KupeLike

Memoria del proyecto

- **Título:** KupeLike
- **Autores:** Eder Ferreira, Naiara Padules, Jon Auzmendi, Odei Alba, Imanol Hidalgo, Jon Ander Alonso, Julen Maeso, Maria Legaristi, Olaia Martinez
- **Tutor:** Sonia Ortiz de Arri
- **Centro docente:** Mondragon Unibertsitatea
- **Área de participación:** Área de las comunicaciones
- **Edición:** 30ª edición

Índice

Índice 1

Introducción 3

Desarrollo web 5

- Framework Symfony 5
- Tipos de entorno en Symfony 5
- Entorno de diseño 6
- Configuración de la base de datos 7
- Controladores 7
- GPS 8
- Iniciar sesión con Facebook (HWIOAuthBundle) 9
  - Logueo del usuario 9
  - Cómo funciona 9
- Avisos de embotellado 10
- 'Pusher' (Actualización en tiempo real) 10
- Estadísticas 11
- Idiomas 12
- Envío de emails (Swiftmailer) 13
- Panel de administración sidreros 14
- Acceso usuarios 14
  - Vista de sidrero 15
  - Vista de Administrador 15
- Mapas 16
  - Individual 16
  - General 17
- Las rutas de la API 17
- Deploy a Heroku 18

Electrónica 19

- Desarrollo 19
  - Estado de la tecnología 19
- Presupuesto 23
- Descripción técnica 24
- Procedimiento del Montaje 26
  - Diseño y fabricación de la PCB 28
  - Montaje del prototipo 30

Liderazgo Emprendedor e Innovación 33

- Características del sector 33

Tipos de sidra 34
Cifras de la producción 34
Consumo 36

Líneas futuras **37**

Bibliografía **37**

Introducción

La fama de las sidrerías en el País Vasco siempre ha sido divulgada mediante el boca a oreja. Este es un proceso largo y lento que no siempre da resultados, por lo que muchas sidrerías sufren dificultades al darse a conocer. A ello se suma la dificultad de encontrar muchas de estar sidrerías, ya que la mayoría están situadas en lugares recónditos a las afueras de pequeños pueblos.

Una vez en la sidrería, no hay manera de saber cuales son las kupelas (barricas) que más han gustado sin probarlas individualmente.

Tras probar las diferentes sidras y tener una opinión (ya sea positiva o negativa) de ellas, en caso de querer comprar botellas de una kupela en concreto, es necesario pasarse de vez en cuando por la sidrería para preguntar si esa kupela ya ha sido embotellada.

KupeLike es un proyecto para ofrecer todas las facilidades posibles tanto a las sidrerías como a los clientes de las mismas. La aplicación web mostrará un listado, fácilmente accesible, de todas las sidrerías junto a sus respectivas kupelas. Al acceder a la aplicación web, ésta detectará la localización del usuario y en caso de que éste se encuentre en una sidrería, abrirá automáticamente la página de la misma. En caso contrario será dirigido a la página principal de la aplicación. Como su propio nombre indica (Kupela = Barrica, Like = Gustar), el objetivo principal es el de ofrecer al cliente la posibilidad de votar por aquellas kupelas que más le hayan gustado y ver el nivel de satisfacción del resto de clientes sobre dichas kupelas. La información ofrecida por los

clientes, será mostrada tanto en la aplicación web como en unos dispositivos electrónicos conectados vía WiFi que estarán situados en cada una de las kupelas de cada sidrería registrada en nuestra aplicación. Estos dispositivos mostrarán en tiempo real la cantidad de votos de los que dispone la kupela en la que estará situada.

Mediante un sistema de alertas añadido, el cliente tendrá la opción de indicar que desea recibir un aviso al email cuando la kupela que le ha gustado sea embotellada. De esta manera, el sidrero podrá enviar fácilmente dicho aviso a todos los clientes que así lo deseen.

La aplicación también mostrará un mapa donde se mostrarán todas las sidrerías registradas en la aplicación y un mapa interactivo por cada una de ellas, el cual mostrará el recorrido necesario para llegar a la misma. En caso de que el cliente esté accediendo mediante un dispositivo móvil, la aplicación le abrirá el navegador del teléfono con la sidrería como destino.

Por si todo esto no fuera suficiente, también se facilitará la información de contacto de las sidrerías tales como la dirección, el email y el teléfono.

Desarrollo web

Framework Symfony

Utilizamos **Symfony**, un *framework* de **PHP**. La versión que hemos utilizado es la **3.1.9**

Tipos de entorno en Symfony

Entorno desarrollo

Cualquier cambio que se haga en el código podrá visualizarse al momento desde éste entorno.

Para visualizar la aplicación en entorno desarrollo, accedemos a la siguiente URL:

http://[nombre-dominio]/web/app_dev.php

Entorno producción

Para poder visualizar la aplicación en entorno producción (es decir, lo que verá el cliente) Symfony nos obliga a limpiar la caché de producción.

Entorno de diseño

Mockup

El **mockup** de la aplicación se encuentra en la carpeta *Mockup* de Drive.

