiento de nuestro proyecto, entre las cuales mostraremos las características más importantes como son:

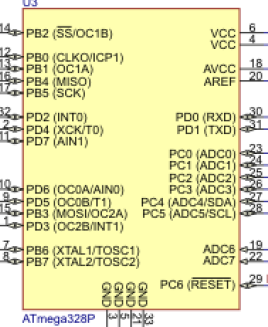
[](http://www.electan.com/catalog/images/3pi.jpg)El microcontrolador Atmega 328P, Micro Gearmotors, la comunicación por medio de la tarjeta AVR Butterfly, la alimentación de nuestro robot, el software AVR Studio versión 4.18 y nuestro simulador Proteus versión 7.7

**Figura 2-1: Pololu Vista Superior**

**2.1 Herramientas de Hardware**

**2.1.1 Microcontrolador**

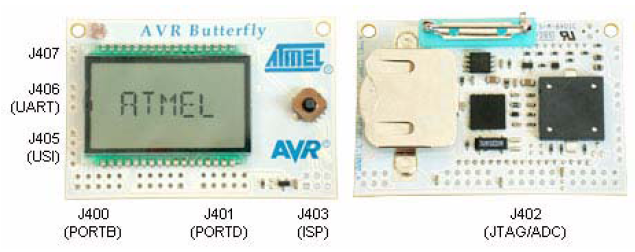
El microcontrolador que usa este robot Pololu 3pi es de marca ATMEL ATmega 328P que corre a 20 MHz de frecuencia, funciona con una memoria Flash de 32 KB, 2KB de memoria en RAM y 1 KB de memoria EEPROM.



**Figura 2-2: Microcontrolador Atmega 328P**

**2.1.2 Comunicación**

La comunicación que utiliza nuestro proyecto es mediante interfaz inalámbrica por radiofrecuencia en la cual interviene nuestro kit AVR Butterfly que contiene el joystick con el que controlaremos los movimientos de nuestro robot pololu 3pi.

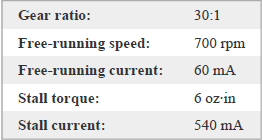


**Figura 2-3: Tarjeta Butterfly Vista Superior e Inferior**

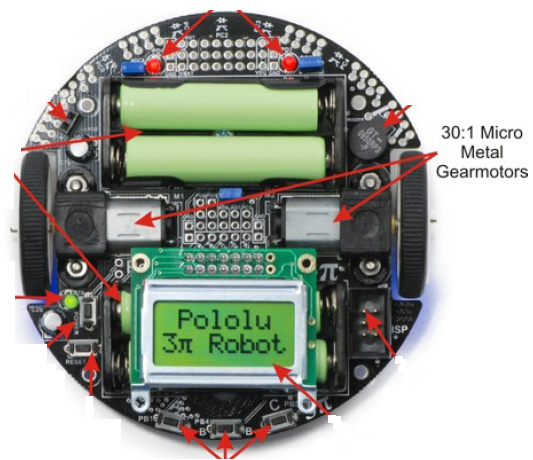
**2.1.3 Motores**

El robot pololu está totalmente ensamblado con dos pequeños motores de metal para las ruedas los cuales tienen excelentes características para nuestro robot entre las cuales tenemos que:





**Tabla 2-1: Características del motor Figura 2-4: Motor del pololu**



**Figura 2-5: Localización de los motores**

**2.1.4 Alimentación**

La alimentación utilizada para nuestro robot consta de 4 pilas AAA que nos da un voltaje aproximado de 6 voltios DC que además son pequeñas, es decir no necesitamos de altos voltajes continuos que nos proporcionarían una gran cantidad de pilas ni usamos las pilas doble AA que son más pesadas.

