****



**FACULTAD INGENIERIA**

**de**

Ing. Sistemas Computacionales Hardware

**MICROPROCESADORES**

**Proyecto Final**

**Comunicación RS232 entre PC-PIC-PIC**

**ENRIQUE GERARDO HERNANDEZ VEGA**

**Elías Marín Chaparro 205043**

**Manuel Eduardo Navarrete Ramírez 181009**

**Alan Uriel García Calvillo 206989**

**OBJETIVO**

El objetivo del proyecto final, es establecer una comunicación entre el la PC y el PIC18f4550, y comunicación entre 2 PICS.

**MARCO TEÓRICO**

**Comunicación Serial**

Los sistemas microprogramables basados en CPU internamente están diseñados para la transferencia de datos en buses o líneas de 8 bits o múltiplos de 8. Así el bus de datos está optimizado para el tratamiento de datos en paralelo lo cual es mucho más rápido que el tratamiento serie.

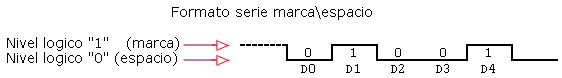
Si la velocidad de transferencia de datos en paralelo es mucho más rápida, ¿porqué se utiliza la transmisión de datos serie?. Algunas respuestas se dan a continuación:

1. Para realizar la comunicación de datos en paralelo se requiere gran cantidad de hilos conductores, pues debe ser establecido un hilo para cada bit de datos, además de las señales de control. Esto encarece notablemente la comunicación en función de la distancia. La comunicación serie requiere 2, 3 ó 4 hilos.
2. Una entrada salida/serie puede ser transmitida a través de pares de cobre, cable coaxial, fibra óptica, vía radio o vía satélite, lo que proporciona comunicación con equipos remotos (redes locales) o muy remotos (Internet a través de las redes telefónicas y de datos).
3. La comunicación paralelo no posee el alto grado de estandarización que ha alcanzado la comunicación serie, lo que permite la intercomunicación entre equipos, por ejemplo mediante RS232, USB o FireWire.

**Transmisión modulada en amplitud**

Dentro de las múltiples posibilidades existentes nos centraremos en la comunicación serie a través de la interpretación de dos niveles lógicos de tensión o corriente denominado *formato marca/espacio*.

El nivel lógico "1" representa un estado de tensión o corriente denominado marca, el nivel lógico "0" representa un estado de tensión o corriente denominado espacio.



Existen varios estándares que usan el formato marca/espacio, de los que nos interesan:

* TTL
* Lazo de corriente de 20mA
* RS232

Niveles para cada estándar.

|  |  |  |
| --- | --- | --- |
|  | Nivel Lógico "1" (Marca) | Nivel Lógico "0" (Espacio) |
| TTL | 5V | 0 |
| Lazo 20 mA | 20 mA | 0 mA |
| RS 232C | -3V a -15V | +3V a +15V |

**Consideraciones en la comunicación serie**

Cuando se transmite información a través de una línea serie es necesario utilizar un sistema de codificación que permita resolver los siguientes problemas :

1. **Sincronización de bits:** El receptor necesita saber donde comienza y donde termina cada bit en la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo (bit para señales binarias).
2. **Sincronización del carácter:** La información serie se transmite por definición bit a bit, pero la misma tiene sentido en palabras o bytes.
3. **Sincronización del mensaje:** Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, por ejemplo, detectar algún error en la comunicación de un mensaje.

**Velocidad de transmisión**

La velocidad de transmisión de datos es expresada en bits por segundo o baudios. El baudio es un concepto más general que bit por segundo. El primero queda definido como el número de estados de la señal por segundo, si sólo existe dos estados (que pueden ser representados por un bit, que identifica dos unidades de información) entonces baudio es equivalente a bit por segundo. Baudio y bit por segundo se diferencian cuando es necesario más de un bit para representar más de dos estados de la señal.

La velocidad de transmisión queda limitada por el ancho de banda, potencia de señal y ruido en el conductor de señal. La velocidad de transmisión queda básicamente establecida por el reloj. Su misión es examinar o muestrear continuamente la línea para detectar la presencia o ausencia de los niveles de señal ya predefinidos. El reloj sincroniza además todos los componentes internos.

