Ulises Dávalos

Objetivo: Creación de microarquitecturas.

Cuando se trabaja con sistemas embebidos pueden suceder dos situaciones:

- Es fácil subestimar la complejidad de un sistema embebido
- Parálisis de acción sobre como comenzar debido a la complejidad final del proyecto

Una respuesta para ambas situaciones puede ser la misma (depende del proceso creador del equipo responsable):

COMENZAR EN PEQUEÑO o lo que es lo mismo, darle valor al prototipado de soluciones posibles.

En ésta práctica usaremos un micro kernel (uKaos) para implementar un sistema operativo de tiempo real, que contra lo que pudiera pensarse, nos ayudara a simplificar el desarrollo de nuestros prototipos al permitir el uso de tareas concurrentes y manejo de eventos.

A diferencia de la anterior, en esta ocasión tocaremos varios temas interesantes.

- Implementaremos el mismo uKaos RTOS en ambos micros, en el PIC16F15313 que nos ayuda a prototipar los módulos que cuidan de las macetas (esclavos) y en el ESP8266 que representa a el módulo manejador (maestro) que coordina a los primeros.
- En ambos prototipos, de manera sencilla, se manejarán eventos (interrupciones)
- Los dos prototipos manejarán tareas (Round Rbin) concurrentes en diferentes divisiones de tiempo.
- Se comunicarán estos prototipos mediante un protocolo serial I2C
- El prototipo maestro ofrecerá un acceso inalámbrico al usuario

Para éste ejercicio necesitarás:

- 2 LEDs de *'hearth beat'* que nos permite saber que los módulos están activos. Estos LEDs tendrán un tiempo de ciclo de 1 segundo aproximadamente (pin 5) en el esclavo, LED interconstruido en el maestro.
- LED que avisa que ya no hay agua en la reserva, (pin 14 del ESP8266), una vez que se alcanza un nivel adecuado se apaga.
- LED de emergencia en el PIC16F15313 que prende cuando se cumplen las condiciones de (pin 6) :
    - En el PIC16F15313 la orden de prendido o apagado viene del ESP8266
    - Falta de agua en la reserva (Dato capturado por el maestro, ESP8266 pin 16)
    - El nivel de humedad ha alcanzado un nivel crítico, sensado en el PIC16F15313, pin 7 y enviado al ESP8266 para su procesamiento.
- Switch que simula el sensor de la reserva de agua conectado al pin 16 del maestro
- Potenciómetro que simula el nivel de humedad en el esclavo, pin 7.
- 2 resistencias (4.7k) para el bus i2c
- Resistencias varias para los LEDs (220~330 Ohms), 4.7k conectada de VDD a pin 4 del esclavo, etc.

Además:

- Programador para el PIC y MPLab
- ESP8266 y Arduino IDE instalado con el soporte para ESP8266 con librería `ESP8266TimerInterrupt`

Código base (esclavo)