**Figura 2-6: Pilas Alcalinas AAA**

**2.2 Herramientas de Software**

**Figura 2-7: Logotipo del AVR Studio**

**2.2.1 Introducción**

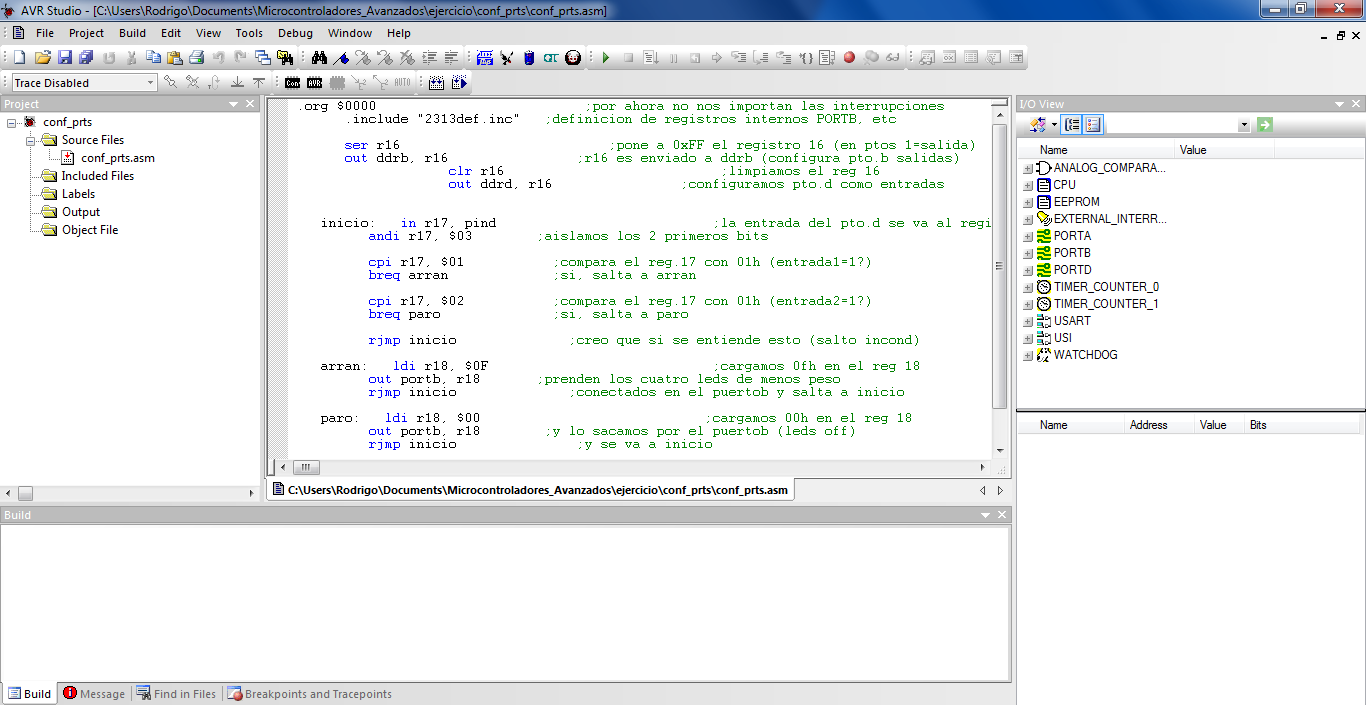
El Software AVR Studio es un entorno de desarrollo integrado para la escritura y depuración que tiene aplicaciones en Windows 98, XP, Me, 2000, NT, 7.

AVR Studio proporciona una herramienta de gestión del proyecto, archivo de origen, editor y simulador de chip. También interactúa con Emuladores de circuitos.

Este software simplifica el desarrollo de tareas, permite a los usuarios reducir significativamente el tiempo de trabajo y genera un mejor ambiente para la realización de los proyectos.

Entre las características más notables tenemos:

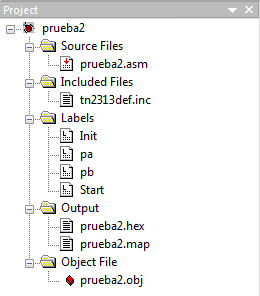
* Entorno de desarrollo integrado para escribir, compilar y depurar.
* Reproducción de memorias configurables incluyendo SRAM, EEPROM, Flash, Registros y E/S.
* Completamente simbólico a nivel de depuración.
* Número ilimitado de puntos de separación (breakpoints).
* Ayuda HTML en línea.
* Amplio programa para opciones de control de flujo.
* Simulación de actividad del puerto y estímulos de entrada al pin.
* Apoyo para C, Pascal y leguaje ensamblador.

****

**Figura 2-8: Entorno de AVR Studio**

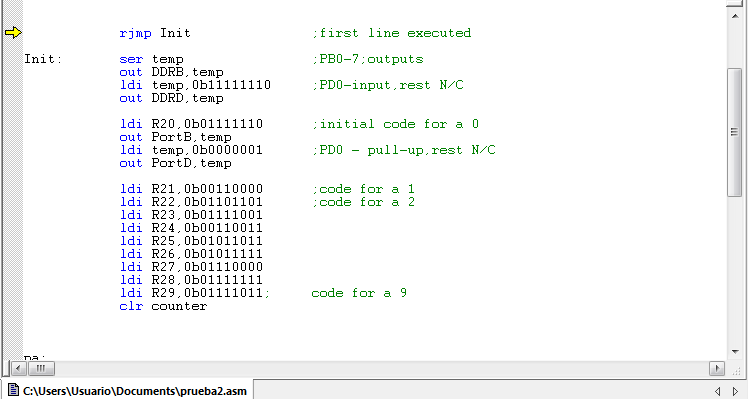
**2.2.1.1 Ventana del Proyecto**

Esta ventana nos muestra el contenido de nuestro proyecto entre ellas las librerías que utilizamos, los lazos, etiquetas que están en nuestro código, el nombre de nuestro archivo fuente, el código .hex y el .map y por último el archivo objeto .obj.



**Figura 2-9: Ventana de Proyecto**

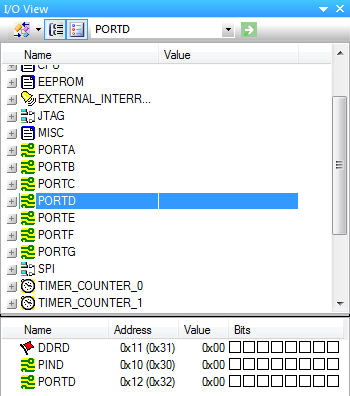
**2.2.1.2 Ventana del código**

Esta ventana es donde escribiremos nuestro código donde incluiremos todo lo que necesitemos las librerías, retardos, etiquetas, lazos que usaremos, etc.

