Practica de Sistemas Embebidos

Menú

José Ángel Alonso Márquez

Carla Iveth Velasco Nieto

24 de septiembre de 2019

Reporte de práctica de Sistemas embebidos.

Introducción

En el siguiente reporte se detalla el funcionamiento de un menú el cual está programado en Arduino, se despliega en una LCD y es controlada por el usuario por medio de un encoder rotativo.

Al iniciar el menú aparece un mensaje que pide al usuario mantener el encoder presionado para poder entrar al menú. Una vez en el menú, podemos ver 4 listas con diferentes nombres: reloj, leds, sensores y listas; se seleccionan presionando una vez el encoder. Para retroceder a la opción anterior…

Para salir del menú, basta con dejar de presionar o mover el encoder, regresará al mensaje inicial.

Para realizar esta práctica utilizamos los siguientes componentes:

* LCD de 20x4
* Módulo I2C
* Jumpers
* Encoder rotativo
* Arduino Mega

Diagrama esquemático que muestra a grandes rasgos lo que es la práctica:

 



Códigos en Arduino:

código APP02: En este código se inicializan las variables para las listas y las opciones de cada menú.

#include <LiquidCrystal\_I2C.h>

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

//=============================================================================

//---[ BIBLIOTECAS y RECURSOS y AJUSTES de la APLICACIÓN ]---

// Aquí están: Definiciones, Bibliotecas, Recursos, Ajustes por Elemento

#include "App02.h"

//=============================================================================

//---[ ADMINISTRACIÓN del MENÚ de Opciones ]---

//--[ Estructura del árbol de SELECCIÓN, DESPLIEGUE y ACCIONES de opciones ]--

#include "Estructura.h"

//=============================================================================

// Definición del arreglo de LEDs (FastLED)

CRGB leds[NUM\_LEDS];

//---[ INSTANCIACIÓN del MENÚ global ]---

MD\_Menu M(navigation, display, // NAVEGACIÓN y DESPLIEGUE del usuario.

mnuHdr, ARRAY\_SIZE(mnuHdr), // Encabezado de Datos del MENÚ

// (menu Header data)

mnuItm, ARRAY\_SIZE(mnuItm), // Elementos de Datos del MENÚ

// (menu Item data)

mnuInp, ARRAY\_SIZE(mnuInp)); // Entradas de Datos del MENÚ

// (menu Input data)

//=============================================================================

// PREGUNTAR si están [DES]Habilitados

//-----[ VARIABLES de la APLICACIÓN ]-----

int8\_t EdoLed = LOW; // Estado inicial del LED

int8\_t EdoInicial = LOW; // Estado inicial del Temporizado

// -----

unsigned long previoMillis = 0; // valor inicial del "Tick" interno.

unsigned long actualMillis; // Valor de "Ticks" del sistema.

// -----

int16\_t Timer0 = 2500; // Valores para Temporizadores (milisegundos).

int16\_t Timer1 = 150; //

int16\_t Timer2 = 750; //

int16\_t Timer3 = 150; //

// -----

//---[ VARIABLES GLOBALES que se MANIPULAN y CONSULTAN desde el MENÚ]---

uint16\_t dummy; // Auxiliar para el manejo del MENÚ.

bool bValue = true; // Valor referencia en "Callback" de funciones.

MD\_Menu::value\_t vBuf; // Buffer de interface para valores del menú.

// Variables para ÍNDICES de LISTAS

uint8\_t IndLista1 = 0; // Índice en Lista 1

uint8\_t IndLista2 = 1; // Índice en Lista 2

uint8\_t IndLista3 = 2; // Índice en Lista 3

uint8\_t IndLista4 = 3; // Índice en Lista 4

uint8\_t IndLista5 = 4; // Índice en Lista 5

uint8\_t IndLista6 = 5; // Índice en Lista 6

uint8\_t IndLista7 = 6; // Índice en Lista 7

uint8\_t IndLista8 = 7; // Índice en Lista 8

uint8\_t IndLista9 = 8; // Índice en Lista 9

uint8\_t IndListaA = 9; // Índice en Lista A

// VARIABLES para el RTC:

int8\_t Horas = 12; // Para el RTC:

int8\_t Minutos = 30; //

int8\_t Segundos = 30; //

int8\_t DiaSem = 4; // 1 = Domingo, 2 = Lunes, Etc.

int8\_t DiaMes = 15; //

int8\_t Mes = 6; // 1 = Enero, 2 = Febrero, Etc.

int16\_t Ayo = 2018; //

// VARIABLES para DATOS empleados en la APLICACIÓN:

uint8\_t Contador1 = 0; // Contador para algún evento

uint8\_t Distancia = 0; // Contador para algún evento

uint8\_t Intensidad = 0; // Contador para algún evento

uint8\_t Iluminacion = 0; // Contador para algún evento

uint8\_t Matiz = 0; // Contador para algún evento

uint8\_t Sombreado = 0; // Contador para algún evento

uint8\_t Tonalidad = 0; // Contador para algún evento

uint8\_t Transparencia = 0; // Contador para algún evento

uint8\_t Radio = 0; // Contador para algún evento

uint8\_t Contador2 = 0; // Contador para algún evento

//---[ VARIABLES GLOBALES que se UTILIZAN en la APLICACIÓN ]---

//---[ RTC ]---

volatile boolean bShowTime = false;

//==============================================================================

//-----[ CÓDIGO de APOYO, FUNCIONES, SUBRUTINAS, MACROS ]-----

//==============================================================================

//---[ CONVIERTE números DECIMALES a BINARIO CODIFICADO en DECIMAL ]---

byte decToBcd(byte val) {

return ( (val / 10 \* 16) + (val % 10) );

