ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Control PID de Velocidad de un motor DC manejado comandado mediante joystick"

TESINA DE SEMINARIO

PREVIA A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

PRESENTADA POR:

GABRIEL ALEJANDRO FIENCO ARELLANO

ERICK MANUEL PERALTA MOLINA

Guayaquil - Ecuador

2011

AGRADECIMIENTO

A Dios por llenar nuestra vida de dichas y bendiciones, y guiarnos siempre bajo su santa voluntad.

A nuestros padres, familiares y compañeros que nos han apoyado en esta meta que nos planteamos como un objetivo firme, su apoyo y esfuerzo nos llena de alegría para culminar una parte importante de nuestra formación académica.

A MSc. Carlos Valdivieso, no solo por ser nuestro tutor sino por ayudarnos de manera incondicional a lo largo de la carrera, y a los compañeros que conformaron este seminario.

DEDICATORIA

A Dios por permitirme culminar una etapa más de mi vida y dejarme compartirla al lado de las personas que quiero y aprecio.

Con mucho cariño principalmente a mi padre, quien siempre ha velado y trabajado por lo mejor para su familia y representa mi ejemplo a seguir, y a mi madre por su apoyo y comprensión.

A mis hermanos Jorge y Carlos con quienes siempre he contado a través de diversas circunstancias.

Familiares y amigos que se han hecho presentes a lo largo de mi carrera, los cuales siempre llevaré en mi corazón.

Gabriel Fienco Arellano

A Dios por estar siempre y en cada etapa de la vida dándome sabiduría para alcanzar mis objetivos.

A mis padres que me supieron guiar por el sendero del bien, por estar dispuestos a darme el apoyo durante mi formación académica.

A los diferentes profesores que impartieron conocimientos y experiencias para el desarrollo intelectual y social.

Erick Peralta Molina

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Valdivieso A.

PROFESOR DE SEMINARIO DE GRADUACIÓN

Ing. Hugo Villavicencio V.

DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesina nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL).

.....

Gabriel Alejandro Fienco Arellano.

.....

Erick Manuel Peralta Molina.

RESUMEN

El objetivo principal de esta tesina es desarrollar, un control PID de velocidad de un motor DC manejado mediante joystick, para esto utilizamos la tarjeta AVR butterfly y jrk21V3, que serán controladas por software implementado por nosotros y basado en el conocimiento adquirido a lo largo de nuestra formación académica. También hemos optado por el uso de los módulos hm-tr 434 para la comunicación inalámbrica entre las 2 tarjetas ya mencionadas.

Este proyecto se desarrolló en lenguaje C, y, se implementó en el programa AVR Studio 4. Parte de las funciones y librerías utilizadas fueron encontradas como programación libre o tomadas del datasheet del atmega169, que es el microcontrolador de la tarjeta AVR butterfly y en base a estos códigos los hemos modificado para que se acoplen a nuestro trabajo.

Las diferentes entradas al controlador PID, tarjeta jrk21v3, funcionan por medio de los 5 movimientos que posee el joystick que forma parte de la tarjeta AVR butterfly, fijando el sentido de giro y velocidad del motor DC, por el cual se muestra un mensaje en el LCD.

ÍNDICE GENERAL

Contenido

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El marco de esta tesina se centra en el manejo mediante joystick de un motor DC que está formado de un controlador PID, para lo cual utilizamos la tarjeta jrk21v3 de pololu. Se emplea una comunicación inalámbrica entre los 2 elementos mencionados, de esta manera se orienta al ámbito de las telecomunicaciones.

Cada capítulo se compone de la siguiente manera:

En el primer capítulo se detalla una descripción general, y se explica el concepto que relaciona nuestro proyecto con el medio que nos rodea, sus posibles aplicaciones en un ámbito social y su parentesco ante trabajos similares con aquel que desarrollamos.

En el segundo capítulo se menciona el hardware a usar: AVR butterfly, módulos hm-tr 434 ttl/rs232, pololu jrk21v3 controlador del motor y pololu enc01a 0j1216 que tiene la función de sensor para emitir la señal de realimentación. Y el software utilizado: AVR Studio 4 con su compilador GCC, el programa de pololu Jrk Configuration Utility, e ISIS PROTEUS para la simulación.

En el tercer capítulo se observa la solución a la problemática, es decir, como se desarrolló el proyecto, los parámetros tomados en cuenta para obtener un funcionamiento adecuado junto con sus desventajas y el código respectivamente compilado que contiene el chip atmega169.

En el cuarto y último capítulo, se muestran valores reales tomados del proyecto funcionando, simulaciones y aquellas funciones que realiza de acuerdo a cada orden que el usuario le imponga como entrada al sistema.

CAPÍTULO 1

1 DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 Situación actual

El manejo de velocidad de un motor DC es algo muy común y de uso diario en nuestros tiempos, sea para un fin recreativo o laboral. El uso de este tipo de motores se lo puede apreciar desde juguetes controlados por niños, hasta maquinaria para fabricar o ensamblar diferentes tipos de elementos manejada ya por gente experimentada.

Nos enfocamos en la necesidad que puede existir por el manejo de los motores DC, pero deseamos que su velocidad se aplique a través de un controlador PID y de esta forma que pueda tener diferentes aplicaciones, ya sea para hacer girar un lector de discos, carros a control remoto, maquinaria, entre otros. También buscamos que nuestro proyecto tenga un uso sin necesidad de una conexión directa, es decir, inalámbricamente, a su vez que dicho control PID de velocidad pueda ser maniobrado por medio de un joystick que lo hace más vistoso y atractivo ante el usuario.

Este trabajo se desarrolló con el fin de dar todas las facilidades para su entendimiento y que tenga un manejo dado tanto por niños como por adultos, dependiendo de la aplicación que se le dé, siempre y cuando se base en los parámetros expuestos en este trabajo.

1.2 Historia

Existen diversos tipos de trabajos que proponen el uso de un control PID, parte de estos proyectos se han desarrollado de manera analógica puesto que un controlador PID puede estar conformado por opamps, resistores, capacitores y transistores. También existe su manejo por medio de microcontroladores integrados a tarjetas, en esto se basa nuestro proyecto.

1.2.1 Posicionador digital 2000 con control PID

FIG 1.1 Posicionador digital 2000.

El Logix 2000 de Flowserve es un posicionador digital con control PID incorporado. Las comunicaciones son 4-20 mA o Modbus. Un controlador PID montado en la válvula actualiza la posición del vástago de la válvula 16 veces por segundo, reduciendo así la demora del sistema de control. El Logix 2000 tiene capacidades de informe de errores y amplio diagnóstico de válvulas. La unidad es totalmente configurable a través de la pantalla de interfaz de usuario, no es necesario ningún dispositivo de configuración externa. [1]

Características:

- Configuración del teclado
- Registrador de datos de 300 puntos
- Comunicaciones Modbus

- Banda muerta <0.03% escala completa
- Repetibilidad <0.035% escala completa
- Repetibilidad <0.035% escala completa

1.2.2 Ventiladores Danfoss

FIG. 1.2 Ventiladores Danfoss.

El convertidor VLT está provisto de un controlador Smart Logic integrado y de 4 controladores PID con ajuste automático, y puede controlar funciones de gestión de aire utilizando ventiladores, válvulas y compuertas. Los sistemas de control para gestión del edificio quedan así liberados, ahorrándose costosos puntos de lectura de datos. [2]

El convertidor HVAC controla un amplio rango de funciones, incluyendo:

- Fines de semana y días laborables
- P-PI de operaciones en cascada para control de la temperatura
- Control multizona de la presión
- Equilibrio de flujo entre aire fresco y aire de salida
- Monitorización de correa

1.2.3 Controlador DataFlo C Serie 17

FIG. 1.3 Controlador DataFlo C Serie 17.

El DataFlo C es un nuevo enfoque de control PID. Esta combinación de microcontrolador basado en controlador de bucle único PID y elemento de control final proporciona control en el punto de uso. El paquete compacto y resistente simplifica los requisitos de cableado aceptando directamente entradas analógicas, RTD o termocupla. La señal no debe acondicionarse, lo que mejora la fiabilidad. [3]

La válvula de control/controlador PID se ajusta fácilmente al bucle con el programa Auto Tune incorporado. El accionamiento DataFlo C, Serie 75 y una válvula de control de asiento característica conforman un elemento de control final ideal para los bucles de control de proceso como; presión, nivel, temperatura, ph, flujo y vacío. [3]

Características:

- Ajuste automática de algoritmo PID
- Parámetros de control o proceso localmente o a través de RS-485
- Calibración de botón pulsador
- Límites de desplazamiento electrónicos y mecánicos
- Alarmas de posición y proceso alto/bajo

CAPÍTULO 2

2 FUNDAMENTO TEÓRICO

El proyecto consta de hardware y software, en la cual van a interactuar la programación de lenguaje de alto nivel de los microcontroladores y la manipulación inalámbrica del control PID de velocidad del motor DC, a través del mando joystick de la tarjeta AVR Butterfly.

2.1 Requerimientos para la aplicación del proyecto

Para programar la tarjeta AVR butterfly que incluye el chip atmega169 se va a utilizar el AVR Studio 4, el cual tiene de 2 compiladores: ATMEL AVR Assembler(compilador lenguaje Assembler) y el AVR GCC(compilador lenguaje C).

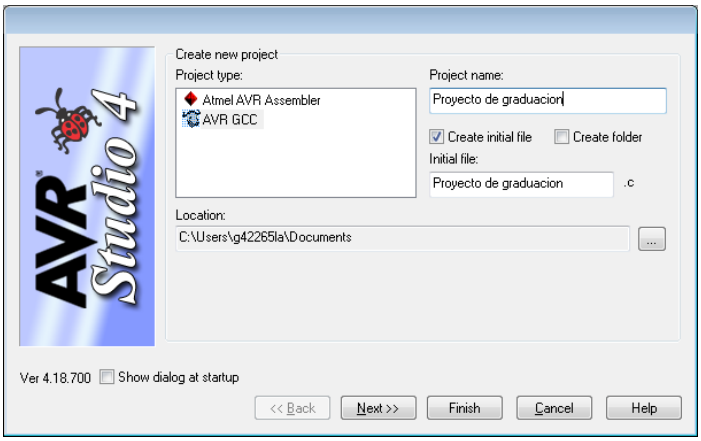


FIG 2.1 Página de inicio del programa AVR STUDIO 4

Para la simulación se va utilizar el software PROTEUS 7.7 SP2, en el cual vamos a probar el código en lenguaje C en la tarjeta virtual AVR butterfly que contiene el atmega169.

FIG 2.2 software PROTEUS 7.7 SP2

En la parte de hardware Hemos optado por el uso de los siguientes elementos para el control PID de velocidad de un Motor DC por medio un joystick:

- AVR Butterfly.
- Módulos HMTR 434.
- Pololu JRK21V3.
- Pololu Encoder enc01a.
- Motor 10:1 Micro Metal gearmotor Hp.
- Pololu Programador USB AVR.

2.2 Herramientas del Software

2.2.1 AVR Studio 4

En la parte de hardware se escogido el AVR Studio 4, que es un software que pertenece a la familia de ATMEL que presenta un entorno estructurado con dos subprogramas los cuales nos permiten crear un archivo en Assembler o en lenguaje C dependiendo de las necesidades del usuario.

Dentro del AVR STUDIO existe el AVR ASSEMBLER, este un programa el cual nos permite implementar funciones escritas en lenguaje ensamblador, el cual es el lenguaje más cercano al lenguaje máquina y él nos permite optimizar las rutinas que se emplean en la configuración de los microcontroladores ATMEL.

En AVR STUDIO 4 consta varias plataformas de depuración y la que vamos a usar en el proyecto es el AVR Simulator 2, para el cual escogemos el atmega169. Los archivos que se generan en la creación de un código en assembler (.***asm***) y en lenguaje C el archivo que se crea en un (.***c***) los cuales son los que guardan la programación que se va a colocar en el microcontroladores.