La aplicación utiliza **Bootstrap** como framework de diseño y **jQuery** como framework de JavaScript.

http://getbootstrap.com/getting-started/

http://api.jquery.com/

Todos **los archivos que necesitamos** utilizar se encuentran en los directorios situados en el directorio ***web/*** del proyecto.

Plantillas Twig

Twig es un sistema de *plantillas* muy potente, el cual viene **instalado por defecto en Symfony** y permite interactuar con los **objetos** y **funciones** del **Symfony**.

```twig
{# app/Resources/views/base.html.twig #}
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

Plantilla base para las vistas, donde en el bloque ***body*** aparecerá dinámicamente el contenido según la página.

```twig
{# app/Resources/views/default/index.html.twig #}
{% extends 'base.html.twig' %}

{% block body %}
    <h1>Welcome to Symfony!</h1>
{% endblock %}
```

Por ejemplo, una vista *index* básica.

Configuración de la base de datos

La configuración para conectarnos a la base de datos se encuentra en el archivo ***app/config/parameters.yml***, donde podemos establecer los parámetros de conexión tanto del entorno **desarrollo** como el de **producción**.

Symfony lee estos parámetros desde los archivos ***app/config/config_dev.yml*** y ***app/config/config_prod.yml***.

Doctrine

Symfony utiliza Doctrine como ORM para la comunicación con la base de datos.

http://docs.doctrine-project.org/en/latest/

Controladores

Los controladores se encargan de procesar las funciones que se llaman desde las rutas de la aplicación.

Estos controladores renderizarán la página necesaria junto con los datos de las sidrerías y/o kupelas necesarias.

Documentación oficial: https://symfony.com/doc/current/controller.html

Tutorial:

https://www.youtube.com/watch?v=GWlH-beIuW0&list=PL-9WnOL7eRJam-XUDSXhyO4b43T35feaA

GPS

A la hora de acceder a la aplicación, **el usuario será dirigido automáticamente a una página que comprobará su posición**. Una vez hecho esto, la aplicación **comprobará si se encuentra cerca de alguna sidrería**.

En caso **negativo**, el usuario será redirigido a la **página principal** de la aplicación.

En caso **afirmativo**, el usuario será redirigido a la **página de la sidrería en la que se encuentre**.

En el caso de que **el usuario decida no revelar su ubicación**, será redirigido a la **página principal** de la aplicación.

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    var perLat = position.coords.latitude;
    var perLong = position.coords.longitude;
    perPos = new google.maps.LatLng(perLat, perLong);
    {% for sagardotegi in sagardotegis %}
        sagarPos = new google.maps.LatLng({{ sagardotegi.latitud }}, {{ sagardotegi.longitud }});
        distancia = google.maps.geometry.spherical.computeDistanceBetween(perPos, sagarPos);

        if (distancia <= 30) {
            id = "{{ sagardotegi.id }}";
        }
    {% endfor %}
    if (id != null){
        var url = '{{ path("sagardotegi_view", {"idSagardotegi":"sagardotegi.id"}) }}';
        url = url.replace("sagardotegi.id", id);
        window.location = url;
    } else {
        window.location = "{{ path('index') }}";
    }
  },
    function (error) {
      if (error.code == error.PERMISSION_DENIED)
          window.location = "{{ path('index') }}";
    });
} else {
    window.location = "{{ path('index') }}";
}
```

Iniciar sesión con Facebook (HWIOAuthBundle)

La aplicación utiliza el Bundle **HWIOAuth**, que nos permite conectarnos mediante nuestro perfil de **Facebook** a **Kupelike**.

Logueo del usuario

Cuando un usuario **pulse en el botón de *"me gusta"*** o el **botón de *"recibir aviso"***de **una barrica** y no tiene la sesión de Facebook iniciada **le pedirá iniciar sesión**, si ya ha iniciado sesión añadirá directamente el voto y en caso de que haya solicitado recibir un aviso, se guardará la solicitud.

Cómo funciona

Desde el cliente **Kupelike** llama a la API de Facebook en el archivo ***/views/Kupela/_facebook.html.twig*** y solicita los permisos necesarios para poder recibir los datos del usuario. Después, envía los datos del usuario mediante *AJAX* al servidor.

```
// obtiene información del usuario y la muestra
FB.api("/me?fields=birthday,name,email,id", function(response){
    // la información se envia en JSON al controlador Kupela
    $('#btn-facebook').click(function(){

        // obtenemos el id de la kupela
        var idKupela = $(this).attr('class');

        // Hace la llamada AJAX
        $.ajax({
            type: "POST",
            dataType: "json",
            url: "{{ path('kupela_like') }}",
            data: {response, idKupela}
        });
    });
});
```

El servidor recibirá los datos y los almacenará para hacer estadísticas.

Avisos de embotellado

Cuando el usuario le dé al botón "Lo quiero" de la cupela que le haya gustado, automáticamente se añadirá un voto nuevo a la kupela y se guardarán los datos de ese usuario para que cuando el sidrero indique que esa kupela ha sido embotellada,

automáticamente reciba dicho aviso. Este aviso automático, contendrá el nombre de la kupela y el nombre de la sidrería a la que corresponde.

'Pusher' (Actualización en tiempo real)

Para actualizar los **datos de los likes en tiempo real**, se ha utilizado el servicio **Pusher**.

