Wifi - Display

Introducción

Este dispositivo se compone de una pequeña caja contenedor impresa en 3D, y sus componentes electrónicos. Un pequeño display LCD (LCD1602 2x16 + interface I2C), un adaptador de corriente y un módulo WIFI/WLAN (ESP8266 ESP-12E) para comunicarnos vía internet.

El dispositivo lo controla un sketch (programado con el IDE Arduino 1.8.6 en C++, para procesar los datos y mostrarlos en el display) cargado en el módulo ESP8266 para conectarlo a la Wifi.

Usaremos un módulo creado en Braint4it desde nuestro servidor web para comunicarnos  con el dispositivo vía HTTP REST enviando la información que vamos a mostrar en el wifi-display.



Diseño

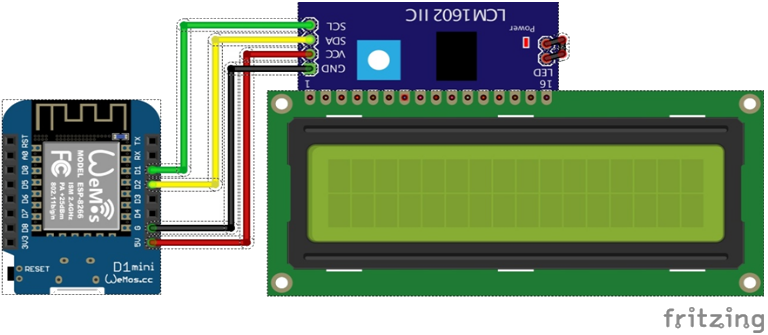
La caja contenedor la forman 2 simples partes (base y tapa). A continuación, puede verlas en 3D.

* Ver modelo 3D
* Base ( ver , descargar STL )
* Tapa ( ver , descargar STL )

# Electrónica

Lista de componentes:

* LCD1602 2x16 + interface I2c.
* D1 mini NodeMcu Lua ESP8266 ESP-12E. Módulo WLAN/Wifi board.
* 4 cables para conexiones, tornillos M3.
* Adaptador DC a 5V 2A.

Aspecto Final Dispositivo en Funcionamiento

Estas imágenes vemos trabajando el dispositivo en sus diferentes fases:

* Buscando la red Wifi para conectarse.
* Identificando la red Wifi, una vez se conecta a ella pasa a modo “esperando datos”.

****





Programando el módulo Brain4it

Función principal del módulo para comunicarnos vía Wifi con nuestro dispositivo ESP8266:

(function

(ctx data)

(local nowdt)

(try

(do

(set nowdt (format-date (date) "yyyyMMdd"))

(### "crea la lista events con todos los eventos que hayan para ese día")

(set events (next-events 50))

(if events

(do

(### "crea la lista eventos con solo los del día en curso")

(set eventos

(find events

e

(and

(>= nowdt (substring (get e 1) 0 8))

(<= nowdt (substring (get e 2) 0 8))

)

)

)

)

)

(### "crea la lista events a partir de la lista eventos, si está vacía => sala disponible")

(if eventos

(set events

(añade-espacios (crear-lista-eventos eventos))

)

)

(### "Pasamos al wimos la lista ‘events’ para visualizarla en el display")

(http "POST" (concat setup/ip\_acceso\_wimos "/ver")

"body" => (string events "format" => "json") )

)

(ex "\*" => (set errorWi (list (format-date (date)) ex)))

)

)

Programando el Sketch para el módulo ESP8266:

Algunas partes del Sketch de C++ para comunicar vía Wifi con el módulo Brain4it de nuestro servidor web, de donde recibimos la información a mostrar en el display:

#include <ESP8266WebServer.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

const char\* ssid = "xxxxxxxx";

const char\* password = "xxxxxxx";

ESP8266WebServer server ( 80);

LiquidCrystal\_I2C lcd(0x27,16,2);

// Definimos estructuras de datos y variables del proceso.

/\*

La tabla que recibe viene como un String y tendrá la siguiente estructura:

"[[fila, colX1, colX2, Mensaje, delay, scroll, Modo], ... ]"

// ejemplos de datos que se reciben en el display

String("[[0,0,16,\"09:00 Reunio del Govern Fins a les 10:00 \",650,1,1]"

",[0,0,16,\"12:30 Reunio Associacions del barris y altres Fins a les 14:30 \",650,1,1]"

",[0,0,16,\"15:00 Conferencia de la dona y el nens ... Fins a les 15:30 \",650,1,1]"

",[0,0,16,\"16:00 Reunio del Govern Fins a les 17:00 \",650,1,1]"

",[0,0,16,\"18:00 Reunio Associacions del barris y altres Fins a les 18:30 \",650,1,1]"

",[0,0,16,\"19:00 Conferencia de la dona y el nens ... Fins a les 20:30 \",650,1,1]]");

\*/

// carga a las variables la cadena recibida con los nuevos datos de la agenda del Brain4it

void ver\_recibido(){

….

}

….

void setup ( void ) {

Serial.begin(115200);

//inicializa el display

….

//se conecta a la wifi

….

//definimos la función que llamaremos al recibir datos del brain4it

server.on("/ver",HTTP\_POST, cargarVariables );

server.begin();

….

}

void loop ( void ) {

// controla que se recibán datos del cliente vía wifi

server.handleClient();

// Hasta que no recibamos datos, no se visualizan. según el modo de visualizar que pasemos(1,2,3,4)

if (datos\_recibidos) {

switch (modo\_visualizar)

{

case 1: visualizar\_modo1();

break;

case 2: visualizar\_modo2();

break;

case 3: visualizar\_modo3();

break;

case 4: visualizar\_modo4();

break;

}

}

delay(50);

}

// Modo 1: visualiza la tabla\_display mensaje a mensaje (el mismo se representa en las 2 filas del display haciendo scroll entre ambas líneas

void visualizar\_modo1()

{

….

salir\_bucle = 1;

}

…

void visualizar\_modo4()

{

}

// imprime en el display posicionando el cursor en las coordenadas dadas

void imprime\_display(int columna, int fila, String mensaje) {

…

server.handleClient(); // detectar si recibimos nuevos datos del brain4it durante la visualización

}