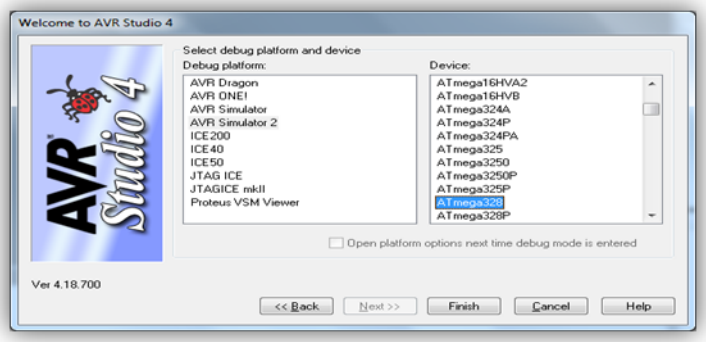


FIG 2.3 Pantalla de selección de plataforma AVR Studio 4

2.2.2 PROTEUS

PROTEUS es una compilación de programas de diseño y simulación electrónica, desarrollado por Labcenter Electronics que consta de los dos programas principales: Ares e Isis, y los módulos VSM y Electra. [4]

El Programa ISIS, **I**ntelligent **S**chematic **I**nput **S**ystem (*Sistema de Enrutado de Esquemas Inteligente*) permite diseñar el plano eléctrico del circuito que se desea realizar con componentes muy variados, desde simples resistencias, hasta alguno que otro microprocesador o microcontrolador, incluyendo fuentes de alimentación, generadores de señales y muchos otros componentes con prestaciones diferentes. Los diseños realizados en Isis pueden ser simulados en tiempo real, mediante el módulo VSM, asociado directamente con ISIS. [4]

Una de las prestaciones de Proteus, integrada con ISIS, es **VSM**, el **V**irtual **S**ystem **M**odeling (*Sistema Virtual de Modelado*), una extensión integrada con ISIS, con la cual se puede simular, en tiempo real, con posibilidad de más rapidez; todas las características de varias familias de microcontroladores, introduciendo nosotros mismos el programa que controlará el microcontrolador y cada una de sus salidas, y a la vez, simulando las tareas que queramos que lleve a cabo con el programa. Se pueden simular circuitos con microcontroladores conectados a distintos dispositivos, como motores, lcd´s, teclados en matriz, etc. Incluye, entre otras, las familias de PIC's PIC10, PIC12, PIC16, PIC18, PIC24, dsPIC33 y algunos elementos de la familia ATMEL. ISIS es el corazón del entorno integrado PROTEUS. Combina un entorno de diseño de una potencia

excepcional con una enorme capacidad de controlar la apariencia final de los dibujos. [4]

ARES, o **A**dvanced **R**outing and **E**diting **S**oftware (*Software de Edición y Ruteo Avanzado*); es la herramienta de enrutado, ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso, permitiendo editar generalmente, las capas superficial (Top Copper), y de soldadura (Bottom Copper). [4]

Utilizando el módulo Electra (Electra Auto Router), el cual, una vez colocados los componentes trazará automáticamente las pistas realizando varias pasadas para optimizar el resultado. Con Ares además se puede tener una visualización en 3D del PCB que se ha diseñado, al haber terminado de realizar la ubicación de piezas, capas y ruteo, con la herramienta "3D Visualization", en el menú output, la cual se puede demorar, solo haciendo los trazos un periodo de tiempo un poco más largo que el de los componentes, los cuales salen al empezar la visualización en 3D. [4]

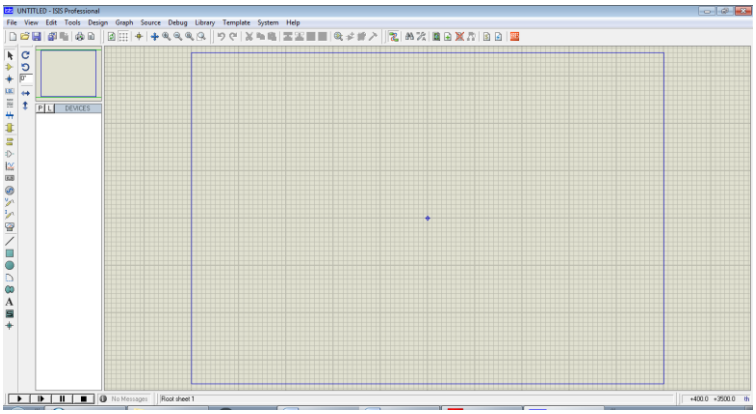


FIG. 2.4 Ilustración interfaz gráfica simulador Proteus

2.2.3 Pololu JRK 21v3 Controlador de Motor

Es el controlador del motor dc que soporta cuatro modos de interfaz: USB, nivel serial lógico, voltaje analógico y por radio control (RC). El controlador se puede utilizar con retroalimentación de velocidad de circuito cerrado o el control de posición que también puede usarse en lazo abierto.

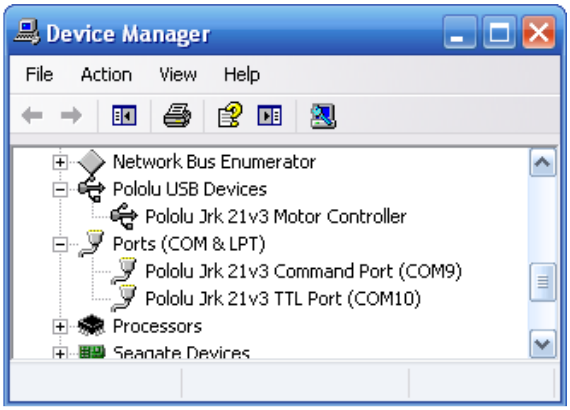


FIG. 2.5 Windows XP administrador de dispositivo Pololu Jrk

El Pololu Jrk Configuration Utility , es el software de aplicación que permite al usuario cambiar todas las configuraciones del Pololu JRK 21v3 Controlador de Motor, así mismo como ver información en tiempo real de su estado.

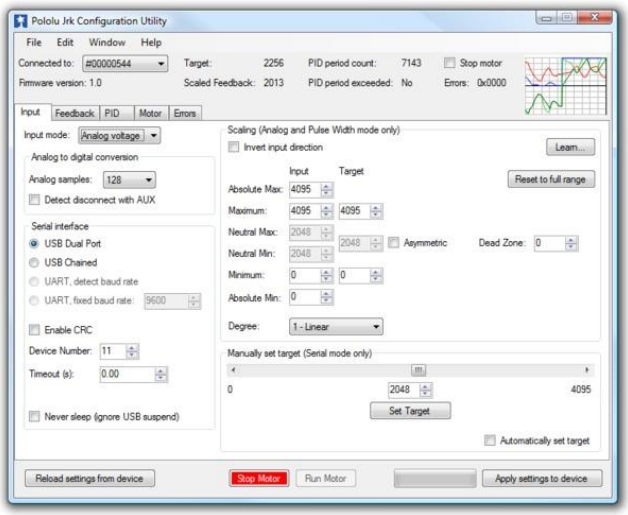


FIG. 2.6 Pololu Jrk Configuration Utility

En la opción Input contiene el modo de configuración de la realimentación, que es controlada y monitoreada. Hay tres modos Input:

- **Serial**
- **Analógico**
- **Ancho de pulso**

2.3 Herramientas de Hardware

2.3.1 AVR Butterfly

El joystick del AVR Butterfly enviará señales, las cuales son controladas por nuestro código que será cargado en el ATMEGA169, también perteneciente a dicho producto y para su transmisión mediante el módulo de radio frecuencia usamos comunicación USART.

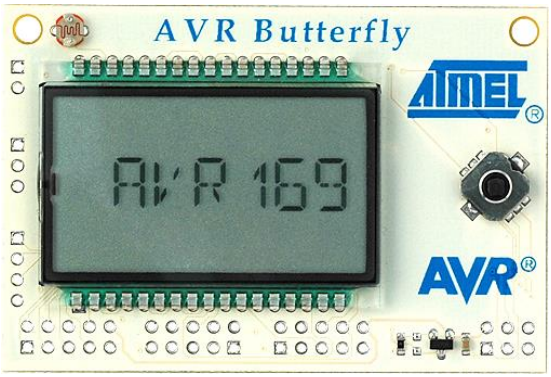


FIG. 2.7 AVR BUTTERFLY

2.3.2 HM-TR 434 TTL/RS232

Posteriormente las señales obtenidas y empaquetadas a manera de una trama de 8bits serán enviadas por los módulos HMTR 434 TTL/RS232, los cuales corresponden a la parte inalámbrica de nuestro trabajo. Se enviarán dichas

tramas por la banda de los 343MHz y serán recibidos por el jrk21v3.

Características del HMTR 434:

- Modulación FSK, alta inmunidad a interferencias.
- Comunicación Half-duplex en 2 vías.
- Bandas ISM 315/434/868/915MHz.
- Frecuencias programables que permite ser usado en aplicaciones FDMA (frequency division multiple acces).
- Conversión de RF a UART auto controlado, confiable y fácil de usar.
- Formato UART configurable, con data rate de 300 - 19200bps
- Interfaz UART estándar con niveles lógicos TTL y RS232 disponibles.
- Dimensiones compactas, 0.100'' separación de pines (similar a un conector SIP o header) y conector SMA para antena. [5]

FIG. 2.8 HM-TR 434 TTL

2.3.3 Pololu Jrk21v3

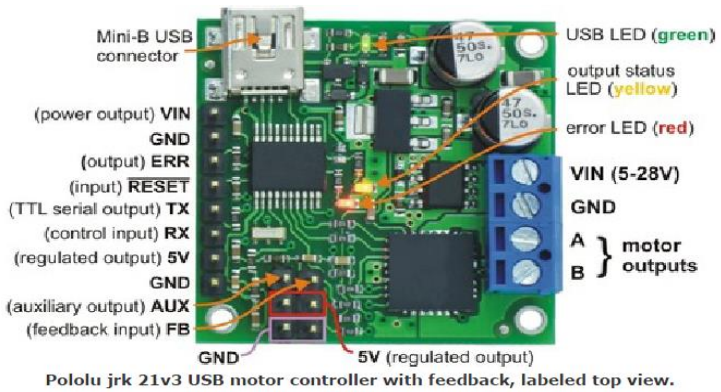
La señal ya captada por el jrk21v3 deberá ser asincrónica dado que este integrado solo maneja UART. La entrada serial que usaremos son códigos específicos los mismo que

deberán ser reconocidos por esta tarjeta y entendidos para mover el motor, cabe recalcar que nosotros no programamos el jrk21v3 nos basamos en su hoja de datos para lograr enviar señales que muevan el motor.

–	Jrk 21v3
Motor channels:	1
Operating voltage:	5 – 28 V
Continuous output current:	3 A
Peak output current:	5 A
Auto-detect baud rate range:	300 – 115,200 bps
Available fixed baud rates:	300 – 115,200 bps
Available PWM frequencies:	20 kHz, 5 kHz
Reverse voltage protection?:	Yes
USB connector style:	USB Mini-B

FIG. 2.9 Especificaciones de la tarjeta pololu jrk21v3

Las señales enviadas se basan en el Protocolo de Comando, específicamente usando los bytes de datos para el cambio de velocidad, mientras que con el bytes de comando su sentido de giro. Esto que hemos mencionado compone a su vez el protocolo compacto, en el cual nos basaremos para enviar estos valores fijos.



Pololu jrk 21v3 USB motor controller with feedback, labeled top view.

FIG. 2.10 Tarjeta jrk21v3

2.3.4 Pololu Encoder enc01a 0J1216

El encoder de pololu a usar trabaja con 2 sensores infrarrojos, dichos sensores se encuentran separados lo suficiente para dar formas de onda con 90 grados fuera de fase, permitiendo que la dirección de rotación sea determinada. Este elemento esta calibrado para trabajar de 4.5V a 5.5V. [10]

FIG. 2.11 Encoder pololu para llanta

CAPÍTULO 3

3 DISEÑO DE LA SOLUCIÓN

3.1 Descripción del proyecto

En el presente trabajo nos enfocamos en el desarrollo de un control P, I, D, para conseguir ajustar la velocidad de un motor DC bajo el mando de un joystick integrado al AVR Butterfly, dicha tarjeta no solo consta con este elemento, también se compone de un LCD que nos permite proporcionar una interfaz gráfica de aquellas especificaciones que implementaremos en nuestro proyecto.

Cabe recalcar que el AVR Butterfly tiene como componente principal un chip perteneciente a la familia ATMEL, es el atmega169, el cual en conjunto con una serie de elementos electrónicos terminan de conformar esta tarjeta proporcionándonos pines de salida para varios usos: comunicación serial, grabación del microcontrolador, o manejo de sus puertos B y D a gusto del usuario.

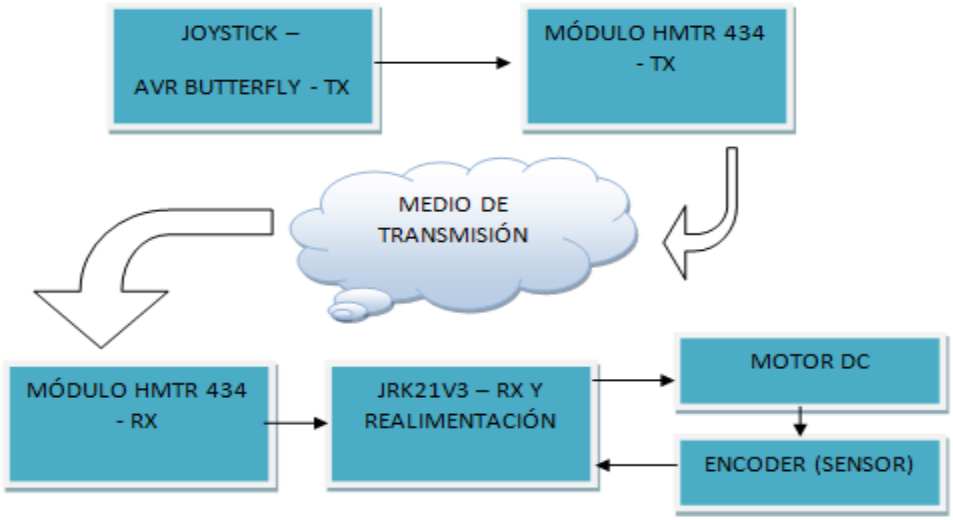
Dado que se necesita una interfaz entre ambos elementos, es decir, el joystick del AVR Butterfly y el Motor DC, usaremos una comunicación por radio frecuencia con los módulos HMTR 434 y la tarjeta jrk21v3 que se encargará del control PID de velocidad de dicho motor, esta a su vez recibirá la señal de realimentación proveniente de otra tarjeta de pololu que tendrá la función de sensor, la tarjeta enc01a 0j1216. Con respeto a los módulos, estos

ayudarán para ver de una forma más fácil y con comodidades la importancia de las comunicaciones inalámbricas que se enfocan en el área de telecomunicaciones tomando en cuenta distancia, tipo de modulación y conexiones entre transmisión y recepción, que son los parámetros que nos interesarán al usar dichos módulos.