**La base de reloj**

Cuando se establece la comunicación es necesario implementar una base de tiempo que controle la velocidad. En un microcontrolador, se utilizaría la base de tiempos del reloj del sistema, si bien, en términos genéricos se utilizaría uno de los siguientes métodos:

1. Mediante la división de la base de reloj del sistema. por ejemplo mediante un contador temporizador programable.
2. A través de un oscilador TTL. Para cambiar frecuencia hay que cambiar el cristal.
3. Generador de razón de baudios. Existen diferentes dispositivos especializados que generan diferentes frecuencias de reloj.

**Modos de transmisión**

Existen dos modos básicos para realizar la transmisión de datos y son:

* Modo asíncrono.
* Modo síncrono.

Las transmisiones asíncronas son aquellas en que los bits que constituyen el código de un caracter se emiten con la ayuda de impulsos suplementarios que permiten mantener en sincronismo los dos extremos.

En las transmisiones síncronas los caracteres se transmiten consecutivamente, no existiendo ni bit de inicio ni bit de parada entre los caracteres, estando dividida la corriente de caracteres en bloques, enviándose una secuencia de sincronización al inicio de cada bloque.

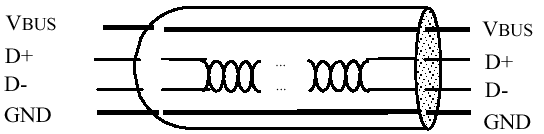
USB

Es un BUS de alta velocidad para conexión de periféricos tanto para MAC como Windows. Permite el intercambio de información entre un Host (PC) y los dispositivos simultáneamente.

Existen 2 tipos de velocidades:

Full-Speed 12 Mbps

Low-Speed 1.5 Mbps



**Alimentación**

Nominalmente +5V, aunque los dispositivos deben permitir fluctuaciones de voltaje de hasta 1 volt.

El Host (PC) puede alimentar hasta 100 mA, Un HUB hasta 500 mA.

**Endpoints**

Es una porción direccionable en un dispositivo USB, es una terminal de comunicación entre el Host y el dispositivo.

Cada dispositivo tiene hasta 16 endpoints, siendo obligatoriamente el endpoint 0 el de control.