- main.c

```
/**
  Generated Main Source File

  Company:
    Microchip Technology Inc.

  File Name:
    main.c

  Summary:
    This is the main file generated using PIC10 / PIC12 / PIC16 / PIC18 MCUs

  Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.81.4
        Device            :  PIC16F15313
        Driver Version    :  2.00
*/

/*
    (c) 2018 Microchip Technology Inc. and its subsidiaries.

    Subject to your compliance with these terms, you may use Microchip software and any
    derivatives exclusively with Microchip products. It is your responsibility to comply with third party
    license terms applicable to your use of third party software (including open source software) that
    may accompany Microchip software.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS
    FOR A PARTICULAR PURPOSE.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP
    HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO
    THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL
    CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT
    OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS
```

```
    SOFTWARE.
*/

#include "mcc_generated_files/mcc.h"
#include "OS_uKaos.h"

// --->> uKaos store area

// This counter is the central part to the ukaos OS.
// Each group of tasks running on the same time slot are triggered
// by the change on sate of the corresponding bit of this counter.

// Time slot by bit position:
//
//  Hi byte                                                         LO byte
//< 33.5s - 16.7s - 8.3s - 4.1s - 2.0s - 1.0s - 524.2ms - 262.1ms >< 131.0ms - 65.5ms - 32.7ms - 16.3ms - 8.1ms - 4.9ms - 2.0ms - 1.0ms >

    unsigned short int
usRippleCount = 0
    ;

    unsigned short int
usRipplePast = 0
    ;

    unsigned short int
usActiveTaskGroup
    ;

// --->> Application store  area

    adc_result_t
 humidity = 0
    ;

    unsigned char
ubAlarmLED = 0;
    ;

// ::::: uKaos OS ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    void
main(void)
    {

    OS_INT:
        {
        SYSTEM_Initialize();

        usRippleCount = 0;                                  // Time base init
        TMR0_SetInterruptHandler( incRippleCount );

        ADC_SelectChannel( sense_ANA0 );                    // Init humidity sensing
        ADC_SetInterruptHandler( storeSensorVal );
```

```
95
96          I2C1_Open();
97
98          INTERRUPT_PeripheralInterruptEnable();              // Enable
    events
99          INTERRUPT_GlobalInterruptEnable();
100
101         ADC_StartConversion();                              // Start
    humidity sensing
102         }
103
104     idle:
105         {
106         if ( usRippleCount == usRipplePast ) goto idle;     // Wait until
    a time slot passes
107
108         usActiveTaskGroup = usRippleCount;                  // Find active
    slot
109         usActiveTaskGroup ^= usRipplePast;
110         usActiveTaskGroup &= usRippleCount;
111
112         usRipplePast = usRippleCount;
113         }
114
115     task_500ms:
116         {
117         if  ( ( usActiveTaskGroup ^ 0x0200 ) == 0 )
118             {
119             beat_RA2_Toggle();                              // Toggle
    hearth beat LED
120             }
121         }
122
123     task_1s:
124         {
125         if  ( ( usActiveTaskGroup ^ 0x0400 ) == 0 )
126             {
127             task_list_1s();                                 // Activate
    tasks running each second
128             }
129         }
130
131     task_33s:
132         {
133         if  ( ( usActiveTaskGroup ^ 0x8000 ) == 0 )
134             {
135             task_list_33s();                                // Activate
    tasks running every ~30 seconds
136             }
137         }
138
139     goto idle;                                              // go for the
    next time slot
140     }
141 /******************************************************************************
142  End of File
143 ******************************************************************************/
```

- Manejador de eventos (event_mgr.c)

```
/*
 * File:   event_mgr.c
 * Author: Terra.Drakko
 *
 * Created on July 21, 2020, 12:05 AM
 */


#include "mcc_generated_files/mcc.h"
#include "OS_uKaos.h"
#include "event_mgr.h"

    void
incRippleCount( void )
    {
    usRippleCount++;
    }

    void
storeSensorVal( void )
    {
    humidity = ADC_GetConversionResult();
    i2c1WrData = humidity;
    }
```

- Tareas corriendo cada segundo (task_list_1s.c)

```
/*
 * File:   task_list_1s.c
 * Author: Terra.Drakko
 *
 * Created on July 21, 2020, 10:43 AM
 */


#include "mcc_generated_files/mcc.h"
#include "OS_uKaos.h"


    void
alarmLEDMgr()
    {
    ubAlarmLED = i2c1RdData;
    led_RA1_SetLow();
    if  ( ubAlarmLED ) led_RA1_SetHigh();
    }

    void
task_list_1s(void)                                  // Round Robin scheduling
    {
 // waterLevelSense();
    alarmLEDMgr();
    }
```

- Tareas corriendo cada medio minuto (task_list_33s.c)

```
/*
 * File:   task_list_33s.c
 * Author: Terra.Drakko
 *
 * Created on July 21, 2020, 1:40 PM
 */


#include "mcc_generated_files/mcc.h"
#include "OS_uKaos.h"

    void
measureHumidity( void )
    {
    if  ( ADC_IsConversionDone() )
        { ADC_StartConversion(); }
    }

    void
task_list_33s(void)                             // Round Robin scheduling
    {
    measureHumidity();
    }
```