3.2 Diagrama de bloques

Se enviarán 5 diferentes tipos de señales que nos provee el manejo del joystick de acuerdo a los movimientos que puede realizar: arriba, abajo, izquierda, derecha y centro.

Usando comunicación UART serán enviadas tramas de 8 bits, a través de uno de los módulos de radio frecuencia, las mismas que serán receptadas por su par y posteriormente enviadas a la tarjeta jrk21v3 de pololu. Dicha tarjeta se encargará de dar la velocidad y sentido de giro correspondiente al motor DC y a su vez se encargará de la realimentación característica del controlador PID de acuerdo a los comandos que el usuario envié por medio de las tramas de 8 bits.

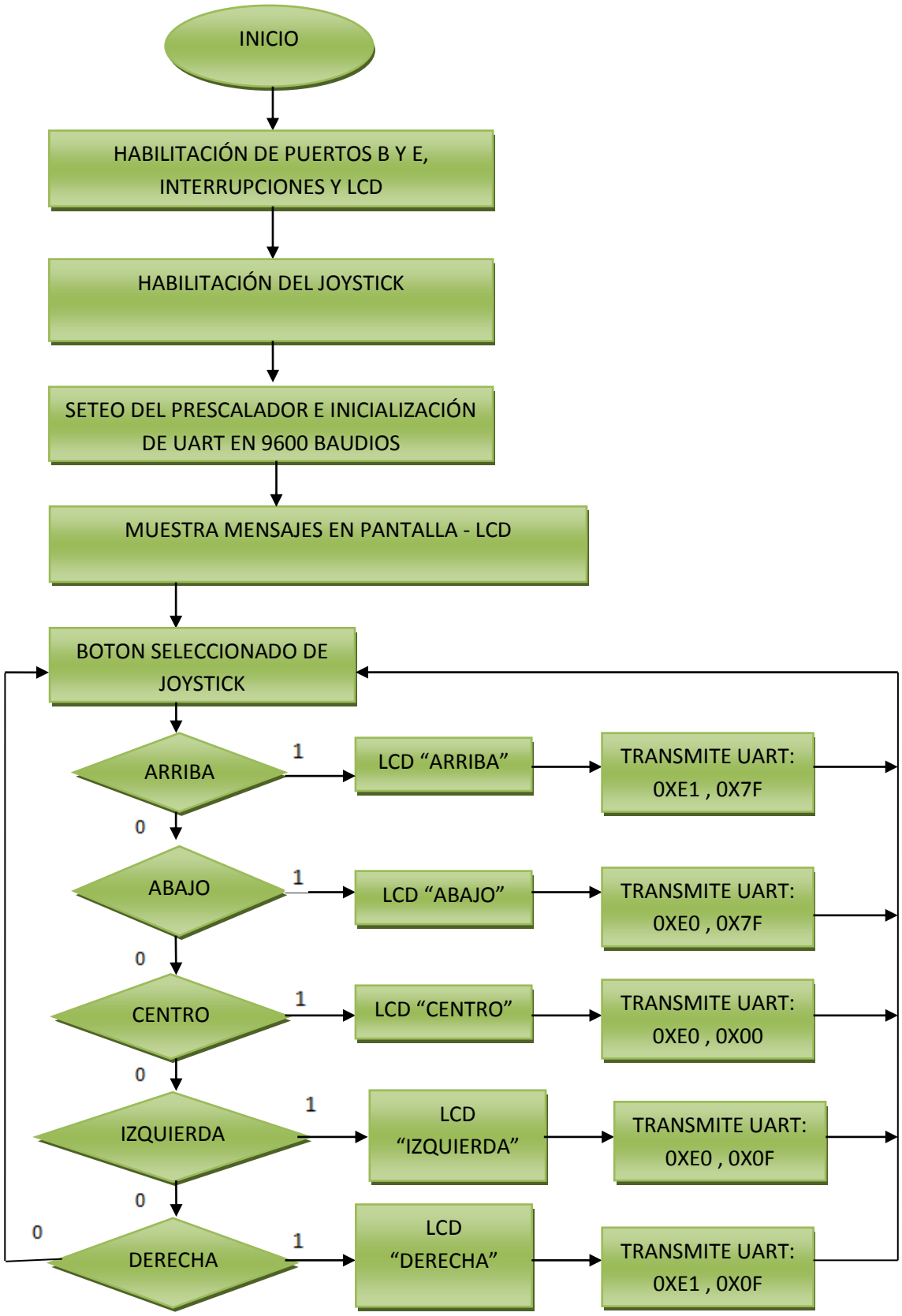


3.3 Entrada del controlador

Esta señal proviene de la tarjeta nombrada anteriormente, es decir, AVR butterfly. Para saber que señales se enviarán por medio de los módulos inalámbricos nos basamos en los valores dentro del rango que ofrece de la tarjeta jrk21v3, los cuales encontramos en la guía de pololu denominada "**Pololu Jrk USB Motor Controller User's Guide**", de esta forma supimos que primero se especifica la trama de 8 bits con el sentido de giro, sea este horaria o anti-horaria, seguida de otra que nos dará la velocidad a la que girará el motor.

De acuerdo a cada movimiento que presente en el motor se especificará en pantalla, LCD incorporado en la tarjeta AVR butterfly, aquel que se ha realizado. Proporcionamos una interfaz gráfica para lograr que nuestro proyecto sea más atractivo al usuario al momento de interactuar con él.

3.3.1 Diagrama de flujo



3.3.2 Código fuente

```
/*                MICROCONTROLADORES AVANZADOS

                TESINA DE SEMINARIO

        CODIGO DE TRANSMISOR CORRESPONDIENTE

            A LA ENTRADA DEL SISTEMA

INTEGRANTES : FIENCO GABRIEL, PERALTA ERICK

PROF: MSC. CARLOS VALDIVIEZO

DESCRIPCIÓN:ÉL MICROCONTROLADOR A USAR ES EL
                ATEMGA169 CONECTADO A UN JOYSTICK
                TODO ESTO EMPAQUETADO EN UNA
                TARJETA DENOMINADA AVR BUTTERFLY
                CON CADA MOVIMIENTO DEL JOYSTICK
                SE LEE UN VALOR A LA ENTRADA
                DEPENDIENDO AL PIN DEL PUERTO AL
                QUE SE ENCUENTRE CONECTADO DICHO
                ELEMENTO.  DE CADA UNO DE LOS 5
                MOVIMIENTOS QUE POSEE SERÁN
ENVIADAS
                2 TRAMAS POR COMUNICACIÓN UART
                CORRESPONDIENTES A GIRO Y
VELOCIDAD
                PARA UN MOTOR DC.
*/
//Declaración de constantes
#define  Centro    0
#define  Arriba    1
#define  Abajo     2
#define  Izquierda 3
```

```
#define  Derecha   4
#define  Otros     5

//Libreias 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,estado_pas;

        //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 52= 9600 baudios
    USART_Init(52);

    LCD_puts("MICROCONTORLADORES",1); // MUESTRA EN PANTALLA
    _delay_ms(4500);
    LCD_puts("AVANZADOS",1);  // MUESTRA EN PANTALLA
    _delay_ms(2000);
    LCD_puts("FIENCO GABRIEL",1); // MUESTRA EN PANTALLA
    _delay_ms(4000);
    LCD_puts("PERALTA ERICK",1);  // MUESTRA EN PANTALLA
    _delay_ms(3000);
    LCD_puts("",1);   // MUESTRA EN PANTALLA

    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
if(input!= estado_pas)
{
switch (input)
            {
            case Centro:
                              statetext = PSTR("CENTRO");
                             Usart_Tx(0XE0);
                              Usart_Tx(0X00);
                              estado_pas = input;
            break;
            case Derecha:
                              statetext = PSTR("DERECHA");
                              Usart_Tx(0XE1);
                              Usart_Tx(0X0F);
                              estado_pas = input;

            break;
            case Izquierda:
```

```
                    statetext = PSTR("IZQUIERDA");
                    Usart_Tx(0XE0);
                    Usart_Tx(0X0F);
                    estado_pas = input;
                break;
                case Arriba:
                    statetext = PSTR("ARRIBA");
                   Usart_Tx(0XE1);
                    Usart_Tx(0X7F);
                    estado_pas = input;
                break;
                case Abajo:
                    statetext = PSTR("ABAJO");
                    Usart_Tx(0XE0);
                    Usart_Tx(0X7F);
                    estado_pas = input;
                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;
}
```

3.4 Recepción de la señal de entrada y controlador

La señal que se recepta, provista por los módulos hm-tr 434 que la obtienen del butterfly, va a ser entendida por un protocolo que posee la tarjeta pololu jrk21v3 y que debe constar de 2 tramas de 8 bits enviadas de forma consecutiva, a esto se lo denomina protocolo compacto.

Para indicar el sentido y velocidad que conducirán a el motor DC usamos el protocolo compacto que funciona de la siguiente manera: byte de comando (MSB set), byte de datos en ese orden deben ser enviadas las tramas. Estos paquetes a su vez se rigen a normas específicas para cada uno, el byte de comando siempre debe tener el bit más significativo en 1 y tiene un rango de 128 – 255 o en hexadecimal de 0x80 – 0xFF, mientras que el byte de datos siempre debe tener su bit más significativo en 0 y trabaja en un rango de 0 – 127 o en hexadecimal 0x00 – 0x7F. [6]

Con respecto al sentido de giro que nos ofrece este protocolo tendremos la opción de: habilitar objetivo de baja resolución en avance o retroceso, lo cual se específica por software.

- Habilitar objetivo de baja resolución en avance, se maneja bajo la siguiente estructura: 0xE1, Magnitud. El valor de la magnitud debe ser un valor en el rango de 0 – 127. [6]

- Habilitar objetivo de baja resolución en retroceso, se maneja bajo la siguiente estructura: 0xE0, Magnitud. El valor de la magnitud debe ser un valor en el rango de 0 – 127. [6]

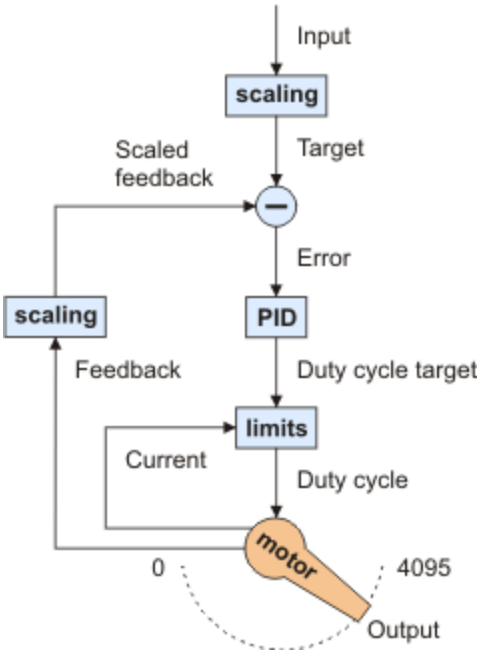


FIG. 3.1 Diagrama de un sistema típico de realimentación, con cualidades del jrk

3.5 Señal de realimentación

Esta señal la obtendremos de la tarjeta enc01a 0J1216 de pololu, sensor, a través de su salida OUTB que se conectará al pin FB de la tarjeta que también pertenece a pololu, jrk21v3. Por medio del programa Jrk Configuration Utility habilitamos la opción FeedBack en modo de frecuencia digital.

Cabe recalcar que trabajamos con un target en el rango de 0 – 4095, por lo tanto debe aplicarse el mismo valor en la opción de realimentación y recordemos que gracias al protocolo compacto obtenemos el target por la expresión: target = 2048 – (16)x(Magnitud).