**Figura 2-10: Ventana del Código**

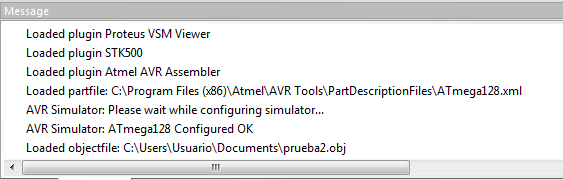
**2.2.1.3 Ventana de Entradas y Salidas**

Esta ventana nos muestra todos los puertos de entrada y salida que existen, además del convertidor analógico digital, comparador analógico, los timer\_counter 0, 1, 2, 3, el watchdog y los USART 0,1, etc.



**Figura 2-11: Ventana de Entradas y Salidas**

**2.2.1.4 Ventana de mensajes**

Esta ventana muestra los mensajes de localización del archivo, mensaje de compilación, mensaje de error y advertencias si es que el código tiene error de escritura o mensaje de OK si el código está bien escrito.

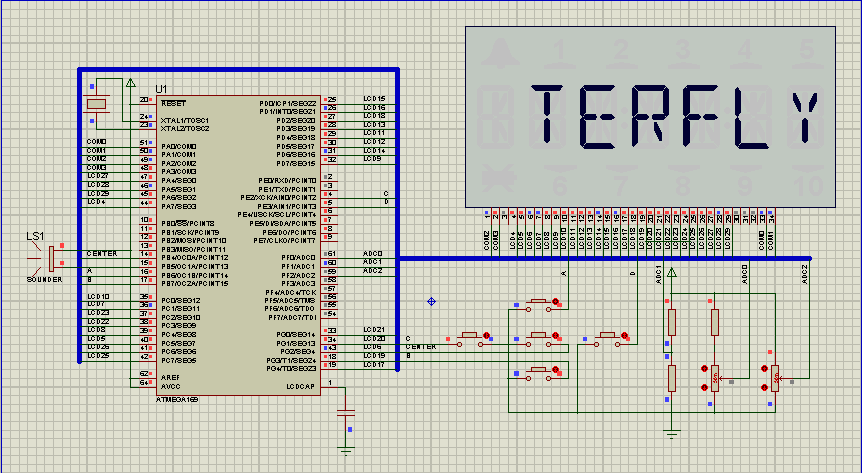
**Figura 2-12: Ventana de mensajes**

**2.2.1.5 Barra de Herramientas**

En esta barra encontramos las diferentes opciones para nuestro código entre ellas: el icono de ensamblaje, el icono de compilación, icono de breakpoint, icono de paso a paso entre líneas, etc.

**Figura 2-13: Barra de herramientas**

**2.2.2 Simulador Proteus versión 7.7**

****

**Figura 2-14: Entorno Proteus 7.7**

Proteus versión 7.7 es un entorno completo diseñado para la realización de cualquier clase de proyecto que conste de elementos electrónicos en todas sus etapas: Diseño, Simulación y Construcción.

Consta de dos partes:

**ISIS.-** Permite diseñar y simular cualquier clase de proyecto electrónico para constatar su funcionamiento.

**Ares.-** Permite el diseño de ruteo del proyecto para ahora si plasmarlo en una placa electrónica. Esta herramienta te ayuda a buscar la mejor forma de cómo colocar los elementos y las líneas del circuito en la placa.

**CAPITULO 3**

**3. DESCRIPCION E IMPLEMENTACION DEL PROYECTO**

**3.1 Diseño Preliminar**

La implementación del proyecto fue dividida en dos partes:

**Transmisor:**

1. Diseñamos una fuente de +5 voltios mediante un regulador de voltaje LM7805 que tiene como función tomar un voltaje de entrada hasta de +40 voltios y setear a un voltaje de salida de +5voltios.
2. Probamos nuestra tarjeta butterfly mediante el hyperterminal de nuestra PC y mediante un osciloscopio para verificar si realmente está trasmitiendo como debe ser los caracteres y las tramas respectivamente.
3. Diseñamos un circuito conversor de señal de RS232 a señal TTL mediante un integrado MAX232, en el cual ingresaremos la señal de salida de nuestra butterfly para obtenerla en TTL y así ingresarla a nuestro módulo de radiofrecuencia.
4. Usamos como módulos de radiofrecuencia los HMTR debido que son inmunes al ruido y por ende nuestra transmisión será casi en un 100%.

**Receptor:**

1. En la etapa receptora tenemos como parte principal nuestro robot que es el que va a recibir las ordenes de cómo y en qué dirección debe moverse.
2. Diseñamos el circuito para nuestro módulo receptor HTMR polarizado con el voltaje que nos proporciona el robot Pololu debido a que este tiene unos pines de VCC y GND donde podemos obtenerlo.