Para utilizar este servicio, primero hay que crear una cuenta en su página oficial (https://pusher.com/). Una vez hecho esto, para poder utilizar este servicio en symfony, es necesario instalar el bundle **LopiPusher** añadiendolo en el archivo AppKernel.php.

```
$bundles = [
    new Lopi\Bundle\PusherBundle\LopiPusherBundle(),
    new Symfony\Bundle\FrameworkBundle\FrameworkBundle
```

A la hora de utilizar este bundle, es necesario definir las credenciales que pusher te facilita, en el archivo **config.yml**.

Hecho esto, simplemente hay que crear una **función que será llamada cuando se actualice el número de votos** que una kupela contiene. Esta función, enviará el número de la kupela y la cantidad de votos que contiene al canal de pusher predefinido:

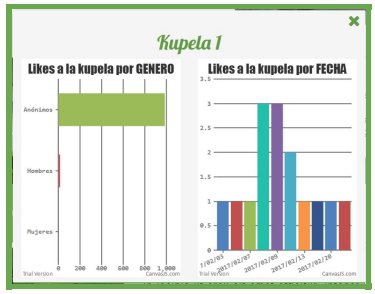
```
public function hacerPusher($nuevoVoto, $id){
    $pusher = $this->container->get('lopi_pusher.pusher');

    $data['message'] = $nuevoVoto;
    $data['id'] = $id;
    $pusher->trigger('my-channel', 'my-event', $data);
}
```

En la página de las kupelas, habrá un código javascript que se encargará de recibir los datos y actualizarlos en la kupela correspondiente.

```
var pusher = new Pusher('a37218af0d3c7dcfceae', {
    encrypted: true
});
var channel = pusher.subscribe('my-channel');
channel.bind('my-event', function(data) {
    document.getElementById(data.id).innerHTML = data.message
});
```

Estadísticas

En la vista de cada kupela, se mostrará un botón llamado “Estadísticas” con el que se mostrarán las estadísticas de dicha kupela, basándose en la fecha de los votos y el género de los usuarios que la han votado.



Idiomas

La aplicación dispone de varios idiomas para facilitar el uso a los usuarios de diferentes nacionalidades.

```yaml
paginas_estaticas:
    menu:
        inicio: "Inicio"
        mapa: "Mapa"
        sidrerias: "Sidrerías"
        contacto: "Contacto"
        nosotros: "Nosotros"
    landing:
        txotx: "Una buena conversación se abre siempre con un Txotx!"
        encuentra: "Encuentra tu sidrería"
    footer:
        inicio: "Inicio"
        mapa: "Mapa"
        sidrerias: "Sidrerías"
        contacto: "Contacto"
        nosotros: "Nosotros"
paginas:
    mapa:
        titulo: "Mapa"
    sidrerias:
        titulo: "Sidrerías"
        llegar: "Cómo llegar"
        contacto: "Contacto"
```

```yaml
paginas_estaticas:
    menu:
        inicio: "Hasiera"
        mapa: "Mapa"
        sidrerias: "Sagardotegiak"
        contacto: "Kontaktua"
        nosotros: "Guri buruz"
    landing:
        txotx: "Elkarrizketa on bat beti Txotx! batekin hasten da."
        encuentra: "Zure sagardotegia aurkitu"
    footer:
        inicio: "Hasiera"
        mapa: "Mapa"
        sidrerias: "Sagardotegiak"
        contacto: "Kontaktua"
        nosotros: "Nor gara?"
paginas:
    mapa:
        titulo: "Mapa"
    sidrerias:
        titulo: "Sagardotegiak"
        llegar: "Jarraibideak"
        contacto: "Kontaktua"
```

Para ello, hay un fichero **messages.<código del idioma>.yml** creado por cada una de las traducciones. Estos ficheros están situados en **app/Resources/translations/**. En

ellos se encuentran todas las frases que dispondrán de traducción. En caso de que un idioma no disponga de la traducción de una frase, se utilizará la traducción del español.

Para poder utilizar estas traducciones, es necesario indicarlo en las rutas. En las rutas de la aplicación (el archivo **app/config/routing.yml**) habrá que especificar que al bundle se accederá con un prefijo (**prefix: /{_locale}**). En las rutas del bundle (el archivo **@KupelikeBundle/Resources/config/routing.yml**) habrá que especificar que la ruta contiene diferentes idiomas en cada una de las rutas que lo requieran.

```
index:
    path: /index
    defaults: { _controller: KupelikeBundle:Index:index }
    requirements:
        _locale: es|eus|en
```

Al utilizar las traducciones en la aplicación, se hará indicando la ruta que la traducción deseada contiene en el archivo de traducciones:

```
<li><a href="{{ path('index') }}">{{ 'paginas_estaticas.menu.inicio' | trans }}</a></li>
```

Después de añadir el fichero messages de un nuevo idioma, es necesario ejecutar el siguiente comando: `php bin/console cache:clear`