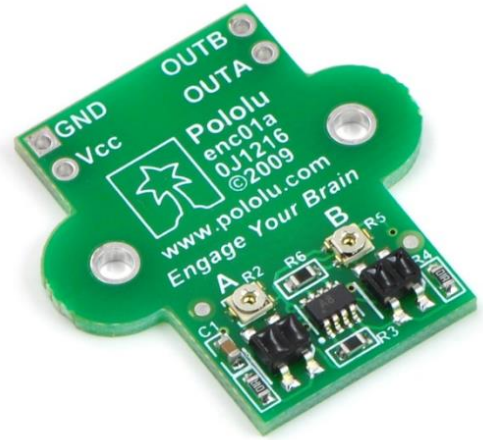
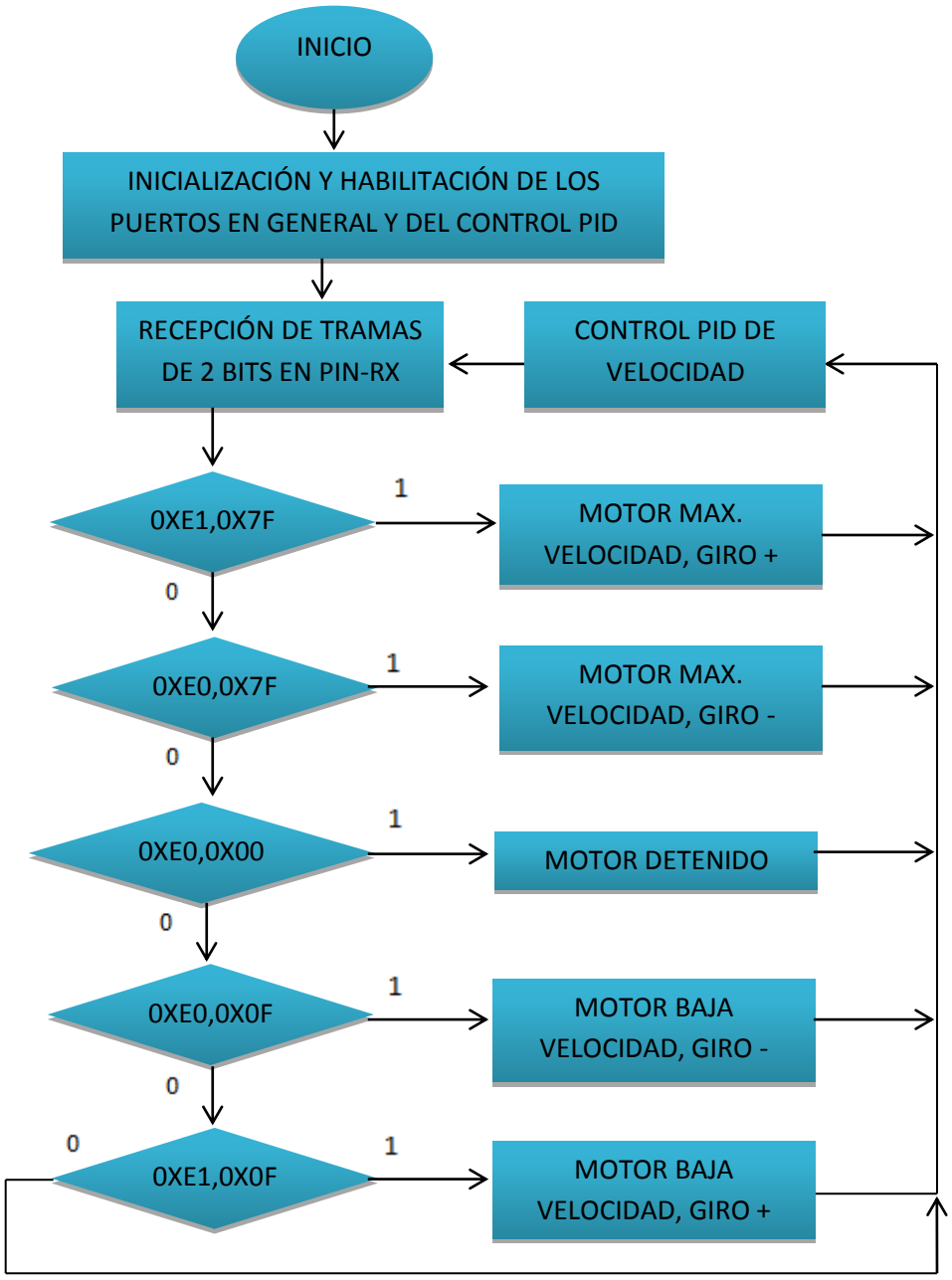


FIG. 3.2 Pololu enc01a 0j1216

FIG. 3.3 Pololu enc01a 0j1216 con llanta

3.6 Diagrama de flujo general



CAPÍTULO 4

4 ANÁLISIS DE RESULTADOS, VALIDACIÓN Y PRUEBAS

4.1 Simulación en PROTEUS

Nosotros hemos tomado como programa de simulación ISIS 7 Professional y hemos cargado el código fuente del transmisor, de la entrada al controlador, en el atmega169 para obtener una idea de cómo reaccionará el sistema de forma virtual. A continuación se muestra parte de los mensajes de inicio que se aprecian en el LCD y cada respuesta que se obtiene al mover el joystick en sus 5 direcciones:

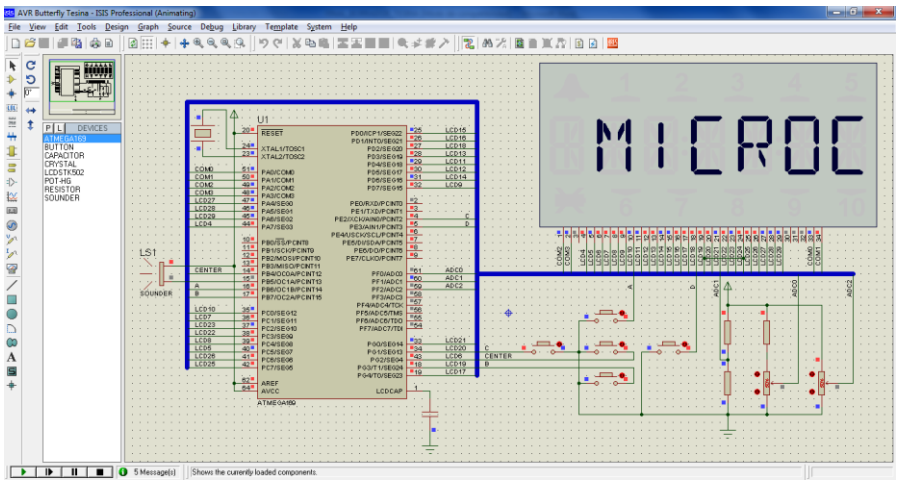


FIG. 4.1.1 Primer mensaje que muestra el proyecto "MICROCONTROLADORES AVANZADOS"

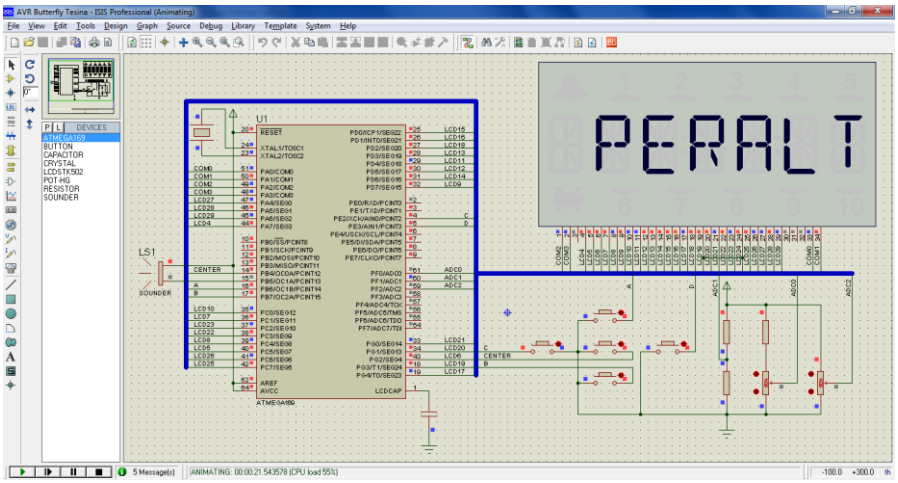


FIG. 4.1.2 Se observa parte del nombre de uno de los integrantes de la tesina.

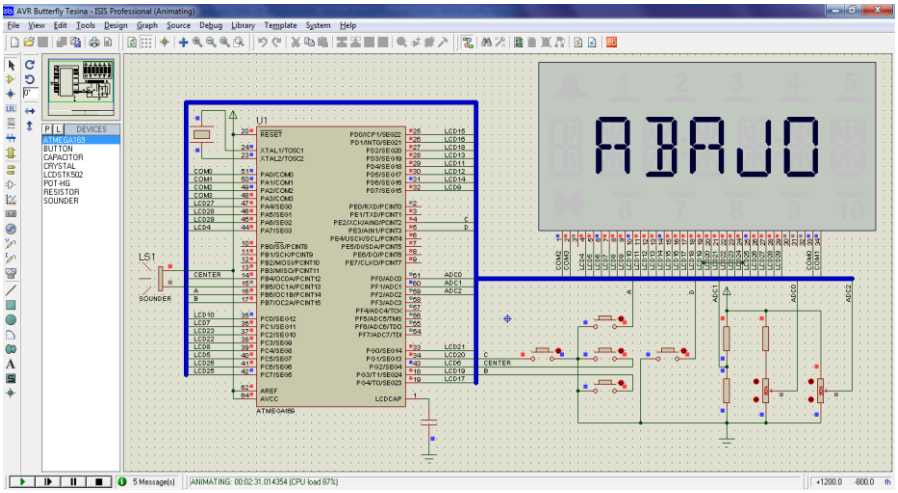


FIG. 4.1.3 Máxima velocidad en sentido horario.

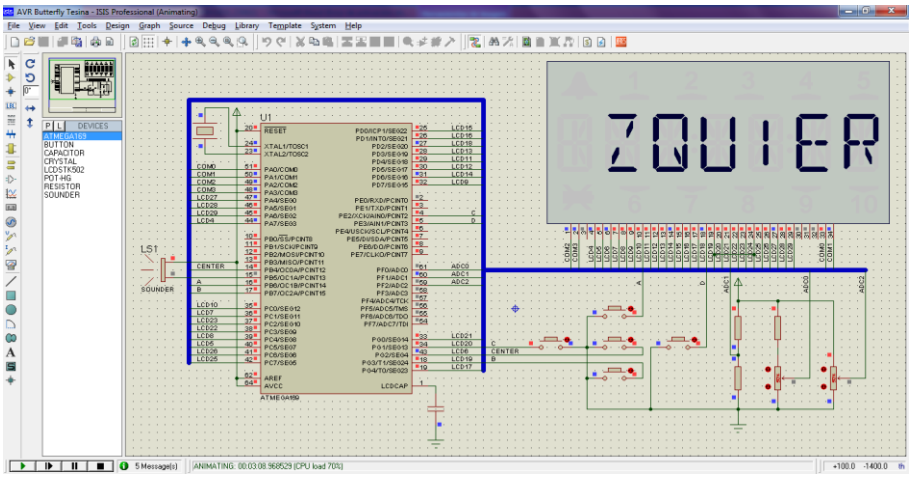


FIG. 4.1.4 Mínima velocidad en sentido horario.

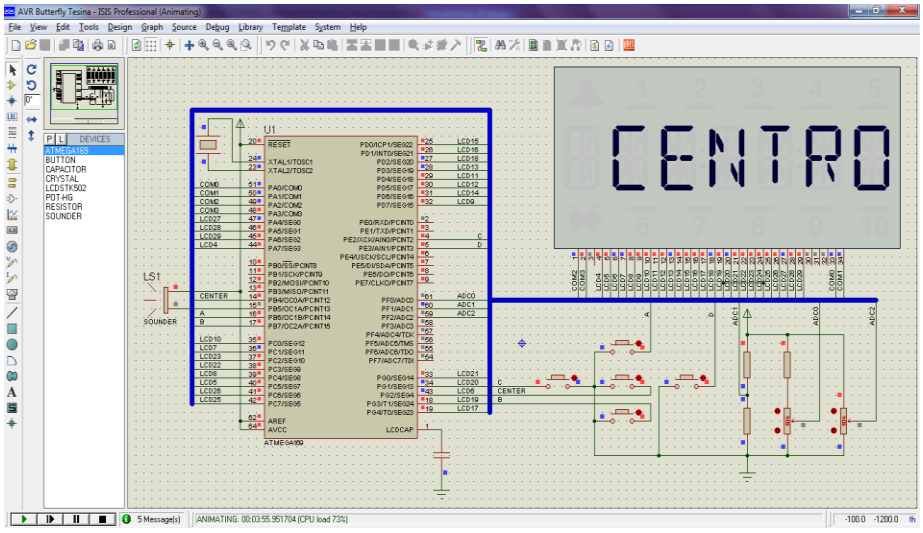


FIG. 4.1.5 Señal que el motor se encuentra sin movimiento alguno.

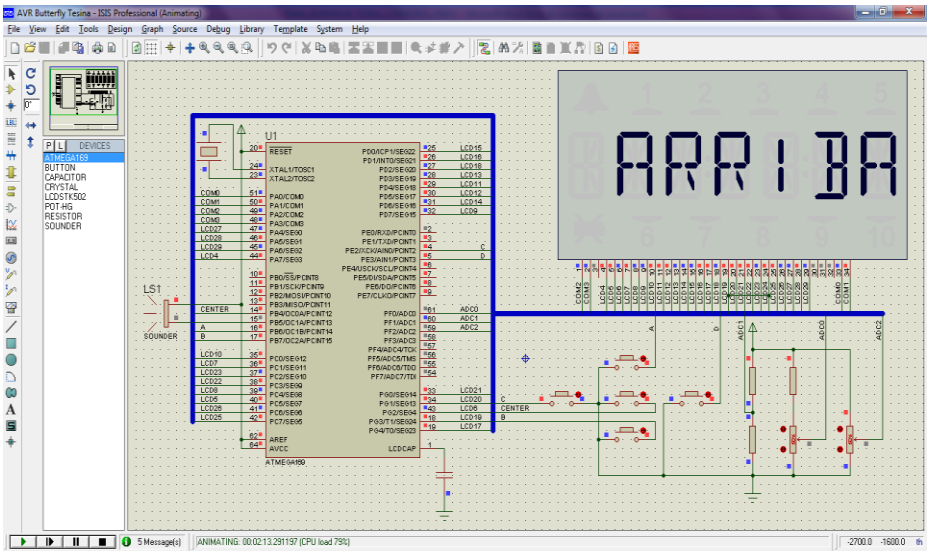


FIG. 4.1.6 Máxima velocidad en sentido anti-horario.