Cabeceras

- event_mgr.h

```
/* Microchip Technology Inc. and its subsidiaries.  You may use this software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
 * TERMS.
```

```
 */

/*
 * File:
 * Author:
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _event_mgr_h_
#define _event_mgr_h_

#include <xc.h> // include processor files - each processor file is
guarded.

// TODO Insert appropriate #include <>

// TODO Insert C++ class definitions if appropriate

// TODO Insert declarations

// Comment a function and leverage automatic documentation with slash star
star
/**
    <p><b>Function prototype:</b></p>

    <p><b>Summary:</b></p>

    <p><b>Description:</b></p>

    <p><b>Precondition:</b></p>

    <p><b>Parameters:</b></p>

    <p><b>Returns:</b></p>

    <p><b>Example:</b></p>
    <code>

    </code>

    <p><b>Remarks:</b></p>
 */
// TODO Insert declarations or function prototypes (right here) to leverage
// live documentation

#ifdef  __cplusplus
extern "C" {
#endif /* __cplusplus */

    // TODO If C++ is being used, regular C code needs function names to
have C
    // linkage so the functions can be used by the c code.

    void
incRippleCount( void )
```

```
    ;

    void
storeSensorVal( void )
    ;

#ifdef  __cplusplus
}
#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */


```

- OS_uKaos.h

```
/* Microchip Technology Inc. and its subsidiaries.  You may use this
software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL,
PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP
HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL
CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES,
IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
THESE
 * TERMS.
 */

/*
 * File:
 * Author:
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not
included
// more than once.
#ifndef _OS_uKaos_h_
#define _OS_uKaos_h_
```

```
33
34  #include <xc.h> // include processor files - each processor file is
    guarded.
35  #include "event_mgr.h"
36  #include "task_list_1s.h"
37  #include "task_list_33s.h"
38
39  // TODO Insert appropriate #include <>
40
41  // TODO Insert C++ class definitions if appropriate
42
43  // TODO Insert declarations
44
45  // Comment a function and leverage automatic documentation with slash star
    star
46  /**
47      <p><b>Function prototype:</b></p>
48
49      <p><b>Summary:</b></p>
50
51      <p><b>Description:</b></p>
52
53      <p><b>Precondition:</b></p>
54
55      <p><b>Parameters:</b></p>
56
57      <p><b>Returns:</b></p>
58
59      <p><b>Example:</b></p>
60      <code>
61
62      </code>
63
64      <p><b>Remarks:</b></p>
65   */
66  // TODO Insert declarations or function prototypes (right here) to
    leverage
67  // live documentation
68
69  #ifdef  __cplusplus
70  extern "C" {
71  #endif /* __cplusplus */
72
73      // TODO If C++ is being used, regular C code needs function names to
    have C
74      // linkage so the functions can be used by the c code.
75  extern
76      unsigned short int
77  usRippleCount
78      ;
79
80  extern
81      adc_result_t
82   humidity
83      ;
84
85  extern
86      unsigned char
```

```
ubAlarmLED
    ;

extern
    volatile uint8_t
i2c1WrData
    ;


extern
    volatile uint8_t
i2c1RdData
    ;

extern
    volatile uint8_t
i2c1SlaveAddr
    ;
#ifdef  __cplusplus
}
#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */

```

- task_list_1s.h

```
/* Microchip Technology Inc. and its subsidiaries.  You may use this
software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL,
PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP
HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES,
IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
THESE
 * TERMS.
 */

/*
 * File:
```

```
 * Author:
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _task_list_1s_
#define _task_list_1s_

#include <xc.h> // include processor files - each processor file is guarded.

// TODO Insert appropriate #include <>

// TODO Insert C++ class definitions if appropriate

// TODO Insert declarations

// Comment a function and leverage automatic documentation with slash star star
/**
    <p><b>Function prototype:</b></p>

    <p><b>Summary:</b></p>

    <p><b>Description:</b></p>

    <p><b>Precondition:</b></p>

    <p><b>Parameters:</b></p>

    <p><b>Returns:</b></p>

    <p><b>Example:</b></p>
    <code>

    </code>

    <p><b>Remarks:</b></p>
 */
// TODO Insert declarations or function prototypes (right here) to leverage
// live documentation

#ifdef  __cplusplus
extern "C" {
#endif /* __cplusplus */

    // TODO If C++ is being used, regular C code needs function names to have C
    // linkage so the functions can be used by the c code.

    void
task_list_1s(void)
    ;

#ifdef  __cplusplus
}
```

```
#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */

```

- task_list_33s.h

```
/* Microchip Technology Inc. and its subsidiaries.  You may use this software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
 * TERMS.
 */

/*
 * File:
 * Author:
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _task_list_33s_
#define _task_list_33s_

#include <xc.h> // include processor files - each processor file is guarded.

// TODO Insert appropriate #include <>

// TODO Insert C++ class definitions if appropriate

// TODO Insert declarations