}

//---[ CONVIERTE BINARIOS CODIFICADOS en DECIMAL a DECIMALES normales ]---

byte bcdToDec(byte val) {

return ( (val / 16 \* 10) + (val % 16) );

}

//---[IMPRIME 2 DÍGITOS, anteponiendo un "0" si es necesario ]---

void p2dig(uint8\_t v) {

if (v < 10) lcd.print("0");

lcd.print(v);

}

//---[ Día de Semana a Cadena [DOW 1=Dom, 2=Lun, 3=Mar, Etc., menos "0"] ]---

const char \*dow2String(uint8\_t code) {

static const char \*str[] = {"---", "Dom", "Lun", "Mar", "Mie", "Jue", "Vie",

"Sab"};

return (str[code]);

}

//--- [ IMPRIME la FECHA y el TIEMPO actual en una pantalla LCD ]---

void printTime(void) {

lcd.setCursor(0, 0);

lcd.print("-> ");

lcd.print(dow2String(RTC.dow)); // Día de la Semana

lcd.print(" ");

p2dig(RTC.dd); // Día del Mes

lcd.print("-");

p2dig(RTC.mm); // Mes

lcd.print("-");

lcd.print(RTC.yyyy); // Año

lcd.setCursor(0, 1);

lcd.print("-> ");

p2dig(RTC.h); // Hora

lcd.print(":");

p2dig(RTC.m); // Minutos

lcd.print(":");

p2dig(RTC.s); // Segundos

RTC.control(DS3231\_12H, DS3231\_OFF); // Para usar formato de 24 horas.

if (RTC.status(DS3231\_12H) == DS3231\_ON)

lcd.print(RTC.pm ? " pm" : " am");

}

//---[ Actualiza el desplegado del RTC ]---

void displayUpdate(void) {

RTC.readTime();

printTime();

}

//-----

#if USE\_POLLED\_CB || USE\_INTERRUPT

// Función de retorno ("callback") y de Interrupción (funciona igual).

void alarmICB(void) {

bShowTime = true; // Ajusta la bandera para actualizar.

}

#endif

//==============================================================================

//-----[ VERIFICACIÓN de que se INVOCA el MENÚ ]-----

void necesitaMenu(void) {

if (navigation(dummy) == MD\_Menu::NAV\_SEL) {

M.runMenu(true);

}

}

//==============================================================================

// Setup()

//==============================================================================

void setup(void) {

// Ensayo de FastLED:

/\*#if fastLED

FastLED.addLeds<NEOPIXEL, DATA\_PIN>(leds, NUM\_LEDS);

for (int i = 0; i < NUM\_LEDS; i++) { // Limpieza de la tira de LEDs.

leds[i] = CRGB::Black;

}

// Parpadea Amarillo-Verde

for (int i = 0; i <= 3; i++) {

leds[0] = CRGB::Yellow; FastLED.show(); delay(300);

leds[0] = CRGB::Green; FastLED.show(); delay(100);

}

leds[0] = CRGB::Purple;

FastLED.show();

· #endif

\*/

//---[ Para uso GENERAL ]---

lcd.init(); // Para poder emplear la LCD en la aplicación LOCAL.

lcd.backlight(); // Encender el LED de fondo.

lcd.clear(); // Limpiar la pantalla.

lcd.setCursor(0, 1);

lcd.print("-> APPLICACION <-- ");

pinMode(LED\_BUILTIN, OUTPUT);

//--- Para uso de la APLICACIÓN

pinMode(Timer0Pin, OUTPUT); // Ajuste de terminales para relevadores.

pinMode(Timer1Pin, OUTPUT); // con Timers

pinMode(Timer2Pin, OUTPUT); //

pinMode(Timer3Pin, OUTPUT); //

//--- Para el manejo del MENÚ

display(MD\_Menu::DISP\_INIT); // Ajuste para DESPLIEGUE del MENÚ.

setupNav(); // Ajuste de NAVEGACIÓN a través del MENÚ.

M.begin(); // Inicialización del Objeto del MENÚ.

M.setMenuWrap(true); //

M.setAutoStart(AUTO\_START); // Ajuste del INICIO por OMISIÓN

M.setTimeout(MENU\_TIMEOUT); // Tiempo de expiración para accionar el MENÚ.

//--- [Para el USO del RTC ]---

//---[ AJUSTES del MEDIOAMBIENTE de la ALARMA1 del RTC ] ---

#if USE\_POLLED

// Nada más que hacer

#endif

#if USE\_POLLED\_CB

RTC.setAlarm1Callback(alarmICB);

#endif

#if USE\_INTERRUPT

// Ajusta el hardware en la tarjeta Arduino.

pinMode(PIN\_INTERRUPT, INPUT\_PULLUP);

attachInterrupt(0, alarmICB, FALLING);

// Ajusta los registros de interrupción del RTC.

RTC.control(DS3231\_A2\_INT\_ENABLE, DS3231\_ON);

RTC.control(DS3231\_A1\_INT\_ENABLE, DS3231\_ON);

RTC.control(DS3231\_INT\_ENABLE, DS3231\_ON);

// "Apaga" la bandera de la Alarma1, en caso de que estuviera "Puesta"

RTC.control(DS3231\_A1\_FLAG, DS3231\_OFF);

#endif

// Ahora se incializa la Alarma1 para actualizar la pantalla cada segundo.