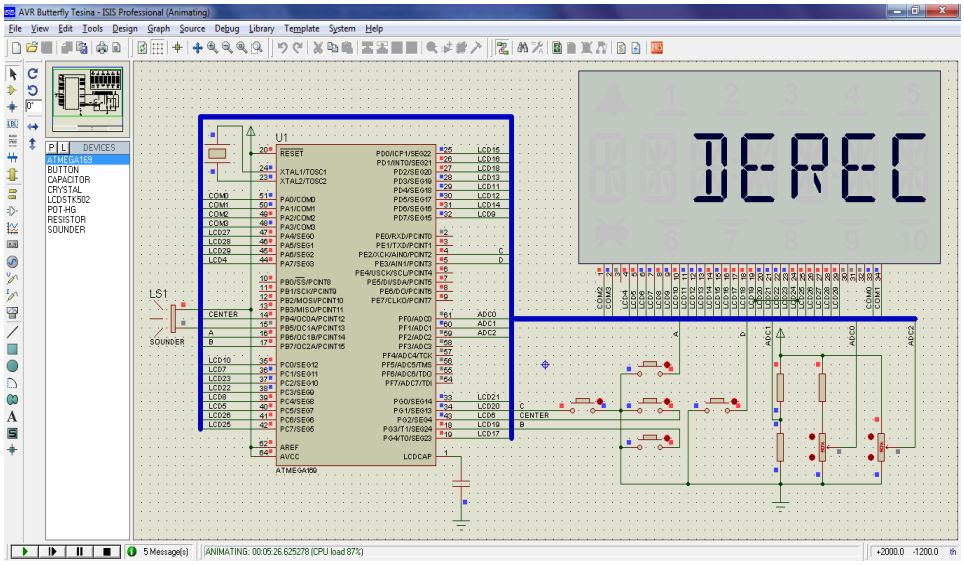


FIG. 4.1.7 Mínima velocidad en sentido anit-horario.

4.2 Configuración del controlador y resultados

Los parámetros para la tarjeta jrk21v3 se establecen vía conexión usb donde nosotros optamos por una entrada en modo serial, a un baud rate fijo de 9600, el número del dispositivo es 11 o 0x0B en hexadecimal. Como ya hemos mencionado el target tiene un rango de 0 – 4095, donde 2048 representa un punto en el cual no se apreciará movimiento por parte del motor.

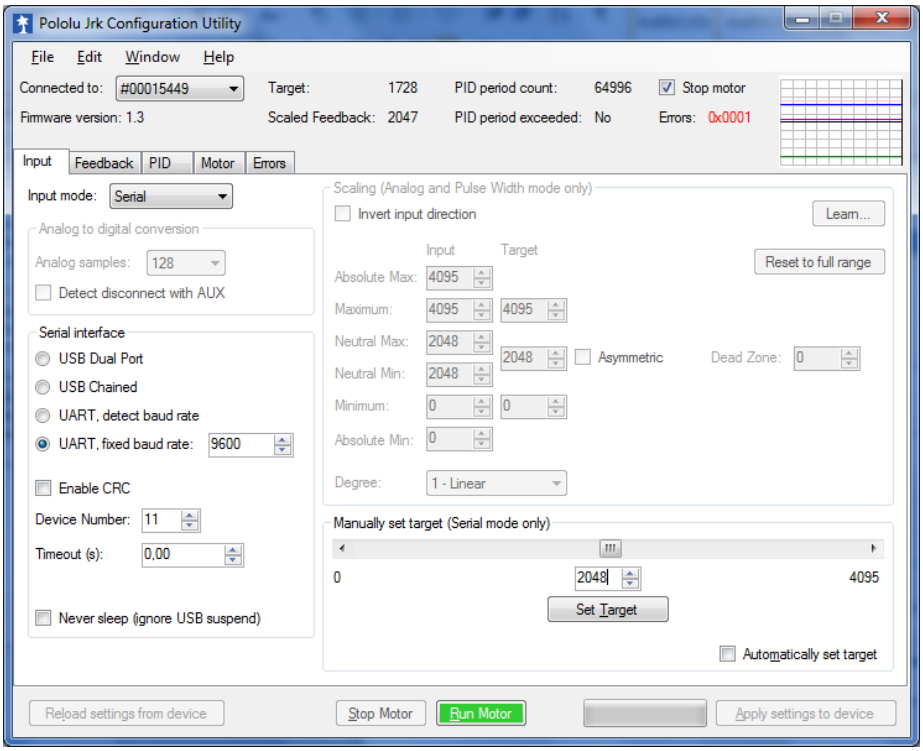


FIG. 4.2.1 Programa Jrk Configuration Utility opción entrada.

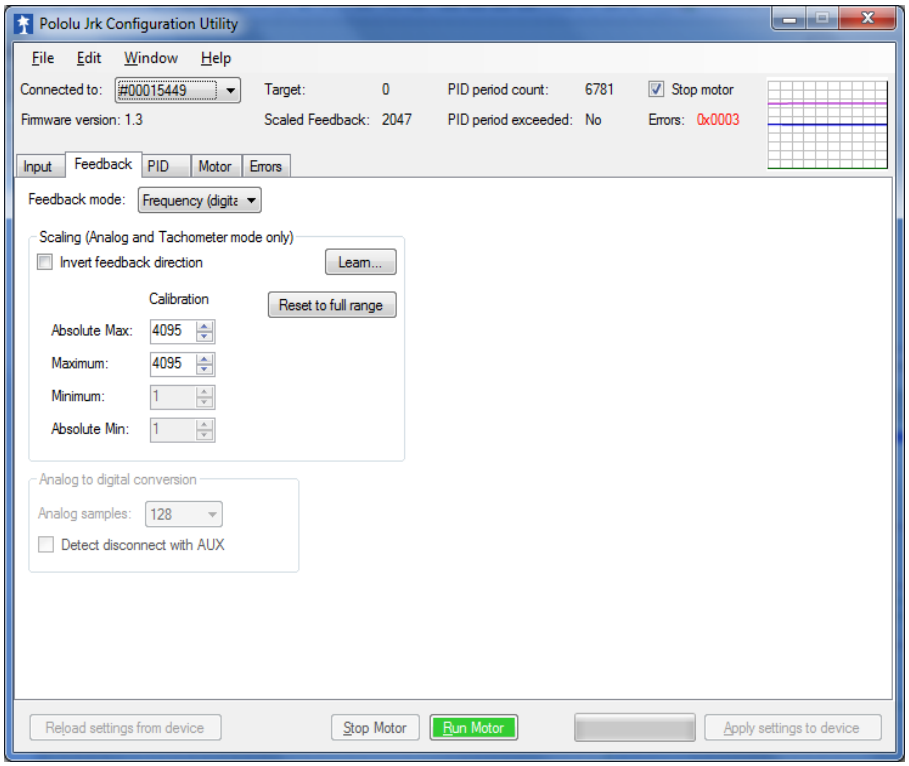


FIG. 4.2.2 Programa Jrk Configuration Utility opción realimentación.

Hemos basado nuestro P, I, D en parámetros que aumenten la ganancia del sistema, tomando como referencia el momento que esta sin realimentación, en un 50% y hemos colocado un integral “I” de modo que no sea tan grande y se ajuste al tiempo requerido, porque si lo aumentamos demasiado conseguiremos una inestabilidad en el control del mismo.

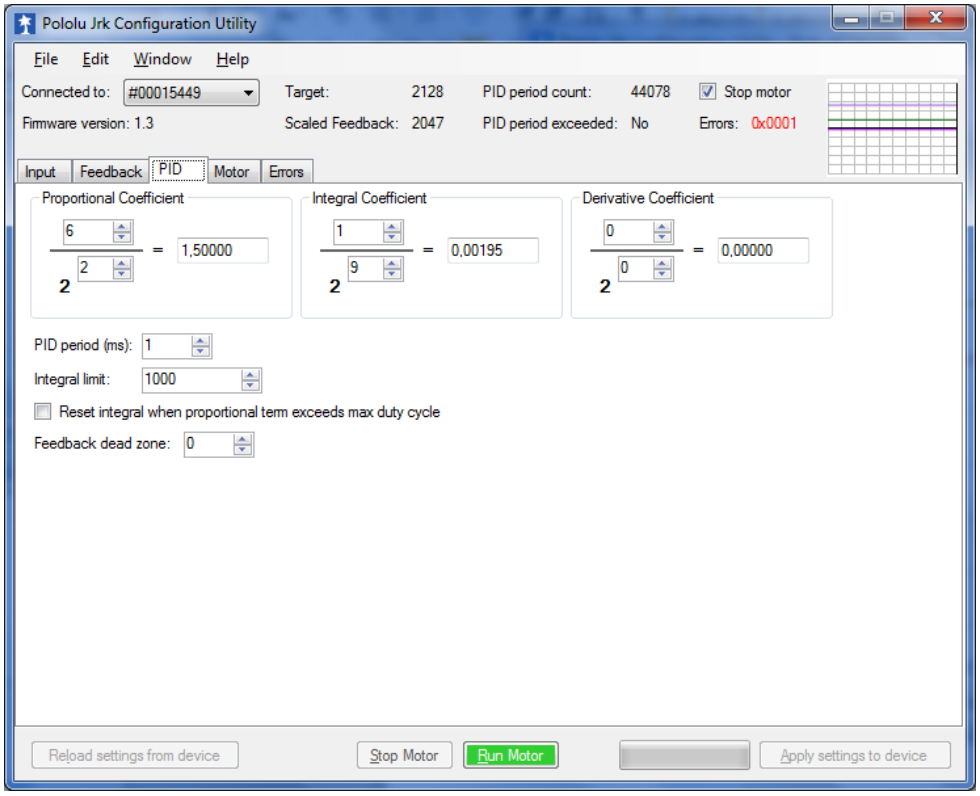


FIG. 4.2.3 Programa Jrk Configuration Utility opción PID.

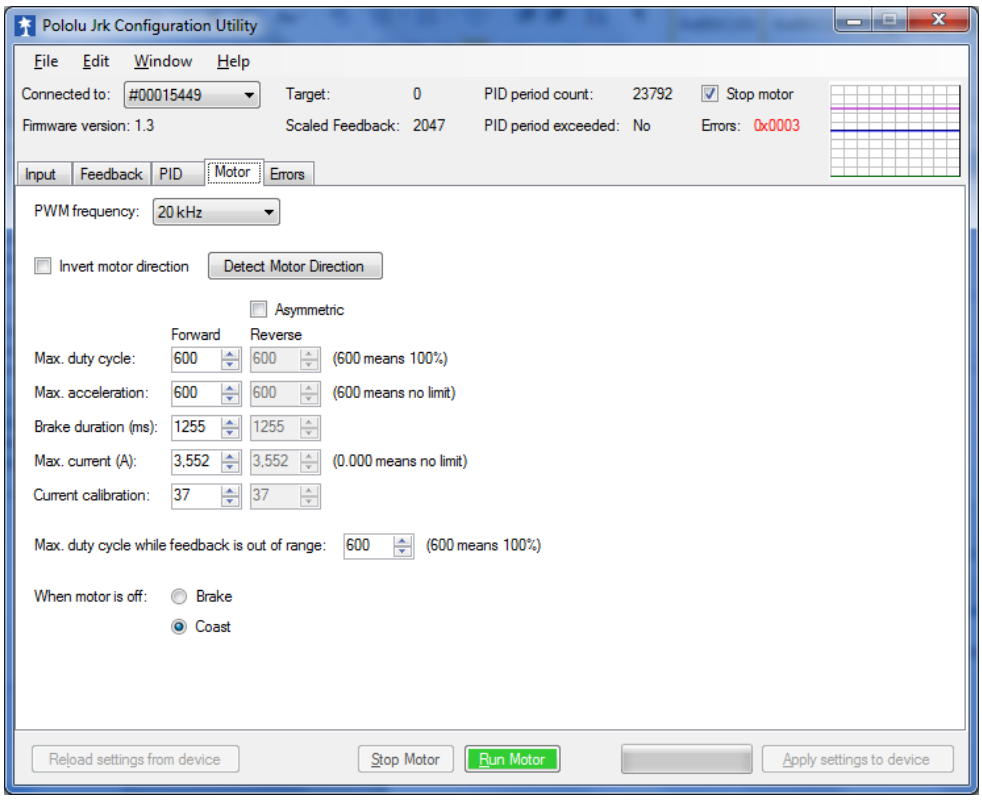


FIG. 4.2.4 Programa Jrk Configuration Utility opción motor.