```

```
// Comment a function and leverage automatic documentation with slash star star
/**
    <p><b>Function prototype:</b></p>

    <p><b>Summary:</b></p>

    <p><b>Description:</b></p>

    <p><b>Precondition:</b></p>

    <p><b>Parameters:</b></p>

    <p><b>Returns:</b></p>

    <p><b>Example:</b></p>
    <code>

    </code>

    <p><b>Remarks:</b></p>
 */
// TODO Insert declarations or function prototypes (right here) to leverage
// live documentation

#ifdef  __cplusplus
extern "C" {
#endif /* __cplusplus */

    // TODO If C++ is being used, regular C code needs function names to have C
    // linkage so the functions can be used by the c code.
    void
task_list_33s(void)
    ;

#ifdef  __cplusplus
}
#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */
```

Código base (maestro)

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>
// These define's must be placed at the beginning before #include
// "ESP8266TimerInterrupt.h"
#define TIMER_INTERRUPT_DEBUG      1
#include "ESP8266TimerInterrupt.h"
#include "ESP8266_ISR_Timer.h"
#include <Wire.h>

// ::: I2C <<-------------------------------------------------------------
```

```
unsigned char uchAddress;       // slave address

void find_First_I2C_Module( void );

// ::: WiFi <<-------------------------------------------------------
ESP8266WiFiMulti wifiMulti;     // Create an instance of the
                                //    ESP8266WiFiMulti class,
                                //    called 'wifiMulti'
ESP8266WebServer server(80);    // Create a webserver object that
                                //    listens for HTTP request on
                                //    port 80

void initWiFi( void );
void handleRoot();              // function prototypes for HTTP
                                //   handlers
void handleLogin();
void handleNotFound();
void handleUser();

// ::: Timer <<-----------------------------------------------------
ESP8266Timer ITimer;
void ICACHE_RAM_ATTR TimerHandler( void );

// ::: uKaos <<-----------------------------------------------------
// This counter is the central part to the ukaos OS.
// Each group of tasks running on the same time slot are triggered
// by the change on sate of the corresponding bit of this counter.

// Time slot by bit position:
//
//  Hi byte
//< 33.5s - 16.7s - 8.3s - 4.1s - 2.0s - 1.0s - 524.2ms - 262.1ms >
//  LO byte
//< 131.0ms - 65.5ms - 32.7ms - 16.3ms - 8.1ms - 4.9ms - 2.0ms - 1.0ms >
unsigned short usRippleCount = 0;

unsigned short usRipplePast  = 0;
unsigned short usActiveTaskGroup;

// ::: Application <<-----------------------------------------------
const int     led    =  2;
const int     in_sw  = 16;
const int     sw_led = 14;
unsigned int  sw_actual, sw_past;
unsigned char water;
unsigned char alarm;
unsigned int  humidity;
unsigned char address;

void task_list_32ms( void );
void task_list_1s  ( void );
void task_list_4s  ( void );

    void
setup()
    {
    serial:
       Serial.begin( 115200 );
```

```
72          while (!Serial);         // wait for serial monitor
73
74      i2c:
75          uchAddress = 0;
76          Wire.begin( 4, 5 );
77          find_First_I2C_Module();
78
79      beat:
80          pinMode( led, OUTPUT );
81          digitalWrite( led, 0 );
82
83      wifi:
84          wifiMulti.addAP( "<tu red>", "<tu contraseña>" );   // add Wi-Fi
85                                   //    networks you want to connect to
86          initWiFi();
87
88      tasks:
89          // up to 16 task handlers.......
90          ITimer.attachInterruptInterval( 500, TimerHandler);
91
92      app:
93          water = 0; sw_actual = 0; sw_past = 0; humidity = 0; alarm = 0;
94          pinMode( in_sw , INPUT  );
95          pinMode( sw_led, OUTPUT );
96      }
97
98  void loop()
99      {
100     web_server:
101         server.handleClient();
102
103     // uKaos
104     idle:
105         {
106         // Wait until a time slot passes
107         if ( usRippleCount == usRipplePast ) goto exit;
108
109         // Find active slot
110         usActiveTaskGroup = usRippleCount;
111         usActiveTaskGroup ^= usRipplePast;
112         usActiveTaskGroup &= usRippleCount;
113
114         usRipplePast = usRippleCount;
115         }
116
117     task_32ms:
118         {
119         if  ( ( usActiveTaskGroup ^ 0x0020 ) == 0 )
120             {
121             // Activate tasks running every 32ms
122             task_list_32ms();
123             }
124         }
125
126     task_500ms:
127         {
128         if  ( ( usActiveTaskGroup ^ 0x0200 ) == 0 )
129             {
```