**3.2 Implementación Física**

Para comenzar la implementación física de nuestro proyecto, comenzamos armando en un protoboard toda nuestra circuitería empezando por nuestra fuente de +5v.

* La fuente de +5v será utilizada para la alimentación del módulo de radiofrecuencia y para la polarización de nuestro circuito max232.

Luego tenemos la implementación de nuestro circuito max232 que recibirá la señal de salida del UART de nuestra tarjeta AVR butterfly el cual la convertirá en señal TTL.

A continuación esta señal será trasladada a nuestro módulo transmisor HTMR que por radiofrecuencia se comunicará con nuestro módulo receptor HTMR que se encuentra en nuestro robot Pololu.

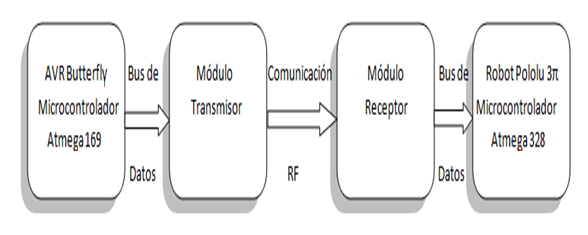
Debemos darnos cuenta en la conexión de nuestros módulos HTMR que entre los pines de trasmisión y recepción estos no se conectan en serie sino cruzados.

**3.3 Descripción del proyecto final**

Al finalizar con lo que es la programación y la implementación del proyecto podemos hacer una breve descripción del funcionamiento del mismo:

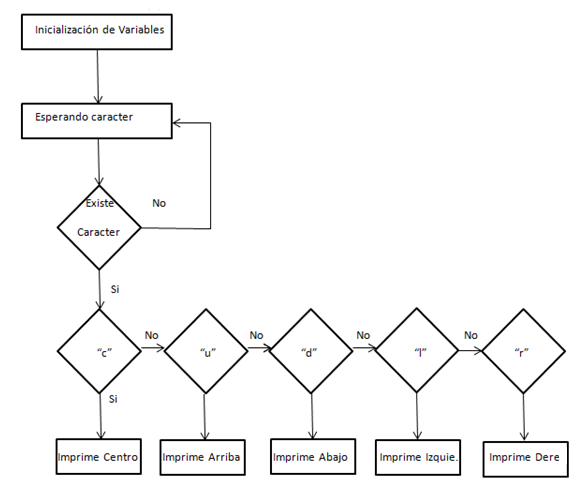
1. En la tarjeta AVR butterfly se muestra al principio la frase “Control Remoto” y se espera hasta que el usuario presione o mueva en cualquiera de las direcciones el joystick para continuar.
2. Una vez presionado o movido el joystick según sea la orden que se haya dado se mostrará la palabra según sea la orden dada, es decir si se presiona saldrá la palabra “Centro”, si se mueve en cualquiera de las direcciones según sea se mostrará las palabras “Arriba”, “Abajo”, “Izquierda”, “Derecha”.
3. Una vez presionado o movido el joystick este por medio del UART transmitirá las ordenes hacia el receptor que es nuestro robot Pololu para indicarle que debe hacer.
4. Ahora en el receptor nuestro robot Pololu al encenderse mostrará “Press B to Start” en donde para inicializarlo debemos realizar esta orden, una vez presionado “B” nos mostrará la frase “Espero Orden” y esto nos indica que está esperando la orden del transmisor o sea de nuestra tarjeta Butterfly.
5. Una vez que está esperando el robot la orden debemos comandar la tarjeta, es decir mover o presionar el joystick según lo que quisiéramos que haga el robot para que este una vez recibida la orden cumpla con aquello.
6. Y listo nuestro robot se moverá por comunicación inalámbrica con interfaz de radiofrecuencia según la dirección en que quisiéramos que se mueva.

**3.4 Diagrama de bloques del proyecto**

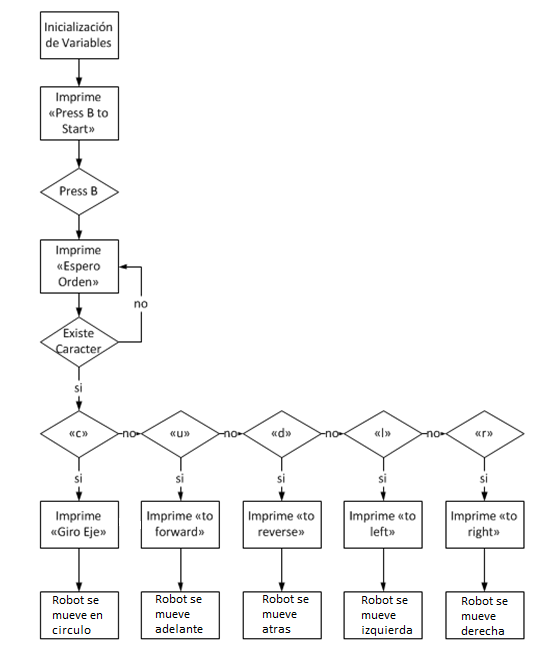


**Figura 3-1: Diagrama de bloques del proyecto**

**3.5 Diagrama de Flujo de la Butterfly**

****

**Figura 3-2: Diagrama de Flujo del Transmisor**

**3.6 Diagrama de Flujo del Robot Pololu**

**Figura 3-3: Diagrama de Flujo del Receptor**

**3.7 Programa principal del transmisor**

#include<avr/io.h>

#include <avr/interrupt.h>

#include <avr/pgmspace.h>

#include <avr/delay.h>

#include <inttypes.h>

#include "mydefs.h"