RTC.setAlarm1Type(DS3231\_ALM\_SEC);

}

//==============================================================================

// Loop()

//==============================================================================

void loop(void) {

static bool prevMenuRun = true;

static int Contador;

// Detecta si se necesita iniciar la ejecución normal del código del usuario

// ¿NO se está EJECUTANDO el MENÚ?

if (!M.isInMenu()) {

// --> INICIA el código NORMAL del USUARIO <--

#if USE\_POLLED

bShowTime = RTC.checkAlarm1();

#endif

#if USE\_POLLED\_CB

RTC.checkAlarm1();

#endif

// Si está "Puesta", actualiza pantalla y "Apaga" la bandera de muestra.

if (bShowTime) {

#if USE\_INTERRUPT

RTC.control(DS3231\_A1\_FLAG, DS3231\_OFF); // "Apaga" bandera de "Alarma1".

#endif

displayUpdate();

}

bShowTime = false;

necesitaMenu(); // Verifica si se ha invocado el MENÚ.

// --> CÓDIGO en PRUEBA

// Secuencia temporizado: Timer0 --> Timer1 --> Timer0 --> Timer1 --> Timer1

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-----------|\_\_\_\_\_\_\_\_\_\_|----------|\_\_\_\_\_\_\_\_\_\_1-------

// Se desconoce el estado inicial del LED que se prende / apaga.

actualMillis = millis();

EdoInicial = HIGH; // Corresponde al estado que adopta la salida por Timer0.

Timer0 = 500;

Timer1 = 500;

if (EdoLed == EdoInicial) {

if (actualMillis - previoMillis > Timer0) {

previoMillis = actualMillis;

EdoLed = !EdoInicial;

}

} else {

if (actualMillis - previoMillis > Timer1) {

previoMillis = actualMillis;

EdoLed = EdoInicial;

}

}

digitalWrite(LED\_BUILTIN, EdoLed);

// CÓDIGO en PRUEBA <---

// ¡OJO! --> DEBE sostenerse el pulsador de CNTL para entrar al MENÚ,

// al menos uos tres segundos, ya luego funciona normalmente <-- ¡OJO!

lcd.setCursor(0, 0);

lcd.print("MultiMenu");

lcd.setCursor(0, 1);

lcd.print("Press any key..");

//lcd.print(Matiz, HEX);

// --> CONCLUYE el código NORMAL del USUARIO <--

necesitaMenu(); // Verifica si se ha invocado el MENÚ.

}

// Revisa si existe una condición para que se active la selección del MENÚ

if (!M.isInMenu() && !AUTO\_START){

necesitaMenu();

}

// Se debe ejecutar de manera normal, ¡Siempre!

M.runMenu();

}

//==============================================================================

Código APP02.h:

Se inicializan las librerías de Arduino que se ocuparán, además hay una serie de opciones en las que seleccionamos el Arduino y las librerías que ocuparemos dependiendo de lo que queremos que haga nuestro menú; esto lo hacemos colocando unos y ceros, uno para activar y cero para desactivar.

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

// Archivo de cabecera para Aplicación Genérica [-----]

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

#pragma once

//==============================================================================

// Resguardo para cuando se invoque más de una vez

#ifndef App02.h

#define App02.h

//==============================================================================

//-----[ ARCHIVOS OBLIGATORIOS ]-----

#include <stdint.h> // Para poder usar diversos tipos de datos

//-----[ PLATAFORMAS de ARDUINO ]-----

#define ARDUINO\_UNO 0

#define ARDUINO\_MEGA 1

#define ARDUINO\_NANO 0

#define ARDUINO\_PRO 0

#define ARDUINO\_OTRO 0

//==============================================================================

//-----[ ELEMENTOS {DES}HABILITADOS para la APLICACIÓN ]-----

#define USA\_MENU 1 // Emplea "MD\_Menu" para interacción con usuario

#define TIMERS 1 // Para Temporizados en la ejecución

#define PAROLA 0 // Despliegue en Matrices de LEDs

#define LCD\_I2C 1 // Pantalla de Cristal Líquido (20x4)

#define TFT 0 // Pantalla TFT, Paralelo, IL19486 /

#define fastLED 1 // Tira de LEDs WS2812

#define SONIDO 0 // Bocina para Avisos

#define I2C\_PCF8254 0 // Para expansión de I/O

#define SPI\_AD 0 // ¿?

#define DS3231 1 // Para RTC

#define XYZ 0 // Sensor de Distancia

#define XYZ1 0 // Sensor de Temperatura

#define XYZ2 0 // Sensor de Presencia / Ausencia

#define XYZ3 0 // Sensor de Pulso Cardíaco

#define OTRO\_RECURSO 0 // Algo que se añadiera después

//==============================================================================

//-----[ DISPOSITIVOS de DESPLIEGUE / NAVEGACIÓN ]-----

// HARDWARE para el manejo del MENÚ (Elemento de SALIDA - Despliegue):

#define DISPLAY\_SERIAL 0 // Monitor Serial (Pruebas)

#define DISPLAY\_LCDSHIELD 1 // LCD con / sin I2C

#define DISPLAY\_PAROLA 0 // Tira(s) de Matrices de LEDs

#define DISPLAY\_TFT 0 // Pantalla de TFT, 2.4" o 3.5"

//------------------------------------------------------------------------------

// HARDWARE para el manejo del MENÚ (Elemento de ENTRADA - Datos):

#define INPUT\_SWITCH 0 // Interruptores mecánicos discretos

#define INPUT\_TACTIL 0 // Interruptores táctiles (capacitivos)

#define INPUT\_LCDSWITCH 0 // Interruptores en terminal Analógica

#define INPUT\_RENCODER 1 // Encodificador rotativo, Interruptor presión

#define INPUT\_TOUCH 0 // Entrada por "Touch Screen"

//==============================================================================

//-----[ CONSTANTES que identifican RECURSOS de la APLICACIÓN ]-----

//-----[Generales]-----

const uint32\_t BAUD\_RATE = 115200; // Velocidad del monitor Serial

//-----[ ESPECÍFICAS por Tipo de Tarjeta ]-----

//.....