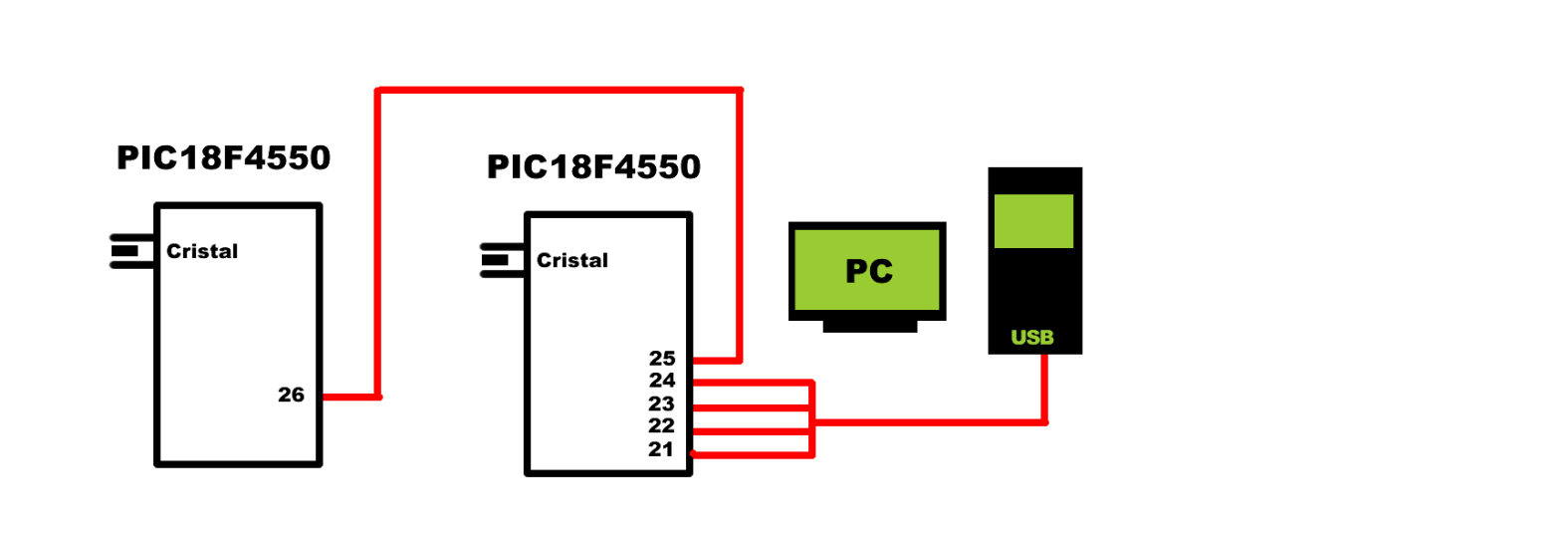
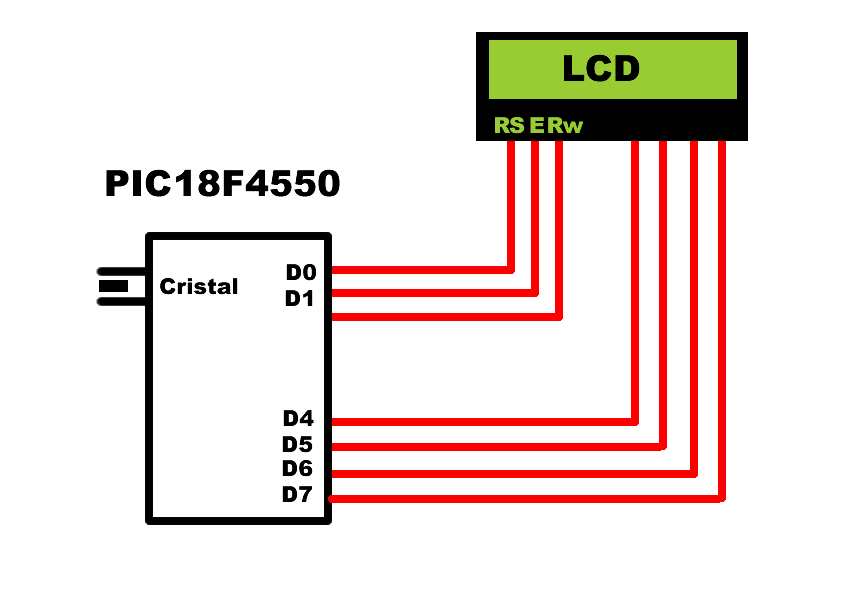
Cada endpoint tiene una sola dirección (4 bits).

**DESARROLLO**

Se utilizaron 3 componentes para establecer una comunicación entre ellos, Un PC utilizado como interfaz visual diseñado y programado mediante el lenguaje JAVA, Funcionando en la plataforma de escritorio con Sistema Operativo Windows XP. Mediante este primer componente se estableció una conexión de salida hacia un segundo componente utilizando el canal de comunicación USB(universal Serial Port), realizando una programación básica solamente para configurar la velocidad de transferencia de 9600 Baudios y realizando el debido manejo de los datos de salida, implementando la librería JPICUSB la cual se encarga del manejo del Puerto USB, En una pantalla lcd se mostrara el estado de un checkbox, 0 si no estaba seleccionado y un 1 si estaba seleccionado.

una vez programada la interfaz visual en el PC, el siguiente paso es Programar los PIC mediante el lenguaje C, como segundo componente del sistema se encuentra un PIC18F4550 lo llamaremos PICA, en este se codifico lo necesario para realizar el enlace de comunicación RS232, configurando en código la velocidad a 9600 Baudios, así como definir los Pines Correspondientes para el Receptor RX, y el Transmisor TX para así establecer el canal de comunicación para aceptar valores provenientes del PC, una vez obtenido el valor , se proceso, para así enviarlo hacia el tercer componente PICB Vía RS232, este programado de igual forma mediante el lenguaje C, este tercer componente requiere cumplir con la velocidad establecida de 9600 Baudios, pero en este caso solamente configurado el puerto RX para recibir información y desplegar el numero obtenido mediante LEDS integrados en la tablilla.