#include "LCD\_functions.h"

#include "LCD\_driver.h"

#include "button.h"

#include "usart.h"

#define Centro 0

#define Arriba 1

#define Abajo 2

#define Izquierda 3

#define Derecha 4

#define Otros 5

intObtener\_Boton(void);

int main(void)

{

sei();

PGM\_P statetext = PSTR("CONTROL REMOTO");

int input;

// 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);

Button\_Init(); // Initialize pin change interrupt on joystick

LCD\_Init(); // initialize the LCD

CLKPR = (1<<CLKPCE); // set Clock Prescaler Change Enable

// set prescaler = 8, Inter RC 8Mhz / 8 = 1Mhz

CLKPR = (0<<CLKPS1) | (1<<CLKPS0);

USART\_Init(103.08);

if (statetext){

LCD\_puts\_f(statetext, 1);

LCD\_Colon(0);

statetext = NULL;

}

while (1)

{

if (statetext){

LCD\_puts\_f(statetext, 1);

LCD\_Colon(0);

statetext = NULL;

}

input = Obtener\_Boton(); // Read buttons

switch (input) {

case Centro:

statetext = PSTR("CENTRO");

Usart\_Tx('c');

break;

caseDerecha:

statetext = PSTR("DERECHA");

Usart\_Tx('r');

break;

case Izquierda:

statetext = PSTR("IZQUIERDA");

Usart\_Tx('l');

break;

case Arriba:

statetext = PSTR("ARRIBA");

Usart\_Tx('u');

break;

case Abajo:

statetext = PSTR("ABAJO");

Usart\_Tx('d');

break;

default:

break;

}

}

return 0;

}

intObtener\_Boton(void)

{

int Temp1, Temp2;

//PB4-->O Centro

//PB6-->A Arriba

//PB7-->B Abajo

//PE2-->C Izquierda

//PE3-->D Derecha

//cli();

//Centro

Temp1=(PINB) & 0b00010000;

if(Temp1==0b00000000)

{

return Centro;}

Temp1=PINB & 0b01000000;

if(Temp1==0b00000000)

{

return Arriba;}

Temp1=PINB & 0b10000000;

if(Temp1==0b00000000)

{

sei();

return Abajo;}

Temp1=PINE & 0b00000100;

if(Temp1==0b00000000)

{sei();

returnIzquierda;}

Temp1=PINE & 0b00001000;

if(Temp1==0b00000000)

{sei();

returnDerecha;}

sei();

returnOtros;

}

**// Button.h**

#ifdef M162

#define PINA\_MASK ((1<<PINA0)|(1<<PINA1)|(1<<PINA2)|(1<<PINA3)|(1<<PINA4))

#else

#define PINB\_MASK ((1<<PINB4)|(1<<PINB6)|(1<<PINB7))

#define PINE\_MASK ((1<<PINE2)|(1<<PINE3))

#endif

#ifdef M162

#define BUTTON\_A 0 // NORTH

#define BUTTON\_B 1 // EAST

#define BUTTON\_C 2 // WEST

#define BUTTON\_D 3 // SOUTH

#define BUTTON\_O 4 // PUSH

#else

#define BUTTON\_A 6 // UP

#define BUTTON\_B 7 // DOWN

#define BUTTON\_C 2 // LEFT

#define BUTTON\_D 3 // RIGHT

#define BUTTON\_O 4 // PUSH

#endif

//Button definitions

#define KEY\_NULL 0

#define KEY\_ENTER 1

#define KEY\_NEXT 2

#define KEY\_PREV 3

#define KEY\_PLUS 4

#define KEY\_MINUS 5

voidPinChangeInterrupt(void);

voidButton\_Init(void);

chargetkey(void);

charButtonBouncing(void);

**3.8 Programa principal del receptor**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* PROYECTO DE GRADUACION\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <pololu/3pi.h>

#include <avr/pgmspace.h>

#include <pololu/3pi.h>

// Clock Speed

#define F\_CPU 20000000UL

#define BAUD 4800

//#define MYUBRR F\_CPU/8/BAUD-1

#define MYUBRR F\_CPU/16/BAUD-1

#define PIND\_MASK ((1<<PIND0)|(1<<PIND1))

/\* Prototypes \*/

voidUSART\_Init( unsigned int );

unsigned char ReceiveByte (void);

int i;

voiduartSetBaudRate(long intbaudrate);

void setup(void)

{

DDRD = 0xFE;

PORTD |= PIND\_MASK;

DDRB = 0x08; // set PORTD for output

PORTB = 0x00; // set LEDs off

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main ()