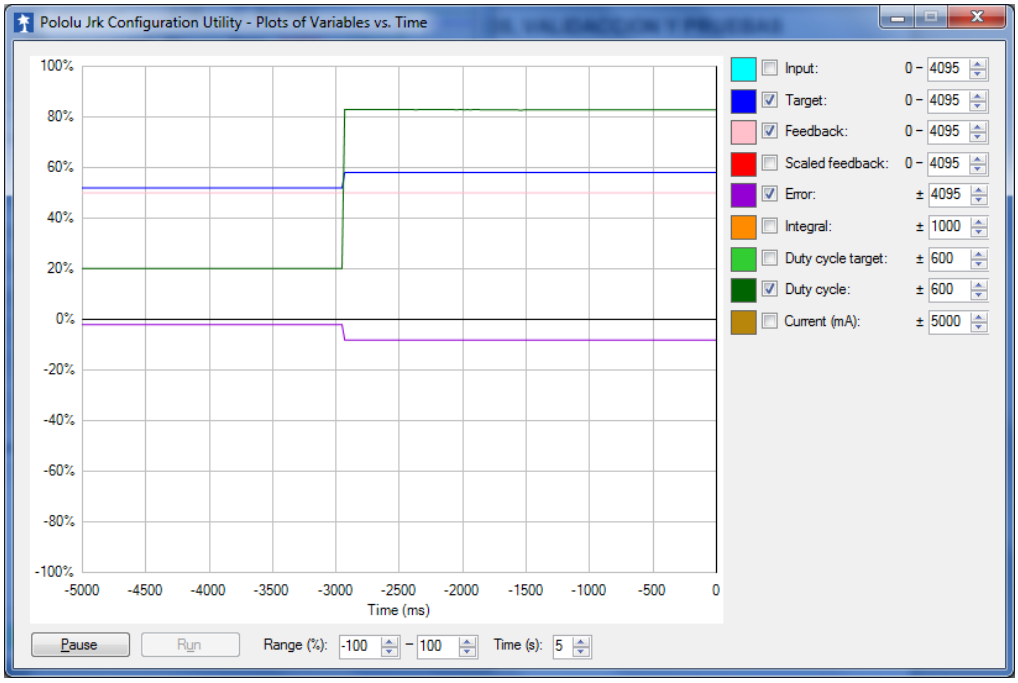


FIG. 4.2.5 Aumento de duty cycle del 20%a un poco mas del 80% en sentido horario.

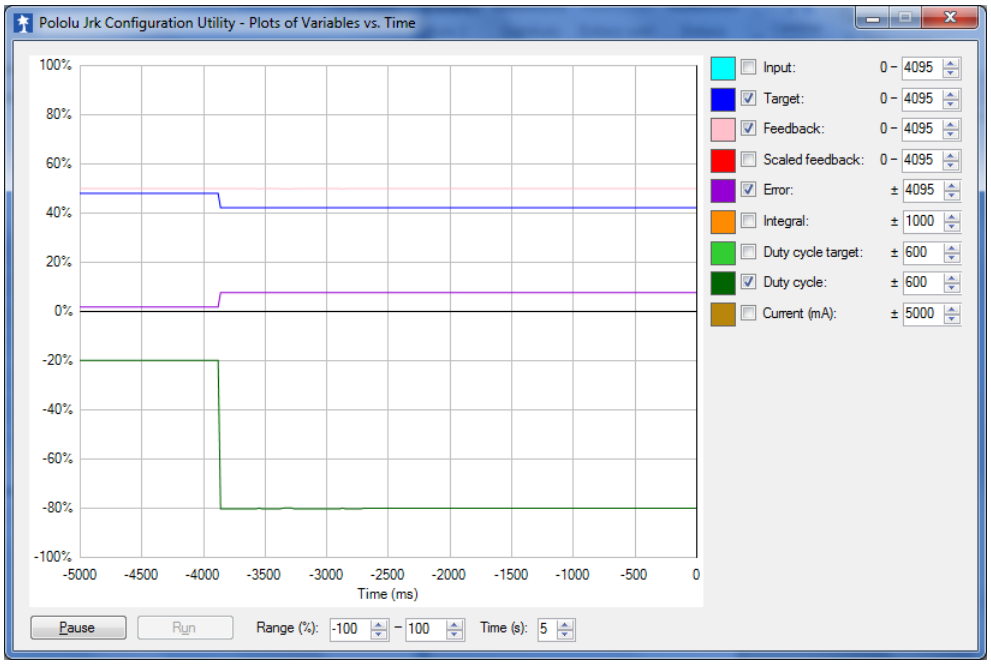


FIG. 4.2.6 Aumento de duty cycle del 20%a un poco mas del 80% en sentido Anti-horario.

4.3 Imágenes reales del proyecto

A continuación se presentan imágenes de todo el hardware funcionando y conectado debidamente:

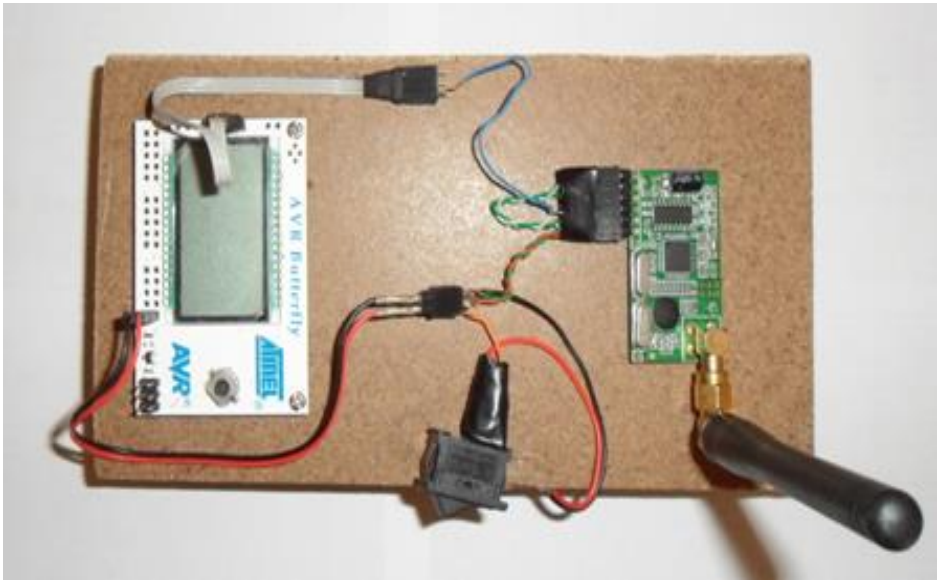


FIG. 4.3.1 AVR butterfly conectado al hm-tr 434 rs232.

FIG. 4.3.2 AVR butterfly mostrando en pantalla el nombre de uno de los integrantes.

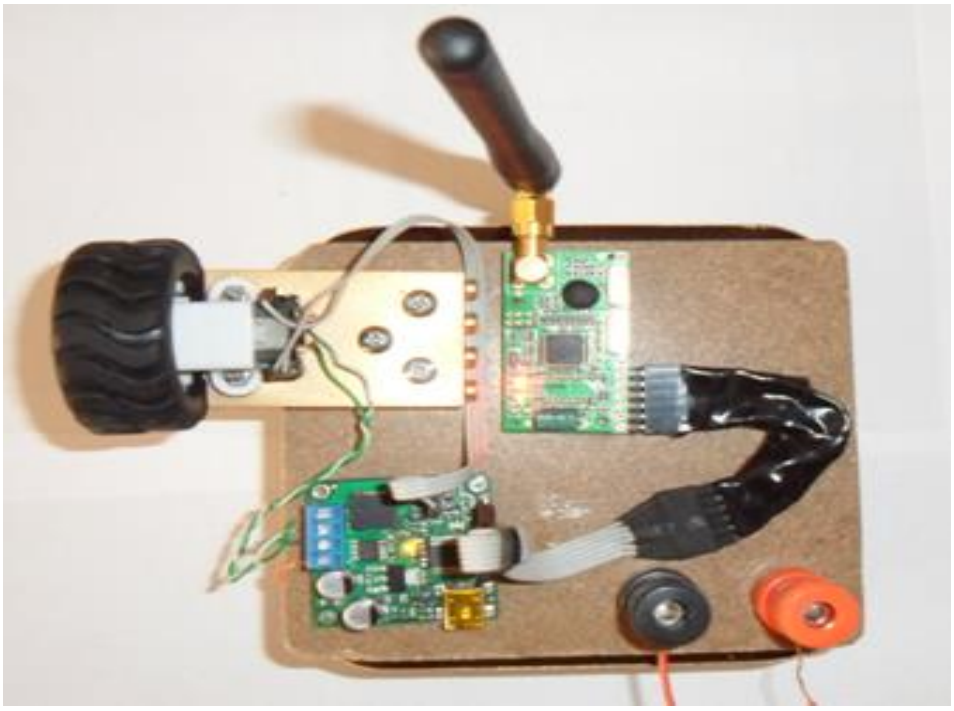


FIG. 4.3.3 hm-tr 434 ttl recibiendo la señal y enviándola al jrk21v3, el cual se conecta con el encoder y el motor.

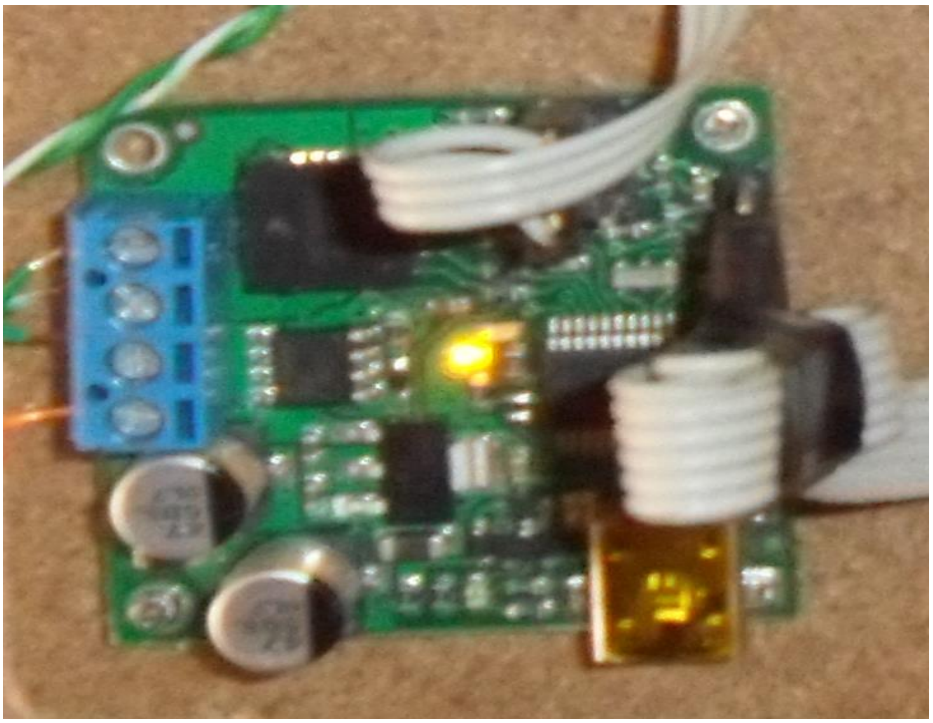


FIG. 4.3.4 Tarjeta de pololu jrk21v3 con led amarillo que representa un correcto funcionamiento.