//==============================================================================

//-----[ BIBLIOTECAS y RECURSOS de ELEMENTOS de la APLICACIÓN ]-----

//..............................................................................

#if USA\_MENU

#include <MD\_Menu.h>

const uint16\_t MENU\_TIMEOUT = 5000; // En millisegundos

const bool AUTO\_START = true; // Auto inicio del MENÚ, si está en "falso"

// EMPLEAR detección e inicio manual.

#endif

//..............................................................................

#if TIMERS

#if ARDUINO\_UNO

const uint8\_t Timer0Pin = 6; // Terminal salida del TIMER0 [¡VERIFICAR!]

const uint8\_t Timer1Pin = 7; // Terminal salida del TIMER1

const uint8\_t Timer2Pin = 8; // Terminal salida del TIMER2

const uint8\_t Timer3Pin = 9; // Terminal salida del TIMER3

#endif

#if ARDUINO\_MEGA

const uint8\_t Timer0Pin = 6; // Terminal salida del TIMER0 [¡VERIFICAR!]

const uint8\_t Timer1Pin = 7; // Terminal salida del TIMER1

const uint8\_t Timer2Pin = 8; // Terminal salida del TIMER2

const uint8\_t Timer3Pin = 9; // Terminal salida del TIMER3

#endif

#if ARDUINO\_NANO

const uint8\_t Timer0Pin = 6; // Terminal salida del TIMER0 [¡VERIFICAR!]

const uint8\_t Timer1Pin = 7; // Terminal salida del TIMER1

const uint8\_t Timer2Pin = 8; // Terminal salida del TIMER2

const uint8\_t Timer3Pin = 9; // Terminal salida del TIMER3

#endif

#endif

//..............................................................................

#if PAROLA

// ¿#include <MD\_MAX72XX.h>?

#include <MD\_MAXPanel.h>

#include <MD\_Parola.h>

#include <SPI.h>

#endif

//..............................................................................

#if LCD\_I2C

// Definiciones LCD que se empleará en esta aplicación, con comunicación I2C:

#define LCD\_ROWS 2

#define LCD\_COLS 16

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

static LiquidCrystal\_I2C lcd(0x27, LCD\_COLS, LCD\_ROWS); // Otra dirección: 0x27

#endif

//..............................................................................

#if TFT

#include <Adafruit\_GFX.h> // Librería para gráficas

#include <MCUFRIEND\_kbv.h> // Librería para la pantalla

#endif

//..............................................................................

#if fastLED

#include <FastLED.h>

#define NUM\_LEDS 30

#if ARDUINO\_UNO

#define DATA\_PIN 6

#elif ARDUINO\_MEGA

#define DATA\_PIN 33 // ¡VERIFICAR!

#endif

#endif

//..............................................................................

#if SONIDO

#include "sound.h"

#if ARDUINO\_UNO

const uint8\_t PIN\_BEEP = 9;

#elif ARDUINO\_MEGA

const uint8\_t PIN\_BEEP = 22;

#endif

#endif

//..............................................................................

#if DS3231

#include <MD\_DS3231.h>

// Selecciona el modo de operación de la librería del RTC DS3231.

// Sólo Modo Interrogatorio ("Polled Mode"), revisa retorno checkAlarm()

#define USE\_POLLED 1

// Modo Interrogatorio + "Callback". Si checkAlarm() activa, invocar "Callback"

#define USE\_POLLED\_CB 0

// Interrupción externa, se debe ajustar el pin PIN\_INTERRUPT para Arduino.

#define USE\_INTERRUPT 0

// Pin para manejar la Interrupción generada por el RTC DS3231.

#define PIN\_INTERRUPT 2

#endif

//..............................................................................

#if OTRO\_RECURSO

//... Biblioteca...

//... Terminales o Recursos Asignados ...

//... Instancia de Objetos asociados ...

//...

#endif

//==============================================================================

// Prototipos de Funciones para Navegación / Display:

//==============================================================================

void setupDisp(void);

void setupNav(void);

bool display(MD\_Menu::userDisplayAction\_t, char\* = nullptr);

MD\_Menu::userNavAction\_t navigation(uint16\_t &incDelta);

//==============================================================================

// Prototipos de Funciones Variables "get / set"

//==============================================================================

// Distribuidas para simplificación de uso, podrían estar en una misma función.

MD\_Menu::value\_t \*mnuLValueRqst(MD\_Menu::mnuId\_t id, bool bGet);

MD\_Menu::value\_t \*mnuBValueRqst(MD\_Menu::mnuId\_t id, bool bGet);

MD\_Menu::value\_t \*mnuIValueRqst(MD\_Menu::mnuId\_t id, bool bGet);

MD\_Menu::value\_t \*mnuSerialValueRqst(MD\_Menu::mnuId\_t id, bool bGet);

//==============================================================================

// --> Funciones agregadas al sistema de MENÚ <---:

//==============================================================================

MD\_Menu::value\_t \*myLEDCode(MD\_Menu::mnuId\_t id, bool bGet);

MD\_Menu::value\_t \*misSensores(MD\_Menu::mnuId\_t id, bool bGet);

MD\_Menu::value\_t \*misActuadores(MD\_Menu::mnuId\_t id, bool bGet);

//==============================================================================

#endif

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

Código Estructura.h: en este código se establecen los valores para cada opción del menú.