{

int j, band;

int temp, temp1, temp2;

char button;

setup(); // Init port pins

do

{

clear();

lcd\_init\_printf();

print("Press B");

lcd\_goto\_xy(0, 1);

print("to Start");

button = wait\_for\_button\_press(ALL\_BUTTONS);

if(button & BUTTON\_B)

{

temp=1;

} else {temp=0;}

}while(temp==0);

band=0;

clear();

USART\_Init(520);

while(1)

{

clear();

print("Espero");

lcd\_goto\_xy(0, 1);

print("Orden");

i = ReceiveByte();

if(i == 'u') // Boton C -> ADELANTE

{

clear();

lcd\_init\_printf();

print("Pololu");

lcd\_goto\_xy(0, 1);

print("to forward");

set\_motors(40,40);

delay\_ms(100);

set\_motors(0,0);

band=0;

}

else if (i == 'd') // Boton D -> ATRAS

{

clear();

print("Pololu");

lcd\_goto\_xy(0, 1);

print("to reverce");

set\_motors(-40,-40);

delay\_ms(100);

set\_motors(0,0);

band=1;

}

else if (i == 'l') // Boton A -> IZQUIERDA

{

clear();

print("Pololu");

lcd\_goto\_xy(0, 1);

print("to left");

set\_motors(0,-40);

delay\_ms(100);

set\_motors(0,0);

band==1;

}

else if (i == 'r') // Boton B -> DERECHA

{

clear();

print("Pololu");

lcd\_goto\_xy(0, 1);

print("to right");

set\_motors(-40,0);

delay\_ms(100);

set\_motors(0,0);

band==1;

}

else if (i == 'c') // Boton STOP -> DETIENE A POLOLU

{

set\_motors(40,-40);

clear();

print("Pololu");

lcd\_goto\_xy(0, 1);

print("GiroEje");

delay\_ms(100);

set\_motors(0,0);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initialize UART \*/

voidUSART\_Init(unsigned intbaudrate)

{

// 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);

}

/\* Read and write functions \*/

unsigned char ReceiveByte (void)

{

/\* Wait for incomming data \*/

while (!(UCSR0A & (1 << RXC0)));

/\* Return the data \*/

return UDR0;

}

voiduartSetBaudRate(long intbaudrate)

{

// calculate division factor for requested baud rate, and set it

int long bauddiv = ((F\_CPU+(baudrate\*8L))/(baudrate\*16L)-1);

UBRR0L= bauddiv;

#ifdef UBRR0H

UBRR0H=(bauddiv>>8);

#endif

}

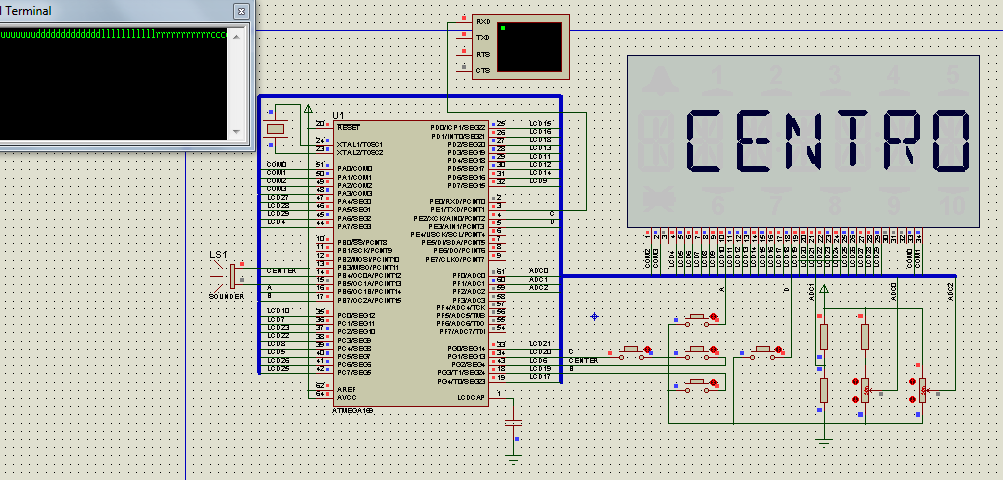
**CAPITULO 4**

**4. SIMULACION Y PRUEBAS**

**4.1 Simulación en Proteus**

Gracias a la ayuda de un excelente programa simulador llamado Proteus nosotros pudimos virtualmente ver cómo funciona nuestro proyecto, si todo lo que queremos que realice lo hace en su totalidad o si algo está fallando, darnos cuenta paso a paso lo que está pasando en las diferentes etapas del proyecto.

En esta primera parte vamos a notar cómo trabaja la parte del transmisor, es decir el funcionamiento de nuestra tarjeta Avr Butterfly como envía el carácter para que así nuestro receptor pueda recibir la orden de movimiento.



**Figura 4-1: Simulación del transmisor**

En la figura 4-1 podemos darnos cuenta claramente como al presionar cualquier botonera se muestra en la pantalla de la LCD el nombre de la dirección según cual hayamos manipulado por ejemplo fue presionada la botonera del centro se presenta la palabra centro.