Para habilitar el cambio de idioma al usuario, es necesario añadir una etiqueta <a> para cada una de las traducciones, con el siguiente formato:

```
<li class="es"><a href="{{ path(app.request.get('_route'), app.request.get('_route_params')|merge({'_locale': 'es'})) }}">ES</a></li>
<li class="eus"><a href="{{ path(app.request.get('_route'), app.request.get('_route_params')|merge({'_locale': 'eus'})) }}">EUS</a></li>
<li class="en"><a href="{{ path(app.request.get('_route'), app.request.get('_route_params')|merge({'_locale': 'en'})) }}">EN</a></li>
```

Envío de emails (Swiftmailer)

La aplicación consta con la opción de enviar email en diferentes ocasiones. Los email se enviarán desde la pestaña de ayuda (cuando un usuario se quiera poner en contacto con el administrador), desde la pestaña de registo (cuando una nueva sidrería quiera registrarse) y con la opción de embotellado (cuando una kupela haya sido

embotellada). Para poder enviar un email, utilizaremos el bundle Swiftmailer de symfony.

A la hora de enviar el email, recibiremos los datos desde un formulario o desde la base de datos (en el caso de enviar el aviso) y daremos formato a estos datos en el servidor para despué enviarlos. De esta manera todos los emails tendrán un formato universal.

Panel de administración sidreros

La aplicación cuenta con un **apartado de administración** por parte **del sidrero** para poder **gestionar las kupelas** y los **datos de su sidrería**. Pudiendo de esa manera, editar, eliminar y añadir diferentes kupelas.

Por ello, primeramente debemos de crear una nueva tabla en la base de datos, que será llamada **USUARIO**

Acceso usuarios

Existen dos tipos de usuarios en la aplicación; **SIDREROS** y **ADMINISTRADORES**.

Sidreros: Acceden con su cuenta y pueden editar tanto su Sidrería como las Kupelas de la misma.

Administradores: Pueden crear, editar y eliminar nuevos usuarios Sidreros y nuevas Sidrerías.

Vista de sidrero

Una vez haya accedido, al usuario Sidrero se le muestra un menú en la parte superior con las opciones “Mi sidrería” (en la que se le mostrará su sidrería), “Mi perfil” (en la que se mostrarán los datos del usuario para poder cambiarlos), “Ayuda” (servirá para ponerse en contacto con el administrador) y “Salir” (servirá para cerrar la sesión).

En la vista de la sidrería, el sidrero podrá editar los datos de la sidrería, añadir nuevas kupelas, editar los datos de las kupelas existentes (excepto los votos) y eliminar las kupelas existentes.

En la vista del perfil, el sidrero podrá editar sus datos de usuario, tales como el nombre de usuario, la contraseña, el teléfono y email de contacto, etc.

Vista de Administrador

Una vez haya accedido, al usuario Administrador se le muestra un menú en la parte superior con las opciones “Administrar usuarios” (en la que se le mostrarán todos los usuarios), “Administrar sagardotegis” (en la que se le mostrarán todas las sagardotegis), “Ayuda” (servirá para ponerse en contacto con los desarrolladores) y “Salir” (servirá para cerrar la sesión).

En la vista de los usuarios, el administrador podrá añadir nuevos usuarios, editar los datos de los usuarios existentes y eliminar los usuarios existentes.

En la vista de las sidrerías, el administrador podrá añadir nuevas sidrerías, editar los datos de las sidrerías existentes (excepto sus kupelas) y eliminar las sidrerías existentes.

Mapas

Individual

Al acceder a una sidrería específica, mediante las primeras líneas de JavaScript, la aplicación comprobará si se está accediendo desde un dispositivo móvil. En caso afirmativo, se abrirá automáticamente el navegador GPS de google maps dirigiéndose a la sidrería seleccionada.

```
window.onload = function(){
  if (/Android|webOS|iPhone|iPad|iPod|BlackBerry|BB|PlayBook|IEMobile|Windows Phone|Kindle|Silk|Opera Mini/i.test(navigator.userAgent)) {
    window.location = "google.navigation:q={{ sagardotegi.latitud }},{{ sagardotegi.longitud }}";
  }
}
```

La función myMap será la que contendrá el mapa. Aquí se definirá la posición de la sidrería, se solicitará la posición del usuario, se cambiarán los iconos de los marcadores y se añadirán las ventanas de información.

La función calculateAndDisplayRoute será la encargada de mostrar la mejor ruta desde el punto en el que se encuentra el usuario, hasta la posición de la sidrería.

Las funciones CenterControl y CenterControl2 serán las encargadas de añadir los botones para hacer zoom y centrar a la posición de la sidrería y a la posición del usuario.

General

Al acceder a la vista general de las sidrerías, automáticamente se adaptará el zoom y el centro para mostrar la posición de todas las sidrerías disponibles. En este caso, no se mostrará ninguna ruta hasta ninguna sidrería.

Ahora, la función CenterControl2 se encargará de centrar y adaptar el zoom a todas las sidrerías.