```
            // Toggle hearthbeat Built in LED
            digitalWrite( led, !digitalRead( led ) );
            }
        }

    task_1s:
        {
        if  ( ( usActiveTaskGroup ^ 0x0400 ) == 0 )
            {
            // Activate tasks running each second
            task_list_1s();
            }
        }

    task_4s:
        {
        if  ( ( usActiveTaskGroup ^ 0x1000 ) == 0 )
            {
            // Activate tasks running each 4 seconds
            task_list_4s();
            }
        }

    exit:
        return;
    }

// ::: Init section <<-------------------------------------------------

    void
initWiFi( void )
    {
    Serial.println("Connecting ...");
    int i = 0;

    while (wifiMulti.run() != WL_CONNECTED)
       { // Wait for the Wi-Fi to connect: scan for Wi-Fi networks, and
         //   connect to the strongest of the networks above
       delay(250);
       Serial.print('.');
       }
    Serial.println('\n');
    Serial.print("Connected to ");
    Serial.println(WiFi.SSID());    // Tell us what network we're
                                    // connected to
    Serial.print("IP address:\t");
    Serial.println(WiFi.localIP()); // Send the IP address of the
                                    //    ESP8266 to the computer

    if (MDNS.begin("esp8266"))
       {              // Start the mDNS responder for esp8266.local
       Serial.println("mDNS responder started");
       }
    else
       {
       Serial.println("Error setting up MDNS responder!");
       }

```

```
    server.on("/", HTTP_GET, handleRoot);        // Call the
                                 //    'handleRoot' function when a
                                 //    client requests URI "/"
    server.on("/login", HTTP_POST, handleLogin); // Call the
                                 //    'handleLogin' function when a POST
                                 //    request is made to URI "/login"
    server.on("/user",HTTP_GET, handleUser );
    server.onNotFound(handleNotFound);           // When a client
                                 //    requests an unknown URI (i.e.
                                 //    something other than "/"), call
                                 //    function "handleNotFound"

    server.begin();              // Actually start the server
    Serial.println("HTTP server started");
    }

// ------------------------------------------------------------------------
    void
find_First_I2C_Module()
    {
    byte error;

    Serial.println
        (
        "\n\nScan for I2C devices on port pair D4(SDA)and D5(SCL)"
        )
        ;
    Serial.print( "Scanning (SDA : SCL) - D4 : D5 - " );

    for ( address = 1; address < 128; address++ )
        {
        // The i2c_scanner uses the return value of
        // the Write.endTransmisstion to see if
        // a device did acknowledge to the address.
        Wire.beginTransmission( address );
        error = Wire.endTransmission();

        if  ( error == 0 )
            {
            Serial.print( "I2C device found at address 0x" );
            if ( address < 16 ) Serial.print( "0" );
            Serial.print( address, HEX );
            Serial.println("  !");
            uchAddress = address;

            break;
            }
        }
    if  ( address == 128 ) Serial.println("No I2C devices found");
    else Serial.println("**********************************\n");
    }