Además podemos observar también la pantalla del hyperterminal del proteus que nos indica la aparición de un carácter relacionado con cada botonera es decir aparece el carácter “c” “u” “d” “l” “r”.

“c” = botonera del centro

“d” =botonera de abajo

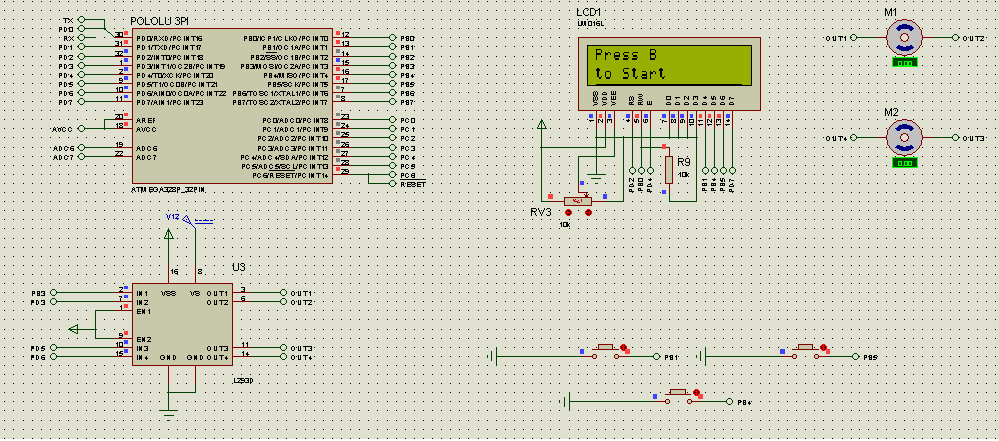
“u” = botonera de arriba

“l” = botonera de la izquierda

“d” = botonera de la derecha

En la figura 4-2 podemos ver la simulación del receptor o sea nuestro robot pololu 3Pi en la cual se muestra en la pantalla del robot la frase “Press B to Start” luego de esto debemos presionar el botón B para iniciar el programa, a continuación sale el mensaje “Espero orden” donde el robot está esperando la dirección a la cual debe moverse sea hacia adelante, hacia atrás, hacia la izquierda, hacia la derecha y dando vueltas.

Además presentamos la simulación de los motores del robot donde apenas llegue la orden ellos comenzarán a moverse.

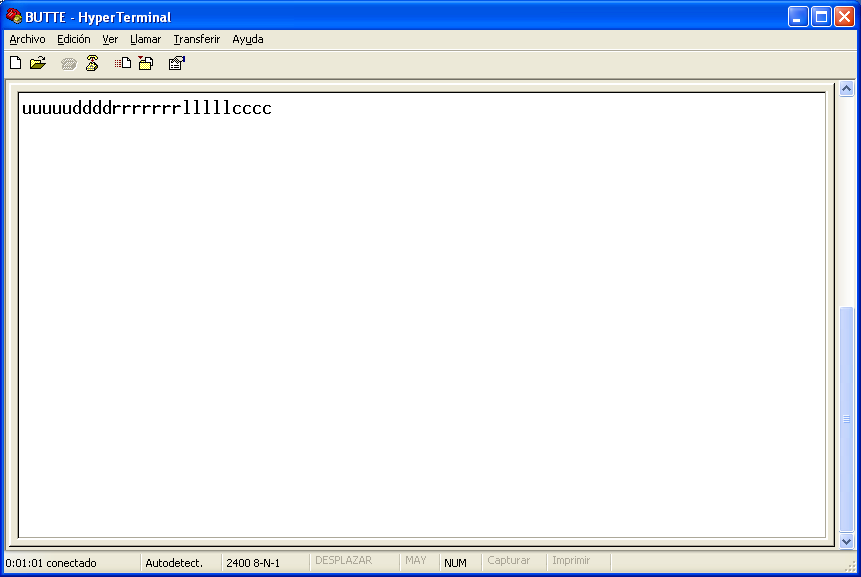


**Figura 4-2: Simulación del receptor**

**4.2 Implementación en Protoboard**

En las siguientes imágenes se muestra la implementación de nuestro proyecto en protoboard con la utilización de todos los elementos mencionados en la sección anterior.