Las rutas de la API

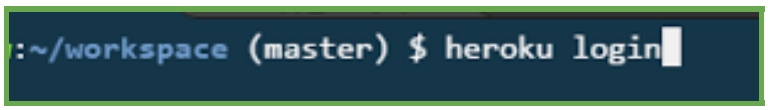
Nombre de la ruta	Ruta	Definición
obtener_likes_kupela	**/es/api/get-likes/{idKupela}**	Devuelve el número de likes de una kupela
agregar_like_kupela	**/es/api/add-like/{idKupela}**	Agrega un voto a una kupela
obtener_datos_sidreria	**/es/api/get-sidreria/{idSidreria}**	Devuelve los datos de una sidrería

Deploy a Heroku

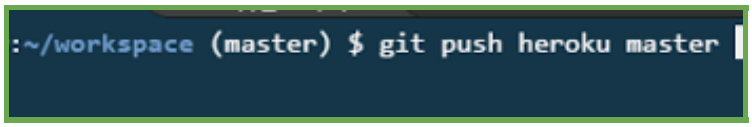
La aplicación se encuentra en producción en el **PaaS Heroku**.

Para hacer un *deploy* de los cambios debemos seguir los siguientes procedimientos:

1. Accedemos con nuestros datos de acceso de *Heroku* **si es la primera vez que nos conectamos desde nuestra máquina**, de lo contrario podemos saltarnos este paso.



2. Con los cambios en nuestro repositorio en el branch master, hacemos el *deploy*.



Heroku automáticamente ejecutará los comandos pertinentes para que la aplicación funcione, como **instalar** los **paquetes necesarios**, **limpiar la caché de producción**, etc.

Electrónica

En cuanto a la parte electrónica, tenemos un producto el cual representa en unos displays todos los me gusta recogidos mediante la aplicación web o en la misma sidrería mediante unos pulsadores. Este estará situado en cada una de las kupelas de la sidrerías con el fin de enseñar y dar la oportunidad de votar las sidras que más gustan.

Desarrollo

Para poder llevar a cabo la recogida y transmisión de datos, hemos creado un producto controlado mediante un microcontrolador (ATMega328p) y un registro de desplazamiento (SAA1064) para poder manejar los displays a nuestro antojo. Para que el producto se conecte con la aplicación web, hemos utilizado un módulo Wifi (ESP8266), y para la recogida de datos en la propia sidrería hemos creado unos botones conectados mediante un módulo Bluetooth (HC-05 y HC-06).

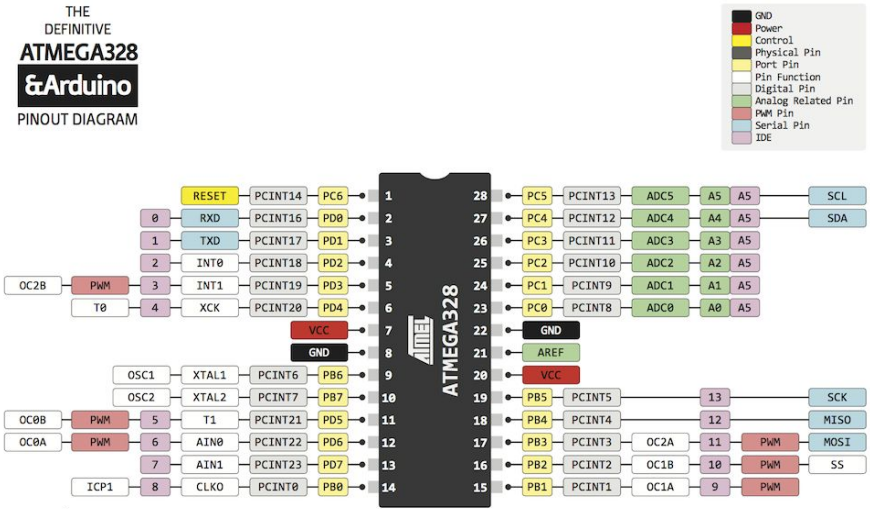
Estado de la tecnología

Como hemos citado anteriormente, esta será la tecnología que utilizaremos para llevar a cabo nuestro proyecto:

ATMega328: Arduino, es una plataforma abierta que se utiliza para la creación de prototipos mediante un software y un hardware muy fácil de usar. Este, tiene la capacidad de recoger información de su entorno mediante sus pines de entrada, y en base a ellos puede controlar diferentes cosas, como pueden ser, luces, motores o otros actuadores.

El ATMega328p, es el microcontrolador que tiene el Arduino y la programación de este se lleva a cabo utilizando un lenguaje especial llamado C++. Es un circuito integrado de alto rendimiento basado en los microcontroladores RISC. Posee 28 pines, de ellos 14 son digitales, 6 analogicos. También posee un cristal de cuarzo de 16MHz en

modo oscilador, la alimentación y el botón de reset entre otras cosas. Se suele programar conectándolo a un ordenador mediante cable USB.

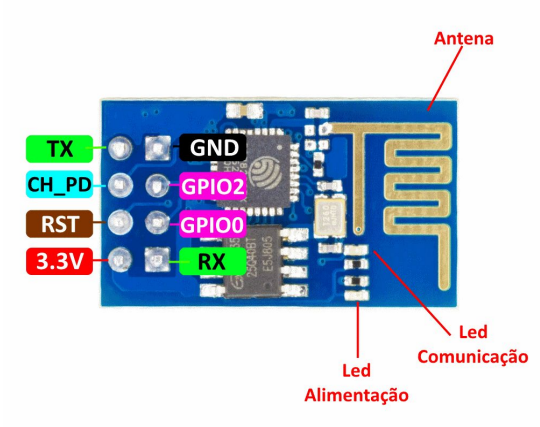


Módulo Wifi ESP8266: El módulo ESP8266 sirve para sumarle conexión Wi-Fi a nuestro Arduino. Esto nos permite controlar nuestro microcontrolador sin necesidad de cables en un entorno de muchos metros, simplemente conectándolo a la red local.