CONCLUSIONES

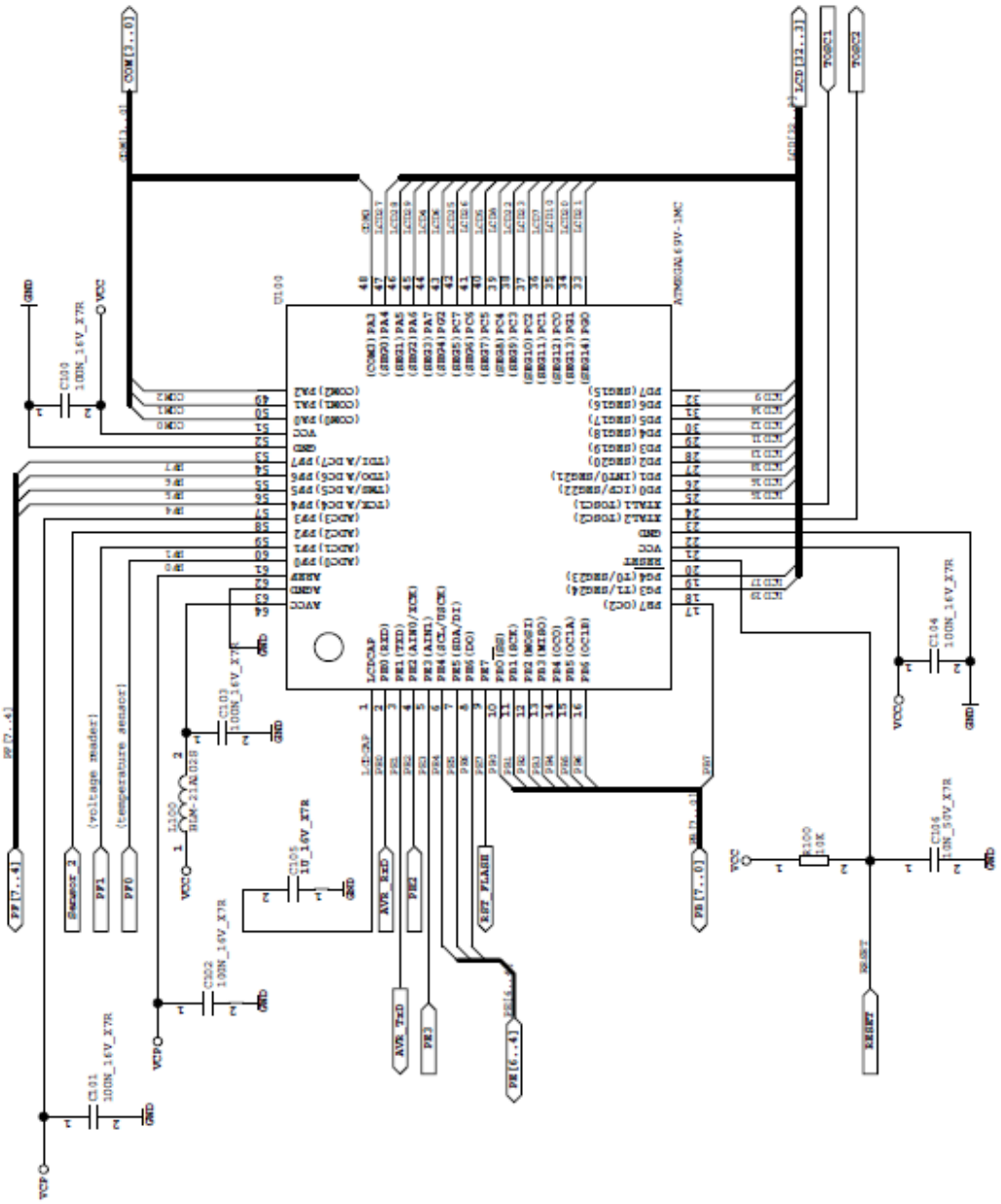
1. Hemos observado que la eficiencia de este proyecto va ligada a los parámetros de la planta, debido a que nuestro sistema es realimentado y se diseñó un controlador digital de tal forma que estabilice la velocidad del motor DC. La retroalimentación se implementó mediante el encoder, que censa el movimiento de la rueda, y envía una señal que es comparada con el valor de referencia dado las tramas de 8 bits enviadas como entrada.
2. Se puede apreciar, que la tarjeta JRK21v3 puede estar funcionando como regulador de velocidad cuando no se tiene el control PID, sin embargo, los límites impuestos al sistema con el programa Jrk Configuration Utility nos proporcionan una manera útil de prevenir daños en el sistema. El máximo ciclo de trabajo impuesto al configurarlo nos ayuda cuando la retroalimentación esta fuera del rango previsto, dado que limita los posibles daños al sistema al punto de estabilizarlo.
3. Hemos concluido que nuestro proyecto tiene varias aplicaciones desde hogareñas hasta con propósitos comerciales, siempre y cuando sean elementos que requieran un motor DC bajo las limitaciones de corriente, voltaje y otras que hemos impuesto, tomando como referencia las características observadas del kit jrk21v3, dado que un mal excediendo sus límites uso solo conllevará a dañarla.
4. Se puede notar, que una de las prioridades del trabajo realizado era conseguir una comunicación que no necesite un cableado directo entre los elementos en general y por esto utilizamos los módulos hm-tr 434 ttl/rs232. Nos vimos obligados a usar los 2 tipos de módulos puesto que el AVR butterfly nos envía una señal en comunicación rs232, pero la tarjeta de pololu jrk21v3 en su modo serial recibe una señal ttl lo cual fue uno de los parámetros importantes a considerar para alcanzar la comunicación inalámbrica deseada.

RECOMENDACIONES

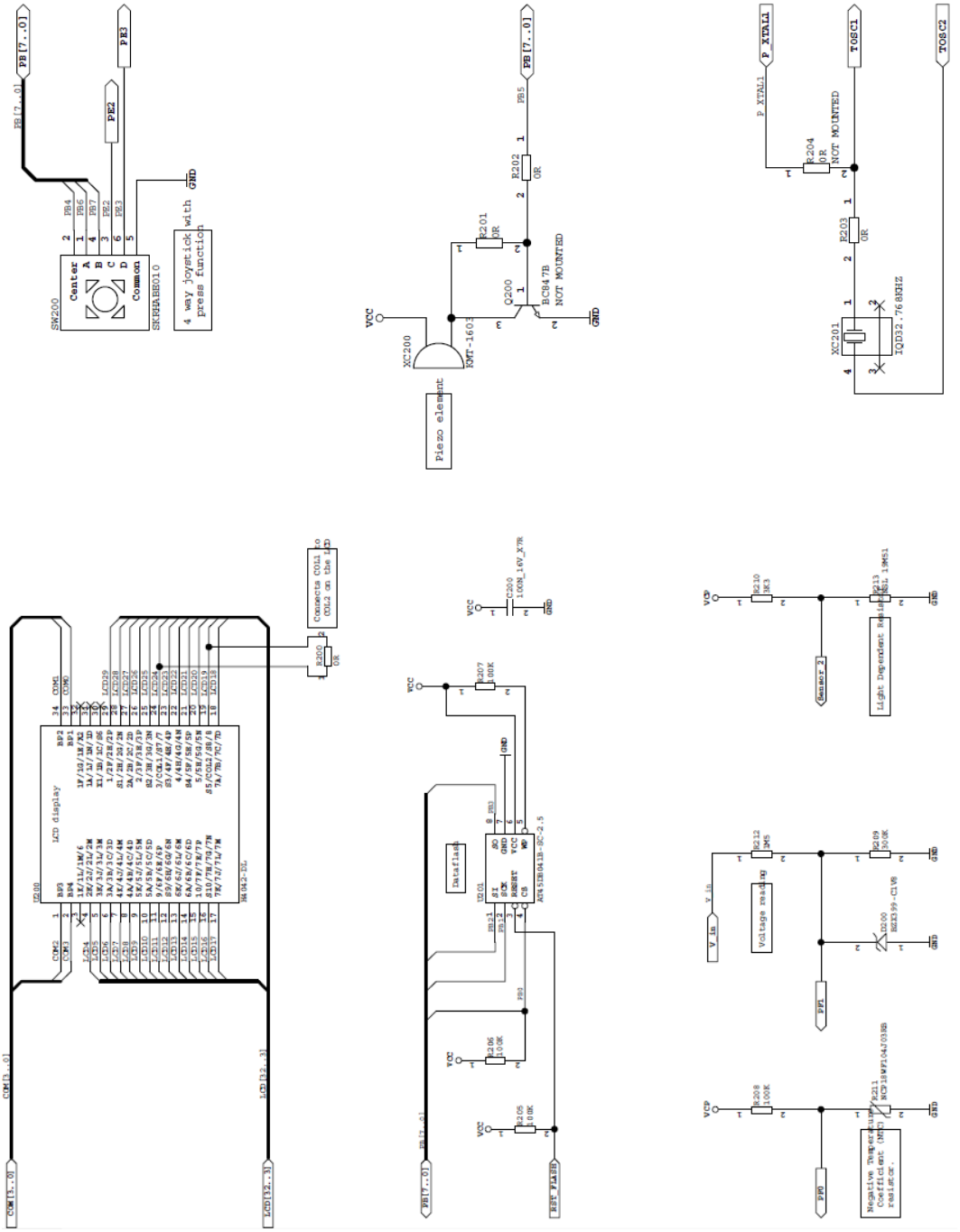
1. Se recomienda que se tome muy en cuenta la lectura y entendimiento de la guía de pololu con respecto al uso del jrk21v3, recordando que no puede ser programado y consta con parámetros fijos para su manejo. También nos presenta la opción de crear un controlador P, I, D lo cual es sumamente útil una vez entendido el manejo del mismo.
2. No utilizar dos hm-tr 434 rs232 para comunicar inalámbricamente ambos elementos, AVR butterfly y jrk21v3, puesto que la tarjeta correspondiente a la sección del controlador recibe señales de tipo ttl y es uno de los principales errores que pudimos apreciar al momento de trabajar en esta tesina de seminario.
3. Al momento de grabar el AVR butterfly usando el programador isp vía usb de pololu, PGM03A, se debe considerar que el mismo utiliza drivers propios que no se relacionan con cualquier otro elemento perteneciente a la misma familia y puede no ser reconocido, es recomendable descargarlos de la página de pololu.
4. Los programas utilizados en la configuración de los módulos inalámbricos y la tarjeta del controlador son considerados software libre por lo que se pueden descargar de la web sin ningún problema. Las funciones y librerías aplicadas en el atmega169 son consideras de igual manera software libre y pueden ser modificadas a gusto del usuario.