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

//-----[ ENCABEZADOS del MENÚ ]-----

const PROGMEM MD\_Menu::mnuHeader\_t mnuHdr[] = {

// Letrero que se muestra al presionar SELect

{ 10, "M E N U", 10, 15, 0 }, // Direcciona a: -[ ELEMENTOS del MENÚ ]- [15]

// ENCABEZADOS del MENÚ RAÍZ (Principales)

{ 11, "RELOJ", 20, 22, 0 }, // Reloj de Tiempo Real - Hora - Min ...

{ 12, "LISTAS", 24, 26, 0 }, // Listas de Opciones

{ 13, "LEDs", 28, 35, 0 }, // Acciones Directas con Salidas

{ 14, "SENSORES", 37, 39, 0 }, // Acciones de Toma de Datos

};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

//-----[ ELEMENTOS del MENÚ ]----- Menu Items -

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

const PROGMEM MD\_Menu::mnuItem\_t mnuItm[] = {

// INICIO del MENÚ Raíz (Principales Opciones de Selección)

{ 10, "RTC", MD\_Menu::MNU\_MENU, 11 }, // Salta a sección anterior.

{ 11, "LISTADOS", MD\_Menu::MNU\_MENU, 12 }, // Encabezados de las...

{ 12, "LED's", MD\_Menu::MNU\_MENU, 13 }, // ...Opciones que se...

{ 13, "SENSORES", MD\_Menu::MNU\_MENU, 14 }, // ...emplean.

//................................................

// Entrada de DATOS de SUB-MENÚS -> "RELOJ"

//................................................

{ 20, "Horas", MD\_Menu::MNU\_INPUT, 20 }, //

{ 21, "Minutos", MD\_Menu::MNU\_INPUT, 21 }, //

{ 22, "Segundos", MD\_Menu::MNU\_INPUT, 22 }, //

//................................................

// Entrada de DATOS de SUB-MENÚS -> "Listados"

//................................................

{ 24, "Lista - 01", MD\_Menu::MNU\_INPUT, 24 },

{ 25, "Lista - 02", MD\_Menu::MNU\_INPUT, 25 },

{ 26, "Lista - 03", MD\_Menu::MNU\_INPUT, 26 },

//................................................

// Entrada de DATOS de SUB-MENÚS -> "LED's"

//................................................

{ 28, "Amarillo", MD\_Menu::MNU\_INPUT, 28 },

{ 29, "Cyan", MD\_Menu::MNU\_INPUT, 29 },

{ 30, "Blanco", MD\_Menu::MNU\_INPUT, 30 },

{ 31, "ERROR SIMPLE", MD\_Menu::MNU\_INPUT, 31 },

{ 32, "ERROR GRAVE", MD\_Menu::MNU\_INPUT, 32 },

{ 33, "EN PROCESO", MD\_Menu::MNU\_INPUT, 33 },

{34, "EN ESPERA", MD\_Menu::MNU\_INPUT, 34 },

{ 35, "NORMAL", MD\_Menu::MNU\_INPUT, 35 },

//................................................

// Entrada de DATOS de SUB-MENÚS -> "SENSORES"

//................................................

{ 37, "Temperatura", MD\_Menu::MNU\_INPUT, 37 },

{ 38, "Infra1", MD\_Menu::MNU\_INPUT, 38 },

{ 39, "Infra2", MD\_Menu::MNU\_INPUT, 39 },

};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

// Elementos (Items) de Entrada para LISTAS ---------

const PROGMEM char list01[] = "SOLO|GATOS";

const PROGMEM char list02[] = "N|O|-|M|A|S|C|O|T|A|S";

const PROGMEM char list03[] = "E|M|E|R|G|E|N|C|I|A|-|B|O|M";

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

const PROGMEM MD\_Menu::mnuInput\_t mnuInp[] = {

{ 20, "Horas", MD\_Menu::INP\_INT, mnuIValueRqst, 2, 0, 0, 23, 0, 10, nullptr },

{ 21, "Minutos", MD\_Menu::INP\_INT, mnuIValueRqst, 2, 0, 0, 59, 0, 10, nullptr },

{ 22, "Segundos", MD\_Menu::INP\_INT, mnuIValueRqst, 2, 0, 0, 59, 0, 10, nullptr },

{ 24, "COM ->", MD\_Menu::INP\_LIST, mnuSerialValueRqst, 4, 0, 0, 0, 0, 0, list01 },

{ 25, "Dir ->", MD\_Menu::INP\_LIST, mnuSerialValueRqst, 3, 0, 0, 0, 0, 0, list02 },

{ 26, "Let ->", MD\_Menu::INP\_LIST, mnuSerialValueRqst, 1, 0, 0, 0, 0, 0, list03 },

// Lo siguiente se prueba usando el ÍNDICE de cada selección:

{ 28, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 29, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 30, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 31, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 32, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 33, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 34, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 35, "Listo", MD\_Menu::INP\_RUN, myLEDCode, 0, 0, 0, 0, 0, 10, nullptr },

{ 37, "", MD\_Menu::INP\_RUN, misSensores, 0, 0, 0, 0, 0, 10, nullptr },

{ 38, "", MD\_Menu::INP\_RUN, misSensores, 0, 0, 0, 0, 0, 10, nullptr },

{ 39, "", MD\_Menu::INP\_RUN, misSensores, 0, 0, 0, 0, 0, 10, nullptr },

};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Código Estructura:

Llama a los valores establecidos en el código “Menú” dependiendo de la opción que sea seleccionada por el usuario.