El modulo mide 1,5x2,5 cm y mediante dos conexiones lo conectamos con Arduino, ya que se conecta por serie. No hay una verdadera conexión Wi-Fi pero es suficiente como para poder mandar y recibir datos.

Este módulo, a diferencia de otros, se configura mediante comandos AT. Funciona a 3,3V y a veces consume mucho, con una corriente de 100mA no es capaz de encenderse. Para más información sobre el módulo y los comando AT os dejamos un enlace aquí abajo.

ESP8266

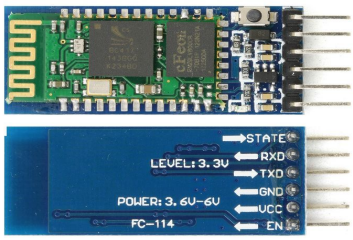


Módulo Bluetooth HC-05: Este módulo nos permite implementar conexión Bluetooth a nuestro Arduino. Al igual que el módulo Wi-Fi, nos permite un control de la placa a distancia sin necesidad de cables.

Sus dimensiones son 28x15x2,35mm y del mismo modo que el ESP8266 se conecta por serie mediante dos conexiones, RX y TX. El módulo puede configurarse de dos formas, en modo esclavo y en modo maestro. En total se compone de 6 pines, dos para la alimentación, otros dos para los anteriormente citados RX y TX y los dos últimos se utilizan para la configuración del módulo.

En cuanto a la alimentación, va de 3V a 6V y las conexiones RX y TX deben ser de 3,3V. El consumo es similar al del módulo Wi-Fi, a veces consume mucho y con poca corriente no es capaz de encenderse pero también hay que decir que una vez encendido esté baja. Para más información os dejamos un enlace aquí abajo.

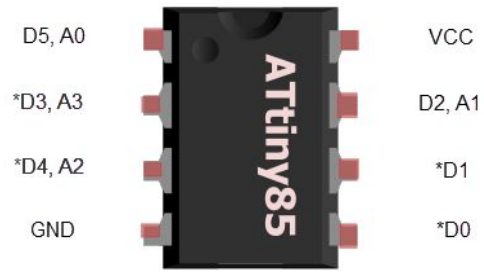
HC-05



Attiny85: Es una plataforma de Arduino pero en un tamaño muy pequeño. El funcionamiento es igual que el de un Arduino normal como puede ser el UNO. Debido al tamaño de este chip, se nos abren muchas posibilidades que no podríamos hacer con un Arduino normal. A parte de eso, no necesita de otros componentes externos para funcionar.

El Attiny tiene un total de 8 pines de los cuales tres son entradas analogicas, dos digitales, la alimentación y el botón de reset. El cristal de cuarzo lo lleva integrado y te da la posibilidad de trabajar en diferentes frecuencias. Para más información os dejamos un enlace aquí abajo.

Attiny

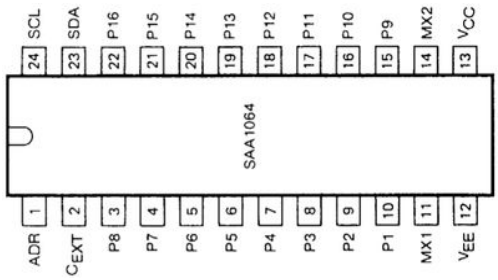


I2C SAA1064: Es un registro de desplazamiento y se caracteriza por que con pocos pines es capaz de manejar y controlar 4 displays a la vez. Tiene un total de 24 pines, de ellos dos se utilizan para la alimentación, otros dos para la conexión SDA y SCL,

18 para el manejo de los displays de 7 segmentos, el ADR o address y el control externo.

Para más información os dejamos un enlace aquí abajo.