**DIAGRAMA BLOQUES**



**DIAGRAMA DE FLUJO**

**Comunicación Serial PIC-PIC**

Inicio

Incluimos los encabezados y librerías necesarias. Y establecemos las configuraciones del usb.

Reservamos el espacio para nuestro código de bootloader

Inicializamos nuestro lcd

En un cilco infinito, verificamos si el push botón fue presionado

En caso de que sea presionado imprimimos en pantalla del lcd el valor del contador.

El PIC A enviara el valor al PIC b mediante tx, y será recibido en rx mostrando el valor en los leds de la segunda placa.

**Comunicación PC-PIC Java**

Inicio

Importamos la librería JPIcusb y la inicializamos.

Cuando el checkbox sea activado o desactivado llamara a la función prender led.

Dependiendo de su estado se enviara un valor mediante la función send\_command

El método QWrite se encargara de enviar el dato a nuestro PIC mediante una conexión USB

**Comunicación PC-PIC en C**

Inicio

Incluimos los encabezados y librerías necesarias. Y establecemos las configuraciones del usb.

Reservamos el espacio para nuestro código de bootloader

Inicializamos nuestro lcd, también el usb mediante usb\_init();

Esperamos que el PIC sea configurado mediante usb\_wait\_for\_enumeration(); y habilitamos todas las interrupciones enable\_interrupts(global);

En ciclo infinito, se verifica que el usb este configurado, si asi es se verifica si se han recibido datos provenientes del PC con usb\_kbhit(1)

Si llegan datos con usb\_get\_packet(1, DatosBuffer, TamBuffer); cogemos el paquete de tamaño 32 bytes(TamBuffer) del EndPoint 1 y toma los dos bytes que llegan y los guarda en DatosBuffer,y luego son guardados en ComandoPC y ParametroPC respectivamente

Verificamos si el byte 0 (RecCommad) que llega es igual a TIPO\_COMANDO = 88

Si así es con printf(LCD\_PUTC,"%d",ParametroPC); imprimimos en el lcd el valor de ParametroPC del byte 1 (DatosBuffer)

**CONCLUSIONES**

Con la elaboración de este proyecto extendimos el conocimiento en la comunicación serial y usb del PIC 18f4550. Para la simulación en ISIS del programa PC-USB se recomienda que se haga en una plataforma de 32 bits, por la compatibilidad de los drivers de usb de Proteus.