// ::: App. tasks section <<----------------------------------------------
    void
readSwitch( void )
    {
    sw_actual = digitalRead( in_sw );
    water     = sw_actual & sw_past;
```

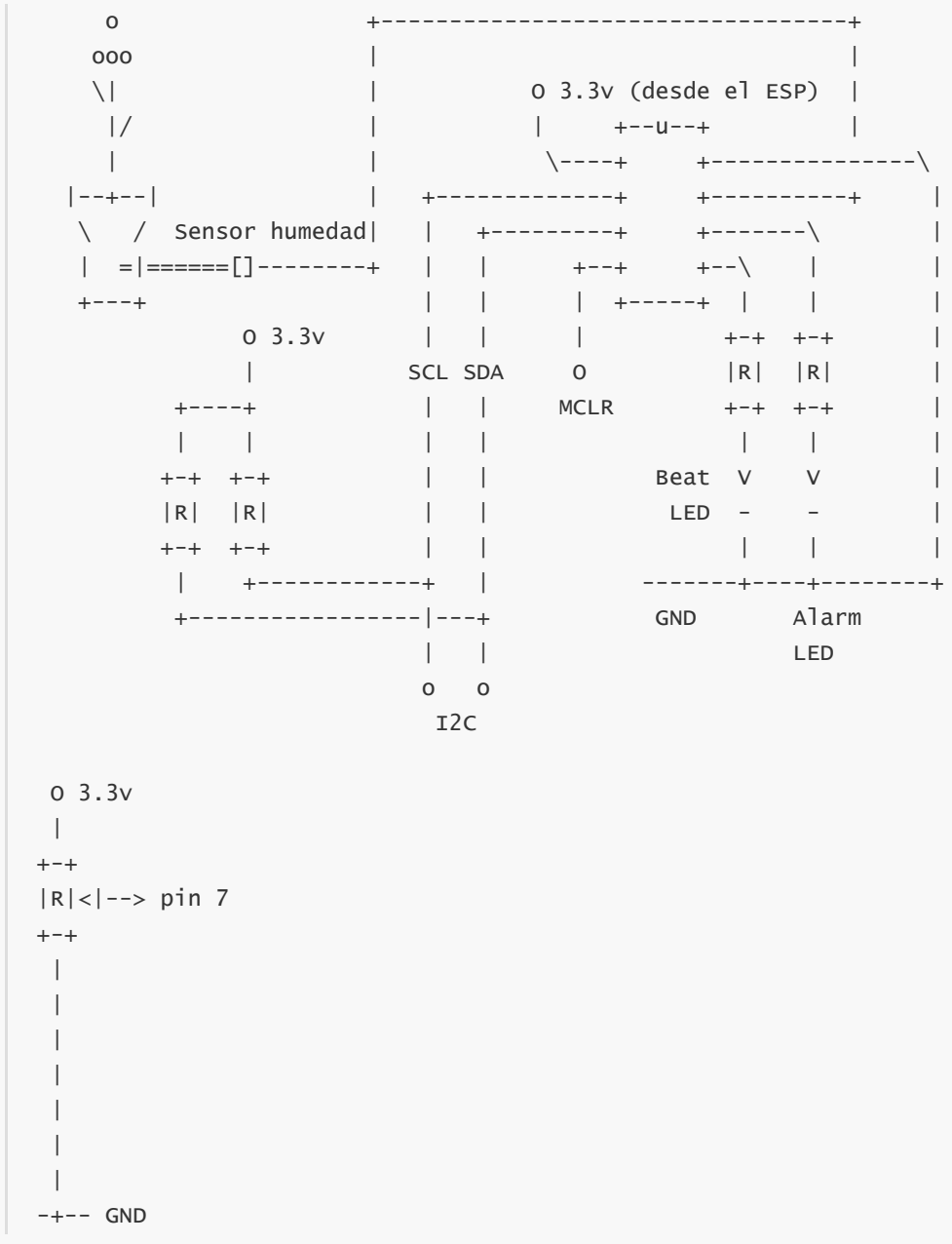
```
246     sw_past   = sw_actual;
247     }
248
249     void
250 echoSwitch( void )
251     {
252     digitalWrite( sw_led, water );
253     }
254
255     void
256 task_list_32ms( void )
257     {
258     readSwitch();
259     echoSwitch();
260     }
261
262 // ----------------------------------------------------------------------
263     void
264 alarmLogic( void )
265     {
266     alarm = 0;
267
268     // Alarm rules
269     if  ( water    == 0    ) goto exit;
270     if  ( humidity <  0x80 ) goto exit;
271
272     alarm = 1;
273
274     exit:
275         return;
276     }
277
278     void
279 txAlarmLevel( void )
280     {
281     Wire.beginTransmission( address ); // transmit to dev. <address>
282     Wire.write( alarm );               // sends value byte
283     Wire.endTransmission();            // stop transmitting
284     }
285
286     void
287 task_list_1s( void )
288     {
289     alarmLogic();
290     txAlarmLevel();
291     }
292
293 // ----------------------------------------------------------------------
294     void
295 readHumidity( void )
296     {
297     // request 1 bytes from slave
298     Wire.requestFrom( ( uint8_t )address, ( uint8_t )1 );
299
300     if  ( Wire.available() )        // read available data
301         {
302         humidity = Wire.read();     // receive a data byte
303         }
```