SAA1064

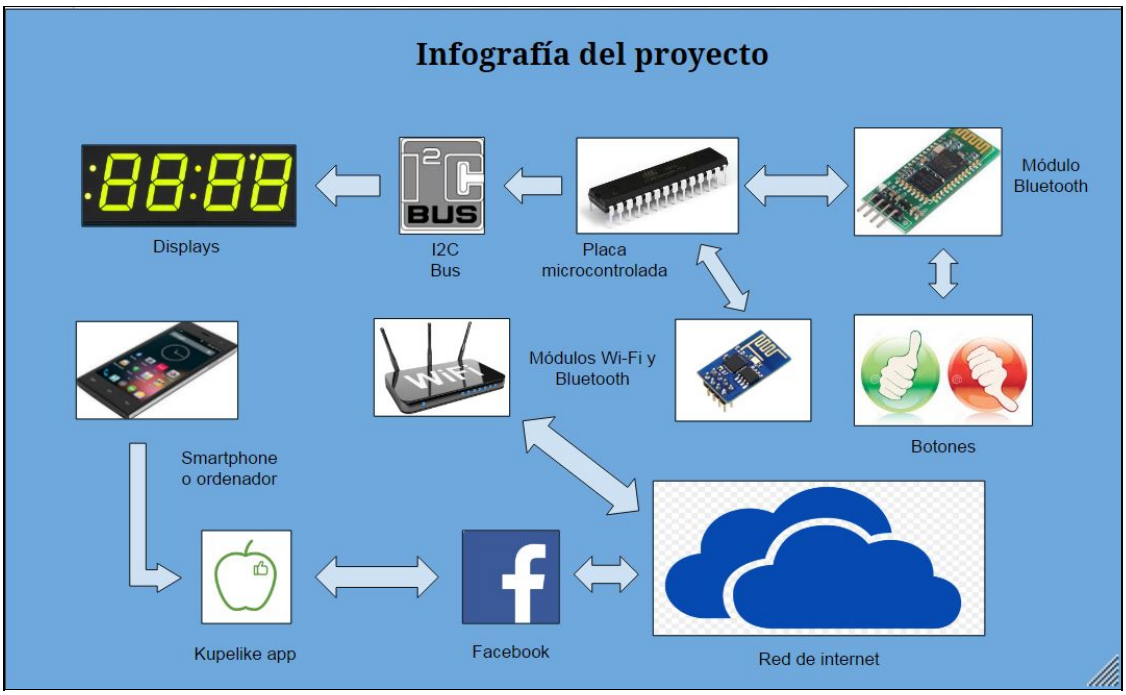


Presupuesto

Producto	Precio (IVA incluido)
ATMega328p	2,49€
Modulo HC-05 x2	12,95€
Modulo ESP8266	3,49€
SAA1064	8,49€
Attiny85	2,69€
7805	0,24€
7812 x3	0,92€
Resistencia 1kΩ x 3	0,09€
Resistencia 10kΩ	0,06€
Cristal de cuarzo 16MHz	3,40€
Condensador 22pF x2	0,04€
Condensador 4700uF	0,48€
Condensador 0.33uF	0,39€
Condensador 2.7nF	0,72€

Condensador 0.1uF x2	0,13€
Puente de diodos B40C	3,80€
Disipador x3	25,12€
Display x4	7,40€
Pulsador x2	2,77€
Conector x8	0,54€
Madera	20,45€
Pintura	8,5€
Total:	**105,16€**

Descripción técnica



Como hemos citado anteriormente, nuestro proyecto se divide en diferentes partes:

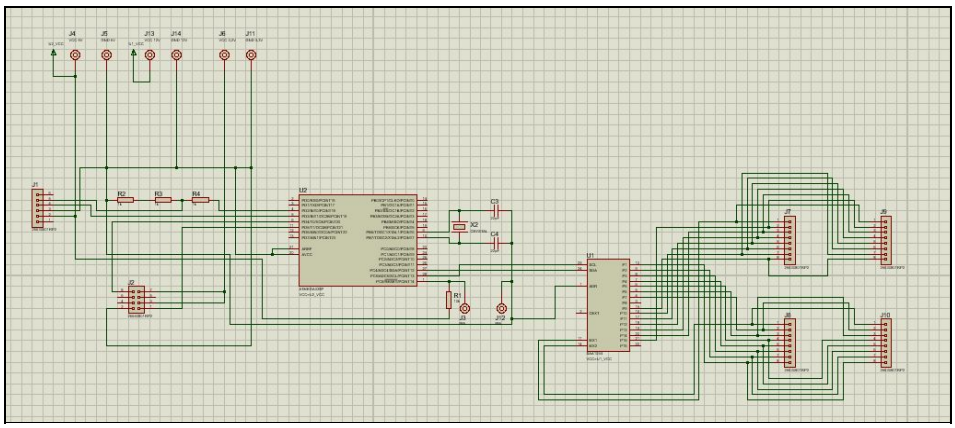
1. Primero, tenemos la aplicación web creada por los alumnos de Zubiri Manteo. En ella aparecerá toda la información sobre las sidrerías y las respectivas sidras de cada una de ella, entre otras muchas cosas. Desde esa aplicación web recogeremos los “Likes” o “Me Gusta” de cada sidra para poder proyectarlas en los displays situados en la mismas sidrerías.
2. En segundo lugar, tenemos el módulo Wi-Fi. Utilizaremos el ESP8266 para añadirle conexión Wi-Fi a nuestra placa microcontrolada y así poder conectarnos a la aplicación web para recibir los datos necesarios.
3. Otro bloque serían los botones. En la misma kupela implementaremos dos botones, uno para votar si te ha gustado la sidra y otro para cuando no te haya gustado, y a raíz de esto implementará o restará un número al contador. Estos botones se conectaran mediante el módulo Bluetooth HC-05 con la placa microcontrolada.
4. El siguiente bloque, y el más importante, es la placa microcontrolada. Esta placa es el cerebro de nuestro producto. Se encarga de recoger los datos recibidos de internet y de los botones y mandarle señales al registro de desplazamiento para poder así proyectar los números en los displays.
5. Por otro lado, tenemos el bus I2C, o el registro de desplazamiento SAA1064, el cual se encarga de proyectar los números en los displays dependiendo de la señal que reciba.
6. Finalmente tenemos los displays, los cuales se encienden dependiendo de la señal que le llega de la placa microcontrolada.