**BIBLIOGRAFIA**

<http://perso.wanadoo.es/pictob/comserie.htm>

<http://www.edutecne.utn.edu.ar/microcontrol_congr/comunicaciones/JPICUS.PDF>

MANUAL KITPIC-USB4550.pdf

18f2550-4550.pdf

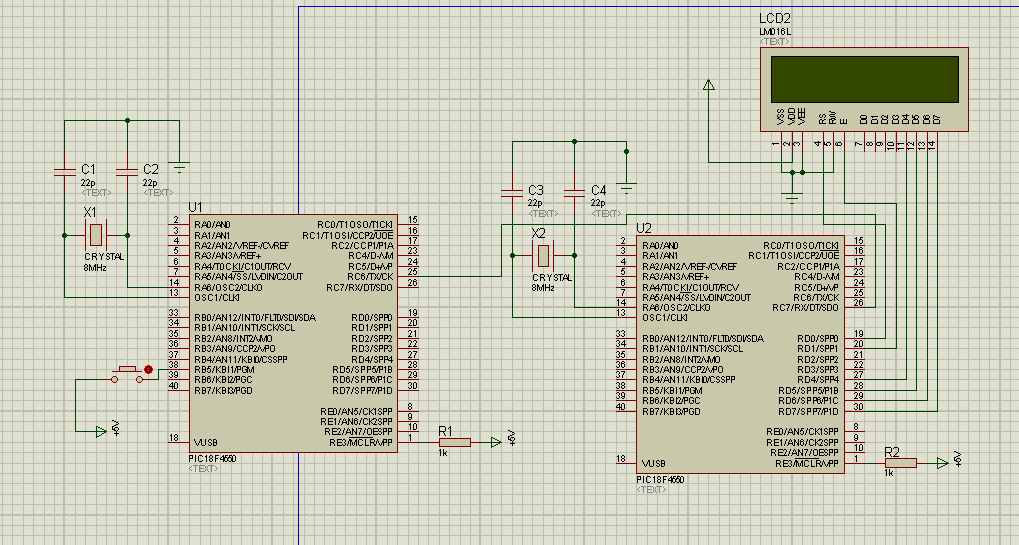
18F4550.pdf

el usb.ppt

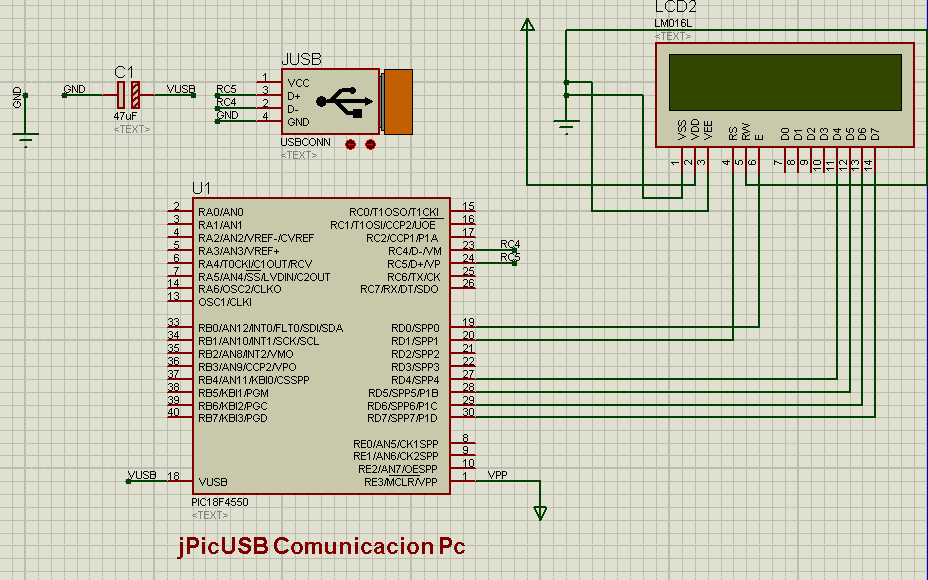
**ANEXOS**

**DIAGRAMA ELECTRICO**

**Comunicación Serial**



**Comunicación Pc-PIC**

****

**CODIGO JAVA PC-PIC**

**import jPicUsb.\*;**

**import java.util.logging.Level;**

**import java.util.logging.Logger;**

**public class Frame extends javax.swing.JFrame {**

**public byte cmd = 88;**

**public Frame() {**

**initComponents();**

**try {**

**iface.load();//inicializamos la libreria Jpicusb**

**} catch (Exception ex) {**

**Logger.getLogger(Frame.class.getName()).log(Level.SEVERE, null, ex);**

**}**

**}**

**public void send\_command(byte comando, byte parametro) {**

**byte[] salida = {comando, parametro};**

**iface.QWrite(salida, 2, 1000);**

**/\***

**QWrite (byte[ ] pData, int dwLen, long dwMilliseconds)**

**Escribe dwLen bytes del arreglo pData,**

**con un timeout de dwMilliseconds milisegundos y luego cierra la conexión.**

**Retorna la cantidad de bytes que se escribieron con éxito**

**\*/**

**}**

**private void encender\_led() {**

**byte Estado = 0x00;**

**if (led.isSelected()) {**

**Estado |= 0x01;**

**}**

**send\_command(cmd, Estado);**

**}**

**@SuppressWarnings("unchecked")**

**// <editor-fold defaultstate="collapsed" desc="Generated Code">**

**private void initComponents() {**

**led = new javax.swing.JCheckBox();**

**setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);**

**led.setText("Prender LED");**

**led.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**ledActionPerformed(evt);**

**}**

**});**

**javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());**

**getContentPane().setLayout(layout);**

**layout.setHorizontalGroup(**

**layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGroup(layout.createSequentialGroup()**

**.addGap(124, 124, 124)**

**.addComponent(led)**

**.addContainerGap(191, Short.MAX\_VALUE))**

**);**

**layout.setVerticalGroup(**

**layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGroup(layout.createSequentialGroup()**

**.addGap(109, 109, 109)**

**.addComponent(led)**

**.addContainerGap(168, Short.MAX\_VALUE))**

**);**

**pack();**

**}// </editor-fold>**

**private void ledActionPerformed(java.awt.event.ActionEvent evt) {**

**encender\_led();**

**}**

**public static void main(String args[]) {**

**java.awt.EventQueue.invokeLater(new Runnable() {**