```
    }

    void
task_list_4s( void )
    {
    readHumidity();
    }

// ::: Event handlers section <<---------------------------------------

    void ICACHE_RAM_ATTR
TimerHandler( void )
    {
    usRippleCount++;
    }

    void
handleRoot()
    {
    // When URI / is requested, send a web page with a button to toggle
    // the LED
    server.send
        (
        200,
        "text/html",
        "<form action=\"/login\" method=\"POST\">"
        "  <input type=\"text\" name=\"username\" "
           "placeholder=\"Username\">"
        "  </br>"
        "  <input type=\"password\" name=\"password\" "
           "placeholder=\"Password\">"
        "  </br>"
        "  <input type=\"submit\" value=\"Login\">"
        "</form>"
        "<p>Try 'Ulises Davalos' and 'pwd123' ...</p>"
        )
        ;
    }

    void
handleLogin()
    {                         // If a POST request is made to URI /login
    if  (
        ! server.hasArg("username")         ||
        ! server.hasArg("password")         ||
          server.arg   ("username") == NULL ||
          server.arg   ("password") == NULL
        )
        {
        // If the POST request doesn't have username and password data
        // The request is invalid, so send HTTP status 400
        server.send(400, "text/plain", "400: Invalid Request");
        return;
        }
    if  (
           server.arg( "username" ) == "Ulises Davalos"
        && server.arg( "password" ) == "pwd123"
        )
```

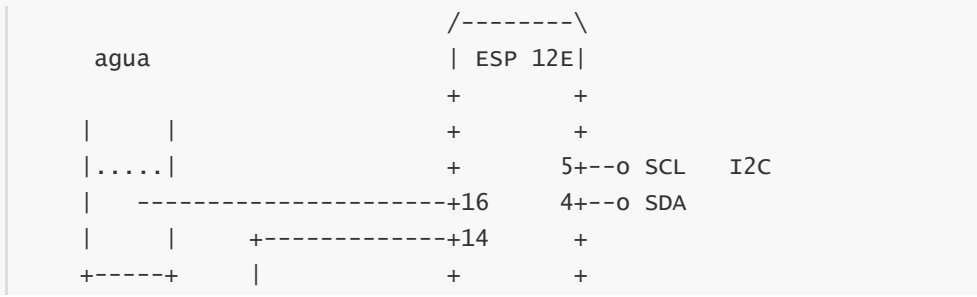
```
        { // If both the username and the password are correct
        server.send
           (
           200,
           "text/html",
           "<h1>Welcome, " + server.arg("username") + "!</h1>"
           "<p>Login successful <a href=\"/user\">Go to ESP8266"
           " I/O!!!</a> </p>"
           )
           ;
        }
    else
        {
        // Username and password don't match
        server.send(401, "text/plain", "401: Unauthorized");
        }
    }

    static char
response[ 200 ]
    ;

   void
handleUser()
    {
    // When URI / is requested, send a web page with a button to toggle
    // the LED
    sprintf
       (
       response,
       "<p>Water Reserve status  = %d </p>"
       "<p>Humidity = %d </p>"
       "<hr>"
       "<p><a href=\"/\">Log out!</a></p>",
       digitalRead( in_sw ),
       255 - humidity
       )
       ;
    server.send
       (
       200,
       "text/html",
       response
       )
       ;
    }

    void
handleNotFound()
    {
    // Send HTTP status 404 (Not Found) when there's no handler for the
    // URI in the request
    server.send
        (
        404,
        "text/plain",
        "404: Not found"
        )
```

```
        ;
    }

// ************************************************************************
// ::: END OF FILE <<-----------------------------------------------------
// ************************************************************************

```

Esquemas de la práctica ('ASCII ART') - esclavo -



Esquemas de la práctica ('ASCII ART') - maestro -



```
                +-+             +         +
                |R|    3.3v o--+-++++++-+---+
                +-+                         |
                 |                          |
                 -                          |
                 V  LED Agua                |
                 -                          |
                 |                          |
                -+--------------------------+-
                                            GND
 O 3.3v
 |
+-+
|R|
+-+
 |
 +--> pin 16
 |
 o
 /
 o
 |
-+-- GND
```

Investigación:

Referencias: