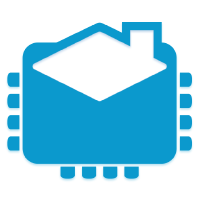
**Estructura del código Souliss**

(Adaptación al Framework actual de un artículo de Juan Pinto)

**[](https://avatars0.githubusercontent.com/u/11448259?v=3&s=200)**

Para esta explicación seguiremos paso a paso con el primer ejemplo de la librería Souliss “e01\_HelloWorld”.

Partiremos de la base de unos mínimos conocimientos para la programación en Arduino (o no, :-), bueno pues por si acaso lo explicaremos TODO:

Al principio del programa, y para aclarar tanto a uno mismo como al resto de usuarios, es aconsejable detallar una pequeña explicación sobre el contenido del programa. 

En él facilitaremos el título, versión, explicación de que hace, que hardware utiliza, y tambien se pueden detallar por ejemplo entradas o salidas reservadas para alguna función concreta. El comentario se inicia con /\* y se finaliza con \*/.

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

    Souliss - Hello World

   This is the basic example, control one LED via a

   push-button or Android using SoulissApp (get it from Play

   Store).

   Run this code on one of the following boards:

      - Arduino Ethernet (W5100)

      - Arduino with Ethernet Shield (W5100)

    As option you can run the same code on the following,

    just changing the relevant configuration file at begin of

    the sketch

      - Arduino with W5200 Ethernet Shield

      - Arduino with W5500 Ethernet Shield

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

Para que el IDE compile correctamente deberemos de iniciar el programa con la siguiente línea:

// Let the IDE point to the Souliss framework

#include "SoulissFramework.h"

Primero observamos una línea comentada "//", no se toma en consideración para el programa. La segunda es obligatoria para poder compilar correctamente, desde la versión 1.6.8, . Hace una supuesta llamada a la librería SoulissFramework, pero esta, es un fichero vacío en el directorio raíz de la librería Souliss.  
  
Seguidamente se incluyen las librerías que utilizará el sketch:

// Configure the framework

#include "bconf/StandardArduino.h"  // Use a standard Arduino

#include "conf/ethW5100.h"          // Ethernet through Wiznet W5100

#include "conf/Gateway.h"           // The main node is the Gateway, we have just one node

#include "conf/Webhook.h"           // Enable DHCP and DNS

**Librería                               Función**

StandardArduino.h        Arduino (Duemilanove, UN, Leonardo o Mega)

ethW5100.h               Utiliza el módulo Ethernet Wiznet W5100

Gateway.h                Para establecer el nodo como Gateway

Webhook.h                Incluye DNS y DHCP

A continuación invocamos la librería SPI (Serial Peripheral Interface), la cual se encarga de las comunicaciones serie.

// Include framework code and libraries

#include <SPI.h>

Y para finalizar con los include llamaremos a Souliss.h, que siempre debe de ser invocada en último lugar, para poder tomar en cuenta el resto de librerías.

/\*\*\* All configuration includes should be above this line \*\*\*/

#include "Souliss.h"

Por último definiremos los Slots que compondrán el NODO. El orden que les asignemos aquí será el que posteriormente nos muestre la APP de Android.

 // This identify the number of the LED logic

#define MYLEDLOGIC  0

Para incluir otro elemento (slot) sencillamente sumaremos uno al slot anterior:

#define MYLEDLOGIC2 1

Y así sucesivamente...  
El nombre "MYLEDLOGIC" se puede modificar evidentemente, aunque he mantenido este para seguir el sketch de ejemplo.  
  
Algunos elementos ocupan DOS SLOTS, por ejemplo, un Sensor de Temperatura o Humedad, necesitan 2 para almacenar los decimales.

En el caso de los sensores el valor del sensor se guarda en el SLOT asignado y en el siguiente, por lo que hay que fijarse en dejar un SLOT libre después de el que hemos declarado (véase que no se utiliza el SLOT 3 porque contiene parte del valor de temperatura).  
  
Añadiendo al ejemplo anterior un SLOT para la Temperatura y otro para la Humedad seguiría así:

#define TEMPERATURE 2 // Identify the temperature logic

#define HUMIDITY    4 // Identify the humidaty logic

Si ahora queremos añadir otro LED:

#define MYLEDLOGIC3 6

Por ejemplo, si ahora queremos añadir un LED del cual queramos controlar la intensidad de luz (PWM) requiere de DOS SLOTS, Es un caso parecido al de los sensores, solo que en este caso hay que declarar un SLOT para el led, y otro para el control del mismo.

#define LEDSTRIP0 7   // This identify the number of the logic on this node

#define LED0      8   // This is the memory slot for the logic that handle the light

Y así consecutivamente hasta tener los SLOTS o elementos a añadir.  
  
Antes de seguir sería necesario explicar que todos los elementos estarían organizados en una serie de Grupos (llamados typical). Tendremos 5 grupos de elementos Típicos:

T1n - Luces

T2n - Motores

T3n - Termostatos o A/A

T4n - Alarmas

T5n - Sensores

Dentro de cada uno hay una serie de elementos (T11, T12,....) y cada uno de ellos tiene un comportamiento o uso, Estos son los que yo más uso:

Typical 11 : ON/OFF Digital Output with Timer Option

Typical 15 : RGB Light (Controlada por infrarrojos)

Typical 16 : RGB LED Strip

Typical 19 : Single Color LED Strip

Typical 51 : Analog input, half-precision floating point (Este es para cualquier tipo de Sensor)

Cada uno de los diferentes SLOT ira asociado a un Típico concreto, por ejemplo, los SLOTs 0, 1 y 6 serán un T11, el SLOT 7 (y serán un T19 y los SLOTS 2 y 4 serán del tipo T51.  
  
Cada uno de los Typical cumple una estructura que es prácticamente común, es decir, tendrán una línea para el Setup, y una o varias para el Loop, pero en definitiva a partir de aquí, solo es cuestión de saber dónde va cada cosa y "montar" el sketch.

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

A partir de aquí empieza el Sketch propiamente dicho.

**SETUP:**

La parte fácil es el void setup(), en este ejemplo se definirán cuatro parámetros diferentes.   
  
  
La Dirección IP:

    // Get the IP address from DHCP  
    GetIPAddress();  
  
Al haber configurado la asignación por DHCP, la IP será tomada directamente del router, si se desea asignar esta dirección manulamente, incluiriamos lo siguiente por encima del void setup ():    
  
  
  
uint8\_t ip\_address[4]  = {192, 168, 1, 77};  
uint8\_t subnet\_mask[4] = {255, 255, 255, 0};  
uint8\_t ip\_gateway[4]  = {192, 168, 1, 1};  
#define myvNet\_address  ip\_address[3]       // The last byte of the IP address (77) is also the vNet address  
#define myvNet\_subnet   0xFF00

Donde modificaremos la dirección IP según nos interese.

El tipo de Nodo:  
  
En DHCP  
  
 SetAsGateway(myvNet\_dhcp); // Set this node as gateway for SoulissApp     
  
En IP fija  
    Souliss\_SetIPAddress(ip\_address, subnet\_mask, ip\_gateway);  
    SetAsGateway(myvNet\_address);    

El Tipycal correspondiente: 

Después solo es cuestión de hacer el SET para cada uno de los nodos según el típico correspondiente, explicado anteriormente.

Para el MYLEDLOGIC declarado antes en el SLOT 0 haríamos lo siguiente.

Set\_SimpleLight(MYLEDLOGIC)

Y la asignación de pines;

Por último debemos configurar todos los PIN's que vayamos a utilizar tanto de entrada como de salida, como haríamos en un sketch normal de Arduino con la función:

pinMode(PIN,OUTPUT); o  pinMode(PIN,INPUT);

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

**LOOP:**

Lo primero a entender del LOOP es que dispone de una organización que permite que las diferentes partes del código se ejecuten en diferentes momentos y con diferentes intervalos de tiempo.

Esto sería la estructura básica:

]

void loop(){

      EXECUTEFAST() {

            UPDATEFAST();

            // De las dos opciones de abajo utilizaríamos la que corresponda según sea GATEWAY O PEER

            FAST\_GatewayComms();

            //FAST\_PeerComms();

      }

      EXECUTESLOW() {

            UPDATESLOW();

      }

}

Y por último dentro de EXECUTEFAST() {} y EXECUTESLOW() {} tenemos la posibilidad de ejecutar determinadas partes del código en determinados momentos, por ejemplo:

Cada 50ms comprueba si se ha pulsado un botón en el PIN 2 y enciende el LED en el PIN 9:

void loop()

{   
    // Here we start to play  
    EXECUTEFAST() {                       
        UPDATEFAST();     
          
        FAST\_50ms() {   // We process the logic and relevant input and output every 50 milliseconds  
            DigIn(2, Souliss\_T1n\_ToggleCmd, MYLEDLOGIC);            // Use the pin2 as ON/OFF toggle command  
            Logic\_SimpleLight(MYLEDLOGIC);                          // Drive the LED as per command  
            DigOut(9, Souliss\_T1n\_Coil, MYLEDLOGIC);                // Use the pin9 to give power to the LED according to the logic  
        }   
                
        // Here we handle here the communication with Android, commands and notification  
        // are automatically assigned to MYLEDLOGIC  
        FAST\_GatewayComms();                                          
          
    }  
}

Para ver los diferentes intervalos que se pueden utilizar en EXECUTEFAST O EXECUTESLOW se pueden ver a partir de la línea 45 de[SpeakEasy.h](https://github.com/souliss/souliss/blob/friariello/base/SpeakEasy.h)   
De esta forma podemos distribuir la ejecución de las diferentes partes del código en intervalos y momentos diferentes.

FAST\_30ms()

FAST\_50ms()

FAST\_70ms()

FAST\_90ms()

FAST\_110ms()

....

Asi como

SLOW\_10s()

SLOW\_50s()

SLOW\_70s()

etc...