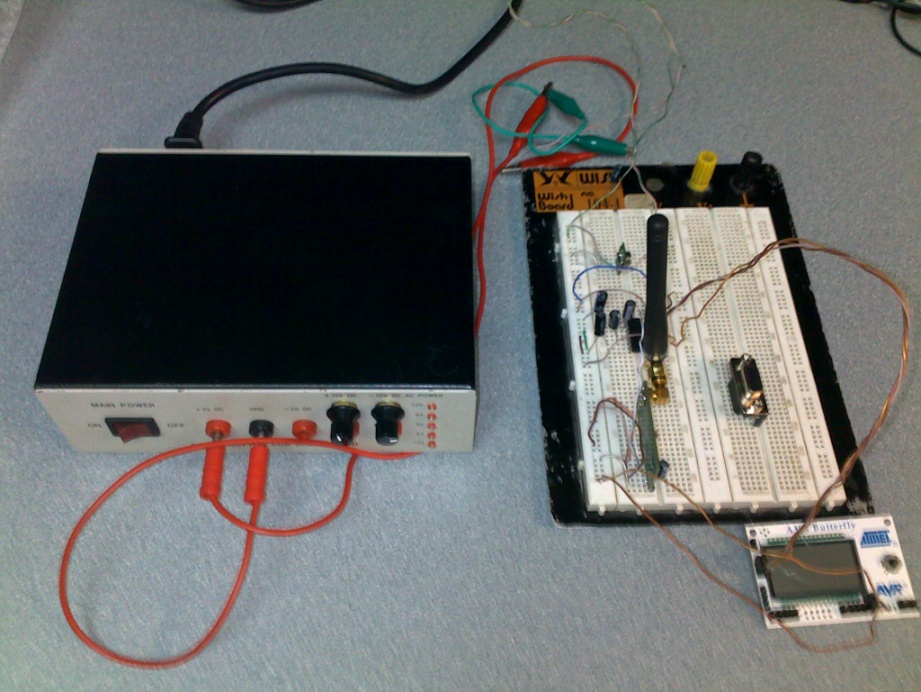
En la siguiente imagen se muestra la transmisión de las tramas representadas por caracteres a través del UART de la tarjeta Butterfly presentadas en el Hyperterminal de la PC.



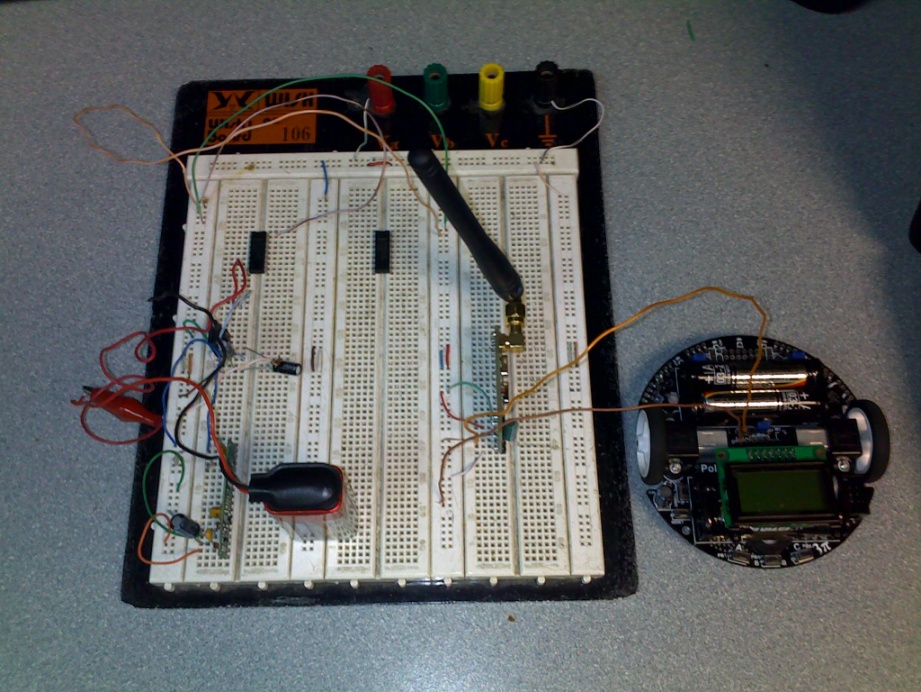
**Figura 4-3: Caracteres mostrados en el hyperterminal**

En la figura 4-4 y demás figuras se muestra la implementación en el protoboard de nuestro proyecto, el diseño preliminar utilizado para prueba de funcionamiento y afinar detalles del mismo.

**Figura 4-4: Presentación de Tarjeta butterfly**

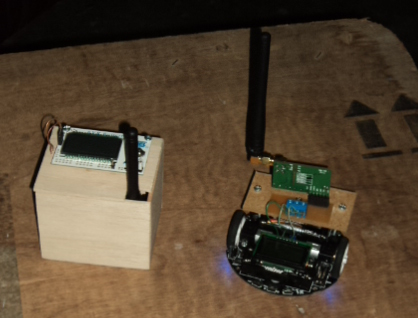
****

**Figura 4-5: Implementación del Transmisor**

****

**Figura 4-6: Implementación del Receptor**

**4.3 Proyecto Final**

En las siguientes figuras se muestra nuestro proyecto ya en funcionamiento de manera real y en la pista de trabajo que puede ser cualquier superficie lisa como por ejemplo el suelo.

**Figura 4-7: Elementos del Proyecto**



**Figura 4-8: Posición Inicial de nuestro Robot**



**Figura 4-9: Robot se mueve adelante**

**Figura 4-10: Robot se mueve atrás**

**Figura 4-11: Robot se mueve a la izquierda**

****

**Figura 4-12: Robot se mueve a la derecha**

**Figura 4-13a: Robot se mueve en círculo**

**Figura 4-13b: Robot se mueve en círculo**