#include <MD\_Menu.h>

// #include "App00.h"

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

//---[ Código "Callback" para atender a entrada de valores set/get del menú ]---

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

// "CALLBACK" Value request callback for integers variables //

MD\_Menu::value\_t \*mnuIValueRqst(MD\_Menu::mnuId\_t id, bool bGet) {

// "id" = Ínidice de la variable accesada

// "bGet" = Valor Booleano, "0" -> Se Seleccionó; "1" -> NO se Seleccionó

// "vBuf.value" = Variable que RECIBE el cambio sugerido [actualizado]

// Atiende Índices (id) del 20 al 26

MD\_Menu::value\_t \*r = &vBuf; // VARIABLE que REGRESA el valor

// que se SELECCIONÓ (vBuf)

RTC.readTime(); // SIEMPRE leer hora-tiempo-calendario para luego ajustar

switch (id) { //

case 20: //

if (bGet) { // SI NO se hubiera seleccionado el campo...

Horas = RTC.h; //... Realiza lo siguiente...

vBuf.value = Horas; //...

}

else { // En caso contrario...

Horas = vBuf.value; // MODIFICA la variable asociada al valor

RTC.h = Horas; //... seleccionado

RTC.writeTime(); //

};

break;

case 21:

if (bGet) {

Minutos = RTC.m;

vBuf.value = Minutos;

}

else {

Minutos = vBuf.value;

RTC.m = Minutos;

RTC.writeTime();

};

break;

case 22:

if (bGet) {

Segundos = RTC.s;

vBuf.value = Segundos;

}

else {

Segundos = vBuf.value;

RTC.s = Segundos;

RTC.writeTime();

};

break;

}

return (r);

}

//------------------------------------------------------------------------------

MD\_Menu::value\_t \*mnuSerialValueRqst(MD\_Menu::mnuId\_t id, bool bGet) {

// "Callback" para valores índice de Listados. Los Índices deben ser GLOBALES.

MD\_Menu::value\_t \*r = &vBuf;

switch (id) {

case 24:

if (bGet)

vBuf.value = IndLista1;

else {

IndLista1 = vBuf.value;

}

break;

case 25:

if (bGet)

vBuf.value = IndLista2;

else {

IndLista2 = vBuf.value;

}

break;

case 26:

if (bGet)

vBuf.value = IndLista3;

else {

IndLista3 = vBuf.value;

}

break;

}

return (r);

}

//------------------------------------------------------------------------------

MD\_Menu::value\_t \*myLEDCode(MD\_Menu::mnuId\_t id, bool bGet) {

// "Callback" para ejecución de código. Aquí sólo se usa el índice.

switch (id) {

case 28:

leds[0] = CRGB::Yellow;

break;

case 29:

leds[0] = CRGB::Cyan;

break;

case 30:

leds[0] = CRGB::White;

break;

// -----

case 31: // ERROR SIMPLE (Tratará de recuperarse...)

do { // AMARILLO (Parpadeo corto)

leds[0] = CRGB::Yellow; FastLED.show(); delay(250);

leds[0] = CRGB::Black; FastLED.show(); delay(250);

necesitaMenu();

} while (navigation(dummy) != MD\_Menu::NAV\_SEL);

break;

case 32: // ERROR GRAVE (Requiere atención para continuar...)

do { // ROJO (Parpadeo corto)

leds[0] = CRGB::Red; FastLED.show(); delay(250);

leds[0] = CRGB::Black; FastLED.show(); delay(250);

necesitaMenu();

} while (navigation(dummy) != MD\_Menu::NAV\_SEL);

break;

case 33: // EN PROCESO (Está haciendo algo intermedio...)

do { // CYAN (Parpadeo medio)

leds[0] = CRGB::Cyan; FastLED.show(); delay(450);

leds[0] = CRGB::Black; FastLED.show(); delay(550);

necesitaMenu();

} while (navigation(dummy) != MD\_Menu::NAV\_SEL);

break;

case 34: // EN ESPERA (Necesita datos, información...)

do { // AZUL (Parpadeo corto)

leds[0] = CRGB::Blue; FastLED.show(); delay(250);

leds[0] = CRGB::Black; FastLED.show(); delay(1000);

necesitaMenu();

} while (navigation(dummy) != MD\_Menu::NAV\_SEL);

break;

case 35: // NORMAL (Todo bien...)

do { // VERDE (Parpadeo corto)

leds[0] = CRGB::Green; FastLED.show(); delay(250);

leds[0] = CRGB::Black; FastLED.show(); delay(1500);

necesitaMenu();

} while (navigation(dummy) != MD\_Menu::NAV\_SEL);

break;

default:

break;

}

FastLED.show();

return (nullptr);

}

//------------------------------------------------------------------------------

MD\_Menu::value\_t \*misSensores(MD\_Menu::mnuId\_t id, bool bGet) {

// "Callback" para ejecución de código. Aquí sólo se usa el índice.

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("= S E N S O R E S =");

lcd.setCursor(0, 1);

switch (id) {

case 37:

// lcd.print("Temperatura");

int dato;

float c;

dato = analogRead(A1);

c = ((500 \* dato) / (1023));

lcd.print(c);

lcd.print(" ° ");

break;

case 38:

lcd.print("Infra1");

break;

case 39:

lcd.print("Infra2");

break;

} while (digitalRead(4) != LOW); // Adecuar al tipo de Menú de Navegación

};

/\* lcd.setCursor(0, 3);

for (int i = 0; i < 10; i++) {

lcd.print(i, HEX);

delay(100);

}

return (nullptr);

}\*/

//------------------------------------------------------------------------------

//

//==============================================================================

Código Menú\_Disp.cpp:

Rutinas para desplegar datos específicos.

//##############################################################################

#include "App02.h"

// Rutinas para el hardware de despliegue de datos específico, definiendo:

// Archivos de encabezado específicos; Definiciones de circuitería;

// Objetos de librería globales; Funciones "callback" para el desplegador.

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

#if DISPLAY\_SERIAL

// Despliega la salida de datos en un monitor serial. Útil para la depuración y

// búsqueda de errores en la definición de la estructura del menú sin

// interactuar con el dispositivo final.

bool display(MD\_Menu::userDisplayAction\_t action, char \*msg){

switch (action){

case MD\_Menu::DISP\_INIT:

Serial.begin(BAUD\_RATE);

break;

case MD\_Menu::DISP\_CLEAR:

Serial.print("\n-> CLS");

break;

case MD\_Menu::DISP\_L0:

Serial.print("\n0> ");

Serial.print(msg);

break;

case MD\_Menu::DISP\_L1:

Serial.print("\n1> ");

Serial.print(msg);

break;

}

return (true);

}

#endif

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

#if DISPLAY\_LCDSHIELD

// Salida para una pantalla LCD de UNA, DOS o CUATRO líneas.

// --> Para una LCD de una sola línea, comentar el código que maneja L0. <--

// Antes de mostrar el mensaje que se imprime se limpia la pantalla (espacios).

// --> Se inicializa (crea OBJETO) en el archivo \*.h

bool display(MD\_Menu::userDisplayAction\_t action, char \*msg) {

static char szLine[LCD\_COLS + 1] = { '\0' }; // Para "limpieza" zona de la LCD

switch (action) {

case MD\_Menu::DISP\_INIT:

lcd.init();

lcd.backlight();

lcd.clear();

memset(szLine, ' ', LCD\_COLS);

break;

case MD\_Menu::DISP\_CLEAR:

lcd.clear();

break;

case MD\_Menu::DISP\_L0:

lcd.setCursor(0, 0);

lcd.print(szLine);

lcd.setCursor(0, 0);

lcd.print(msg);

break;

case MD\_Menu::DISP\_L1:

lcd.setCursor(0, 1);

lcd.print(szLine);

lcd.setCursor(0, 1);

lcd.print(msg);

break;

}

return (true);

}

#endif

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

#if DISPLAY\_PAROLA

// Salida simple para una Matriz LEDs (Una línea) para la librería Parola.

#include <MD\_MAX72XX.h>

#include <MD\_Parola.h>

#include <SPI.h>

// Define el número de dispositivos matrices e interface de hardware.

// NOTA: Verificar que coincida el número de los pines con la circuitería.

// Falta el tipo de display: FC16\_HW, GENERIC\_HW, PAROLA\_HW, ICSTATION\_HW

#define HARDWARE\_TYPE MD\_MAX72XX::PAROLA\_HW

#define MAX\_DEVICES 16

#define CLK\_PIN 13

#define DATA\_PIN 11

#define CS\_PIN 10

// Conexión al SPI por Hardware:

MD\_Parola P = MD\_Parola(CS\_PIN, MAX\_DEVICES);

bool display(MD\_Menu::userDisplayAction\_t action, char \*msg) {

switch (action) {

case MD\_Menu::DISP\_INIT:

P.begin();

break;

case MD\_Menu::DISP\_CLEAR:

P.displayClear();

break;

case MD\_Menu::DISP\_L0:

// Trabajar esta parte para hacer 2 "Zonas" de desplegado, ésta es una

// P.print(msg); // Solamente se emplea una zona, no existe "Línea 0"

break;

case MD\_Menu::DISP\_L1:

P.print(msg);

break;

}

return (true);

}

#endif

//##############################################################################

Código Menu\_Nav.cpp:

Código para la navegación.

//##############################################################################

#include "App02.h"

// Rutinas para circuitería específica para dispositivos de navegación

// (introducción de datos para indicaciones de acciones en la selección del

// menú), definiendo: Archivos de encabezados; Definciones de circuitería;

// Objetos de librerías globales; Funciones de "callback" de navegación.

//==============================================================================

#if INPUT\_SWITCH

// Implementación de 3 interruptores momentaneos, uno para INCrementos, otro

// para DECrementos, otro para la SELección (un "click") y uno final para el

// ESCape ("presionado largo").