**public void run() {**

**new Frame().setVisible(true);**

**}**

**});**

**}**

**private javax.swing.JCheckBox led;**

**}**

**CÓDIGO SERIAL PIC-PIC**

**ENVIO**

**#include "adclcd.h"**

**#include "LCD416.c"**

**#define LCD\_ENABLE\_PIN PIN\_D1**

**#define LCD\_RS\_PIN PIN\_D0**

**#fuses xt,nomclr,noprotect,nolvp**

**//#use delay(clock=4M)**

**#use rs232(uart1,baud=9600,xmit=PIN\_C6,bits=8,parity=N)**

**#use standard\_io(d)**

**#use fast\_io(b)**

**#use fast\_io(a)**

**#build(reset=0x800)**

**#build(interrupt=0x808)**

**#org 0x0000, 0x07ff void bootloader() {}**

**int boton, x ;**

**void main()**

**{**

**lcd\_init();**

**set\_uart\_speed(9600);**

**setup\_spi(SPI\_SS\_DISABLED);**

**setup\_wdt(WDT\_OFF);**

**setup\_timer\_0(RTCC\_INTERNAL);**

**setup\_timer\_1(T1\_DISABLED);**

**setup\_timer\_2(T2\_DISABLED,0,1);**

**setup\_ccp1(CCP\_OFF);**

**setup\_comparator(NC\_NC\_NC\_NC);**

**lcd\_gotoxy(1,1) ;**

**delay\_ms(1000);**

**x = 1;**

**while(true)**

**{**

**boton = input( PIN\_B5 );**

**if (boton==0){**

**putc(x);**

**printf(LCD\_PUTC,"%d",x);**

**delay\_ms(1000);**

**lcd\_gotoxy(1,1) ;**

**x++;**

**delay\_ms(2000);**

**}**

**}**

**RECIBE**

**#include "recibe.h"**

**int sre = 0;**

**//Variables para desplegar el dato recibido**

**#use rs232(baud=9600, xmit=pin\_c6,rcv=pin\_c7,parity=N)**

**//Directiva para el uso del puerto serie**

**#build(reset=0x800)**

**#build(interrupt=0x808)**

**#org 0x0000, 0x07ff void bootloader() {}**

**#int\_RDA**

**RDA\_isr() //Vector de subrutina de servicio**

**{ sre = getc ();**

**//Guarda el valor del dato recibido**

**return sre;**

**//Regresa el valor del dato recibido**

**}**

**void main()**

**{**

**setup\_adc\_ports(NO\_ANALOGS|VSS\_VDD);**

**//setup\_adc(ADC\_OFF|ADC\_TAD\_MUL\_0);**

**setup\_psp(PSP\_DISABLED);**

**setup\_spi(FALSE); setup\_wdt(WDT\_OFF);**

**setup\_timer\_0(RTCC\_INTERNAL);**

**setup\_timer\_1(T1\_DISABLED);**

**setup\_timer\_2(T2\_DISABLED,0,1);**

**setup\_comparator(NC\_NC\_NC\_NC);**

**setup\_vref(FALSE); enable\_interrupts(INT\_RDA); enable\_interrupts(GLOBAL);**

**setup\_oscillator(OSC\_8MHZ|OSC\_TIMER1|OSC\_31250|OSC\_PLL\_OFF);**

**set\_uart\_speed (9600); //Configura la velocidad de transferencia**

**set\_tris\_d (0x00); //Configura el puerto D como salida**

**printf("Conexion Exitosa"); //Envia una cadena de caracteres para comprobar conexion**

**while (1) {**

**output\_D(sre); // Se pone en 1 el bit 7 del puerto D para el control de los displays**

**delay\_ms(25); //Retardo de 25 mS**

**}**

**}**