ANEXOS

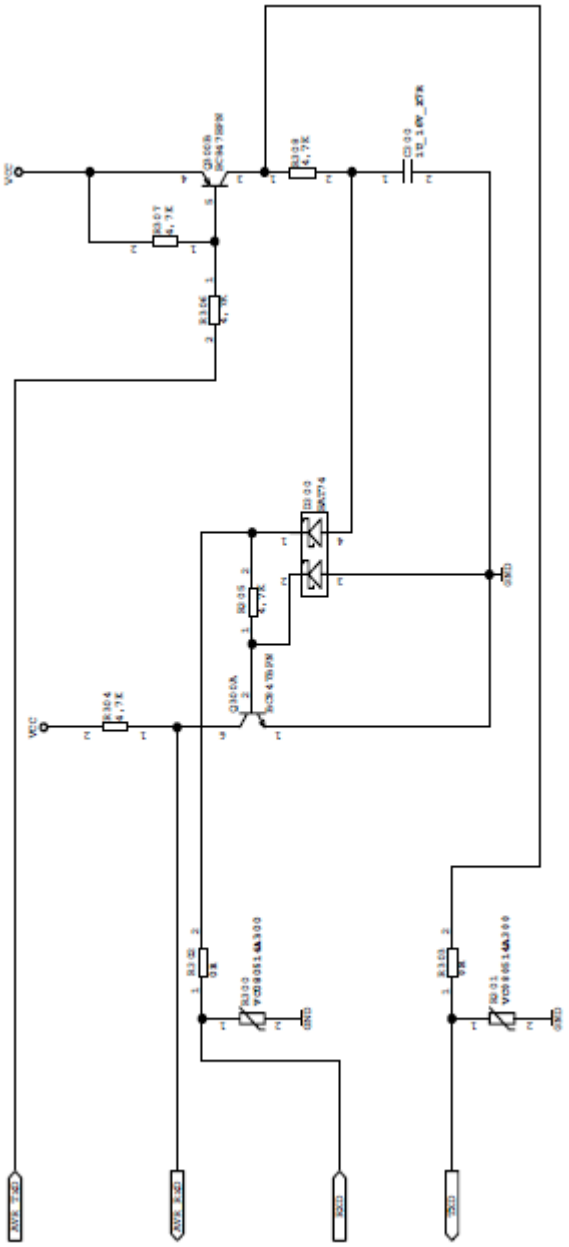
ESQUEMÁTICO AVR BUTTERFLY 1 DE 4



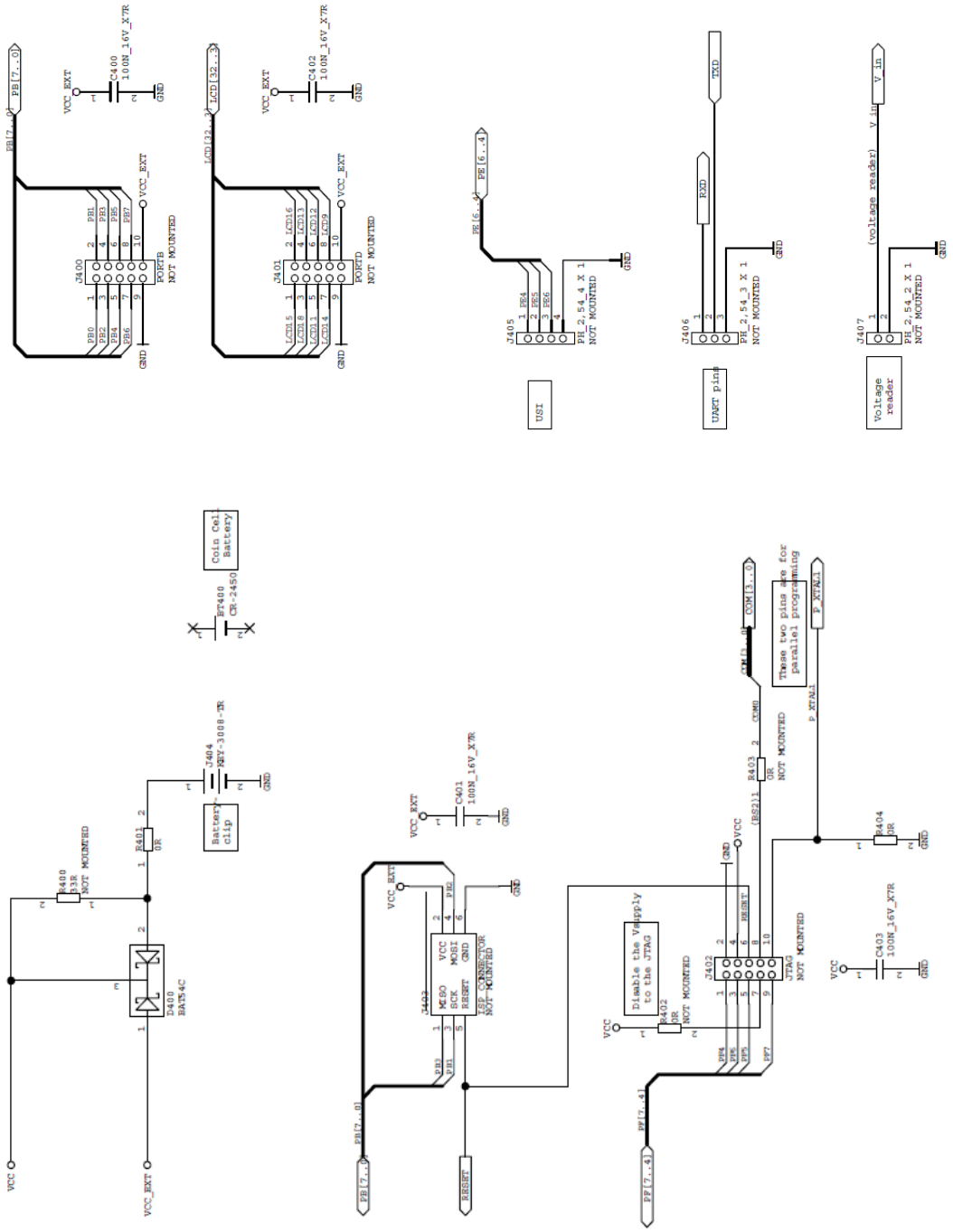
ESQUEMÁTICO AVR BUTTERFLY 2 DE 4



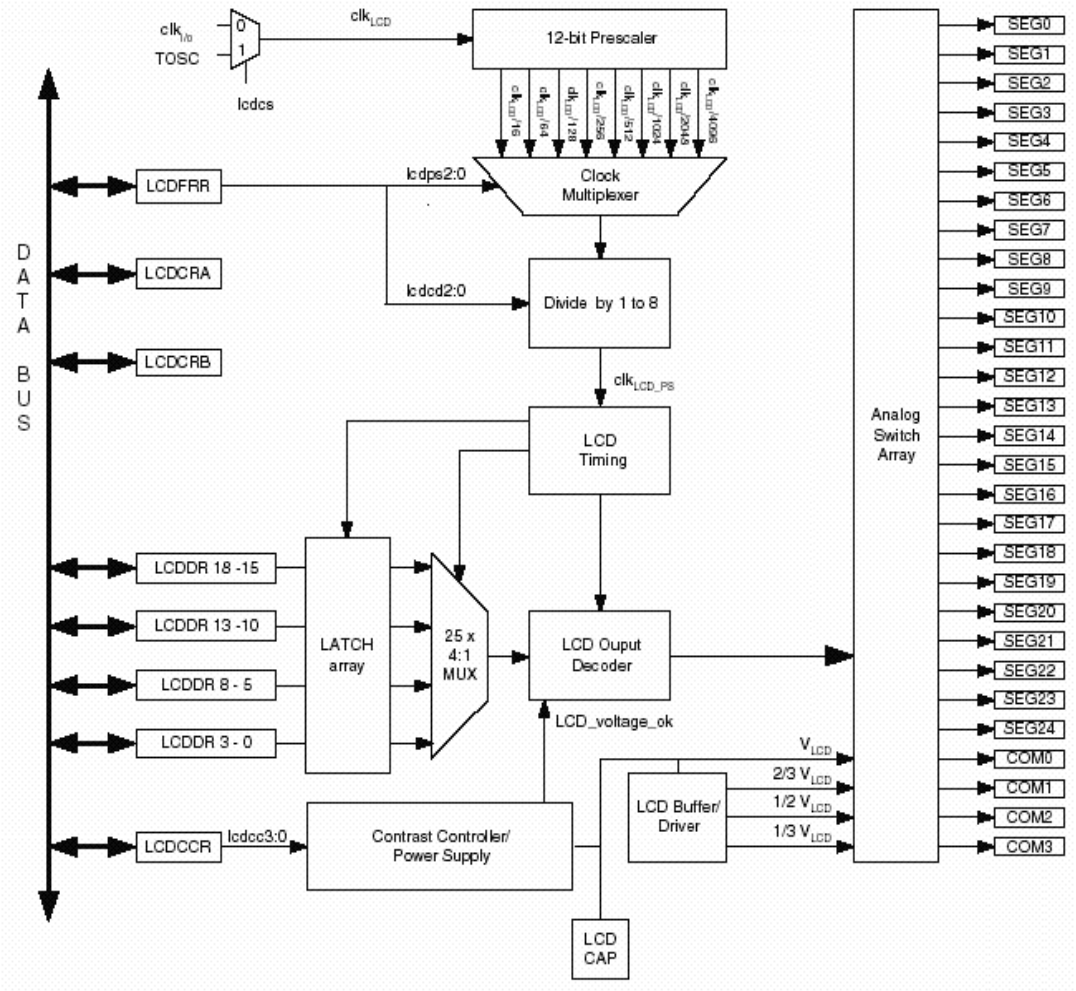
ESQUEMÁTICO AVR BUTTERFLY 3 DE 4



ESQUEMÁTICO AVR BUTTERFLY 4 DE 4



ESQUEMÁTICO LCD



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

ESPECÍFICACIONES USART ATMEGA169

USART

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

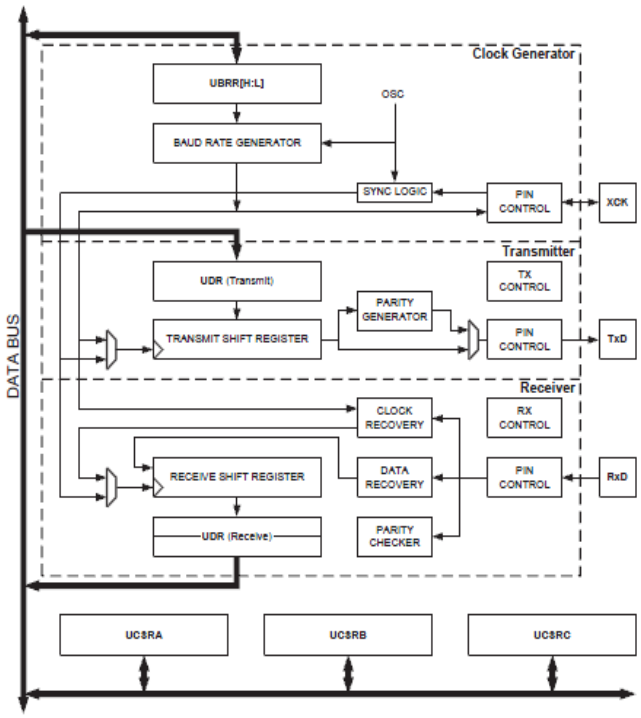
- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Master or Slave Clocked Synchronous Operation**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

The PRUSART0 bit in "Power Reduction Register - PRR" on page 34 must be written to zero to enable USART module.

Overview

A simplified block diagram of the USART Transmitter is shown in Figure 69. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 69. USART Block Diagram(1)



POSIBLES ERRORES TARJETA JRK21V3

4.f. Error Reporting Commands

There are several errors that can stop the jrk from driving its motor. The jrk's response to the different errors can be configured (**Section 3.f**).

Both of the error reporting commands result in a two-byte serial response form the jrk. Each bit in those two bytes represents a particular error. If the bit is 1, it means that the error occurred or is occurring. If we call the least-significant bit 0, and the first byte transmitted contains bits 0-7, then the correspondence between bits of the error bytes and errors are as follows:

- **Bit 0: Awaiting command**

If this bit is set, the jrk will not drive the motor until it receives a command that clears this bit. Any version of the Set Target command will clear the error bit. A Set Target command can be sent from the configuration utility, from a computer using the virtual Command Port (unless the jrk is configured to receive commands on RX), or from the RX line if the jrk is configured to receive commands on RX. This error occurs in Serial Input mode when the Jrk is powered on.

- **Bit 1: No power**

This error occurs when the jrk is connected to USB, but it detects no motor power connected to VIN and GND, so it can not drive the motor. If this error occurs, check your power supply and power connections.

- **Bit 2: Motor driver error**

This error occurs when one of the motor driver's fault conditions are triggered, and the motor driver shuts down the motor and reports the error to the jrk's microcontroller. This error also occurs when the jrk is connected to USB and motor power becomes disconnected. When this error occurs, the jrk will try to automatically recover from it by toggling the appropriate lines on the motor driver. The jrk 21v3's motor driver fault occurs on under-voltage, over-temperature, or over-current conditions. The jrk 12v12's motor driver fault occurs when it detects that motor output A is shorted to ground or VIN.

- **Bit 3: Input invalid** (Pulse Width Input Mode only)

In Pulse Width Input Mode, the jrk will only update the input value if it has received four good pulses in a row. For example, if the jrk receives five good pulses, a bad pulse, and then five more good pulses, it will update the input value after pulses 4, 5, 10, and 11. This error occurs if the jrk goes more than 120 ms without updating the input value. The jrk can recover from the error by receiving four good pulses in a row. This error does not occur in Analog Input Mode or Serial Input Mode.

- **Bit 4: Input disconnect**

This error occurs when the input is above the *Absolute maximum* or below the *Absolute minimum* (these parameters can be set in the configuration utility). Additionally, when using the *Detect disconnect with AUX*

option in Analog Input Mode, the jrk periodically tests to see whether the input potentiometer is disconnected and generates this error if it finds that it is (**Section 3.b**).

- **Bit 5: Feedback disconnect**

This error occurs when the feedback is above the *Absolute maximum* or below the *Absolute minimum* (these parameters can be set in the configuration utility). The absolute maximum and absolute minimum can be set using the configuration utility. Additionally, when using the *Detect disconnect with AUX* option in Analog Feedback Mode, the jrk periodically tests to see whether the feedback potentiometer is disconnected and generates this error if it finds that it is (**Section 3.c**).

- **Bit 6: Maximum current exceeded**

This error occurs when the motor current limit is exceeded. The limit can be set using the configuration utility (**Section 3.e**).

- **Bit 7: Serial signal error**

A hardware-level error that occurs when a byte's stop bit is not detected at the expected place. This can occur if you are communicating at a baud rate that differs from the jrk's baud rate.

- **Bit 8: Serial overrun**

A hardware-level error that occurs when the UART receive buffer is full. This error should not occur during normal operation.

- **Bit 9: Serial RX buffer full**

A firmware-level error that occurs when the firmware's buffer for bytes received on the RX line is full and a byte from RX has been lost as a result. This error should not occur during normal operation.

- **Bit 10: Serial CRC error**

This error occurs when the jrk is running in CRC-enabled mode and the cyclic redundancy check (CRC) byte at the end of the command packet does not match what the jrk has computed as that packet's CRC (**Section 4.d**). In such a case, the jrk ignores the command packet and generates a CRC error.

- **Bit 11: Serial protocol error**

This error occurs when the jrk receives an incorrectly formatted or nonsensical command packet. For example, if the command byte does not match a known command or an unfinished command packet is interrupted by another command packet, this error occurs.

- **Bit 12: Serial timeout error**

When the serial timeout is enabled (**Section 3.b**), this error occurs whenever the timeout period has elapsed without the jrk receiving any valid serial commands. This timeout error can be used to shut down the motors in the event that serial communication between the jrk and its controller is disrupted.

- **Bits 13-15: Reserved**

These bits do not represent any errors; they will always read as zeroes.

BIBLIOGRAFÍA

[1] Flowserve Corporation, Posicionador digital 2000 con control PID, http://www.flowserve.com/es_ES/Products/Valves/Positioners/Digital/2000-Digital-Positioner-with-PID-control,es_ES, fecha de consulta mayo 2011

[2] Danfoss, Ventiladores, http://www.danfoss.com/Spain/BusinessAreas/DrivesSolutions/Industries/Fan.htm, fecha de consulta mayo 2011

[3] Flowserve Corporation, Controlador DataFlo C Serie 17, http://www.flowserve.com/Products/Automation/Controller/17-Series-DataFlo-C-Controller,en_US, fecha de consulta mayo 2011

[4] Wikipedia, Proteus (electrónica), http://es.wikipedia.org/wiki/Proteus_%28electr%C3%B3nica%29, fecha de consulta mayo 2011

[5] Mercado libre, Modulo Inalambrico Uhf Hm-tr 434 Ttl/rs232 Transciver, http://articulo.mercadolibre.com.ec/MEC-8705880-_JM, fecha de consulta mayo 2011

[6] Pololu Corporation, Pololu Jrk USB Motor Controller User's Guide, http://www.pololu.com/docs/pdf/0J38/jrk_motor_controller.pdf, fecha de consulta mayo 2011, capítulos 2, 3 y 4

[7] Purdue University, AVR Simulation with the ATMEL AVR Studio 4, Purdue University, fecha de consulta mayo 2011

[8] Atmel Corporation, Atmega169, Atmel Corporation, fecha de consulta mayo 2011

[9] Pardue Joe, C Programming for Microcontrollers, Smiley Micros, fecha de consulta mayo 2011

[10] Pololu Corporation, Encoder for Pololu Wheel 42x19mm, http://www.pololu.com/catalog/product/1217, fecha de consulta mayo 2011