//------------------------------------------------------------------------------

#include <MD\_UISwitch.h>

const uint8\_t INC\_PIN = 2;

const uint8\_t DEC\_PIN = 3;

const uint8\_t CTL\_PIN = 4;

const uint8\_t ESC\_PIN = 5; // Agregado, mejora el acceso al teclado básico.

uint8\_t pins[] = {INC\_PIN, DEC\_PIN, CTL\_PIN, ESC\_PIN}; // Arreglo Interruptores

MD\_UISwitch\_Digital swNav(pins, ARRAY\_SIZE(pins), LOW); // "swNav" == Todos

void setupNav(void) { // los Interruptores

swNav.begin(); //... Inciados

swNav.enableRepeat(false); //... Sin Repeticiones

}

//-----[ RUTINA para LECTURA de la NAVEGACIÓN ]-----

MD\_Menu::userNavAction\_t navigation(uint16\_t &incDelta) {

MD\_Menu::userNavAction\_t nav = MD\_Menu::NAV\_NULL; // Variable "nav" regresa el

// "COMANDO de Navegación"

switch (swNav.read()) { // Verificar si no debiera ser:

case MD\_UISwitch::KEY\_PRESS: { // "case MD\_UISwitch::KEY\_LONGPRESS:"

Serial.print(swNav.getKey()); // Lectura de la tecla

switch (swNav.getKey()) { // Verifica cuál tecla se leyó

// Versión MD\_UISwitch [ACTUAL], se analiza el valor del "pin":

case INC\_PIN: nav = MD\_Menu::NAV\_INC; break;

case DEC\_PIN: nav = MD\_Menu::NAV\_DEC; break;

case CTL\_PIN: nav = MD\_Menu::NAV\_SEL; break;

case ESC\_PIN: nav = MD\_Menu::NAV\_ESC; break;

}

}

break;

}

incDelta = 1;

return (nav);

}

#endif

//------------------------------------------------------------------------------

#if INPUT\_LCDSWITCH

// Empleo de interruptores momentáneos encontrados en "LCD shield". Las teclas

// "up" y "down" se asocian con INC y DEC, mientras "izquierda" y "derecha"

// se asocian a ESC y SEL.

//------------------------------------------------------------------------------

#include <MD\_UISwitch.h>

#define LCD\_KEYS\_PIN A0

// Estos valores funcionan para la mayoría de los casos:

MD\_UISwitch\_Analog::uiAnalogKeys\_t kt[] = {

{ 10, 10, 'R' }, // Derecha

{ 140, 15, 'U' }, // Arriba

{ 315, 15, 'D' }, // Abajo

{ 490, 15, 'L' }, // Izquierda

{ 725, 15, 'S' }, // Selecciona

};

MD\_UISwitch\_Analog lcdKeys(LCD\_KEYS\_PIN, kt, ARRAY\_SIZE(kt));

void setupNav(void) {

lcdKeys.begin();

}

//-----[ RUTINA para LECTURA de la NAVEGACIÓN ]-----

MD\_Menu::userNavAction\_t navigation(uint16\_t &incDelta) {

incDelta = 1;

if (lcdKeys.read() == MD\_UISwitch::KEY\_PRESS)

{

switch (lcdKeys.getKey())

{

case 'D': return (MD\_Menu::NAV\_DEC);

case 'U': return (MD\_Menu::NAV\_INC);

case 'R': return (MD\_Menu::NAV\_SEL);

case 'S':

case 'L': return (MD\_Menu::NAV\_ESC);

}

}

return (MD\_Menu::NAV\_NULL);

}

#endif

//------------------------------------------------------------------------------

#if INPUT\_RENCODER

// Implementado con encodificador rotativo e interruptor de presión integrado.

// Una rotación en el sentido horario, INC; en sentido contrario a DEC.

// Presionando interruptor central equivale a SEL ("press") o ESC ("long press").

//

// Si se estuviera editando un valor, el encodificador rotatorio también

// aplicará un factor de que que tan rápido es rotado para incrementar la tasa

// de variación del cambio del valor. Esto aplicará en entradas numéricas.

#include <MD\_REncoder.h>

#include <MD\_UISwitch.h>

extern MD\_Menu M;

const uint8\_t RE\_A\_PIN = 2;

const uint8\_t RE\_B\_PIN = 3;

const uint8\_t CTL\_PIN = 4;

const uint8\_t ESC\_PIN = 5; // Agregado, mejora el acceso al teclado básico.

uint8\_t pins[] = {CTL\_PIN, ESC\_PIN};

MD\_REncoder RE(RE\_A\_PIN, RE\_B\_PIN);

MD\_UISwitch\_Digital swNav(pins, ARRAY\_SIZE(pins), LOW);

void setupNav(void) {

RE.begin();

swNav.begin();

swNav.enableRepeat(false);

swNav.setPressTime(150);

swNav.setLongPressTime(600);

/\*

const uint16\_t KEY\_PRESS\_TIME = 150; //Default key press [milliseconds]

const uint16\_t KEY\_DPRESS\_TIME = 250; //Default double press [milliseconds]

const uint16\_t KEY\_LONGPRESS\_TIME = 600; //Default long press [milliseconds]

const uint16\_t KEY\_REPEAT\_TIME = 300; //Default between repeats [milliseconds]

const uint8\_t KEY\_ACTIVE\_STATE = LOW; ///

\*/

}

//-----[ RUTINA para LECTURA de la NAVEGACIÓN ]-----

MD\_Menu::userNavAction\_t navigation(uint16\_t &incDelta) {

MD\_Menu::userNavAction\_t nav = MD\_Menu::NAV\_NULL;

uint8\_t re = RE.read();

if (re != DIR\_NONE) {

incDelta = (M.isInEdit() ? (1 << abs(RE.speed() / 10)) : 1);

return (re == DIR\_CCW ? MD\_Menu::NAV\_DEC : MD\_Menu::NAV\_INC);

}

switch (swNav.read()) {

case MD\_UISwitch::KEY\_LONGPRESS: {

Serial.print(swNav.getKey());

switch (swNav.getKey()) {

case CTL\_PIN: nav = MD\_Menu::NAV\_SEL; break;

case ESC\_PIN: nav = MD\_Menu::NAV\_ESC; break;

}

}

break;

}

return (nav);

}

#endif

//##############################################################################