Procedimiento del Montaje

Para empezar con el procedimiento del montaje, lo primero de todo, necesitamos comprobar que los componentes los cuales vamos a utilizar son los correctos y que el funcionamiento del circuito corresponde con lo que esperamos de él.

Para ello utilizaremos un software llamado ISIS, el cual nos ayuda a simular en tiempo real el circuito que estamos creando. Una vez conseguidos los resultados fijados, nos disponemos a recoger los materiales solicitados en la lista anterior.

En nuestro proyecto hemos tenido que simular tres circuitos: Fuente de alimentación, control de Displays y Aumento de likes mediante pulsador.



Montaje y Comprobación de Circuito en la Protoboard

En esta fase, vamos a probar el circuito físicamente en una protoboard. De este modo, comprobamos que todo el material funciona correctamente, ya que el buen

funcionamiento del circuito en la simulación no garantiza que lo haga una vez montado. En caso de que haya algún error, se harán las medidas respectivas para lograr el buen funcionamiento del circuito.

Para conseguir los propósitos establecidos hemos seguido estos pasos:

1. Como bien hemos mencionado anteriormente el primer paso es cerciorarse del correcto funcionamiento. Para ello montaremos el circuito en una protoboard, con mucho cuidado ya que el mal funcionamiento de un componente puede poner en peligro toda la placa.
2. Una vez montado, implementaremos las mejoras necesarias para el circuito, como fusibles para proteger el circuito, y quitaremos los elementos no necesarios.
3. Para finalizar el montaje, crearemos nuestra propia placa, añadiendo las mejoras.

Atmega328p

Como vamos a utilizar un Arduino, vamos a crear nuestro propio shield de Arduino con un microcontrolador Atmega328.

Para este proceso, seguiremos los mismos pasos que en la etapa anterior, pero antes, necesitamos preparar nuestro ATMega desde el software Arduino, procederemos a quemar el bootloader y cargar como programador desde el arduino al microcontrolador deseado.

Para ese proceso comunicaremos el microcontrolador con el arduino con los puertos Rx y Tx, por comunicación serie.

Diseño y fabricación de la PCB

En esta fase vamos a diseñar y fabricar una placa siguiendo diferentes pasos:

1. Diseño del esquema de la PCB.

Para llevar a cabo nuestro esquema no podemos usar ISIS, para ello tenemos que usar otro Software, ARES. Para hacer bien el circuito, crearemos encapsulados, con la medida exacta de los componentes, para que la placa ocupe lo menos posible y sea los más compacta posible.

Como en nuestro proyecto hemos usado Arduino UNO, le vamos a quitar el Atmega328p, y solo vamos a usar ese chip, es decir, vamos a diseñar nuestro propia shield Arduino como hemos dicho anteriormente.

Los encapsulados que hemos hecho nosotros son para el botón, el módulo Bluetooth y para el módulo WIFI.

El problema más llamativo de todos son los pines de alimentación, para eso pusimos etiquetas, para referirnos a la alimentación de cada chip

2. Diseño de la PCB

Para hacer el diseño correcto de una PCB hay que tener en cuenta diferentes factores como en tamaño de las pistas, el posicionamiento de los componentes, el tamaño de los path-s…

Solo hemos diseñado una placa por cada bloque, y hemos ido mejorando dicha placa según nuestras necesidades. El proyecto está separado en tres bloques, el control de los Display-s, la fuente de alimentación y el control del botón. Para nuestra placa microcontroladora hemos creado una similar al Arduino Shield, con los chips que se necesitaran para el recuento de los display-s.

El botón de reset y los pines de alimentación son componentes con mucha importancia. En base a estos componentes se la diseñado la placa, para que todos los componentes estén en bien colocados y de forma ordenada.

3. Fabricación de la PCB

Don Bosco, dispone de una máquina diseñada para fresar las PCB-s, y dicha máquina la hemos utilizado para fabricar nuestras propias PCB-s. La máquina en cuestión, es una fresadora se llama Board-Master LPKF Protomat S62.

Antes de empezar la fabricación hay que crear un archivo que la máquina lo reconozca. Para ello, usamos el programa CircuitCAM. En este programa se recrea la placa real con sus paths y pistas y se hacen los últimos recortes.

Una vez extraido el archivo LMD, hay que conectar el ordenador con la LPKF, cargar los archivos, y configurar la máquina para que empiece a fresar las placas.

Montaje del prototipo

Para que el funcionamiento del prototipo sea adecuado, hay que seguir diferentes pasos:

1.- Soldar los elementos en la PCB

En esta fase, hemos estañado todos los pines, los zócalos… para que los pines que usamos se conecten y desconecten con más facilidad. Por otro lado, hemos colocado los pines de alimentación y el botón de reset.

Para facilitar el posicionamiento de los componentes, primero es conveniente colocar los componentes más bajos, y luego ir poniendo los componentes más grandes o altos. Esto se hace para que a la hora de soldar los componentes no se caigan o se queden mal alineados.

Para hacer soldaduras, hemos usado el taller, y es conveniente utilizar el soldador de punta fina, para lograr un trabajo más detallado, y trabajar más cómodo.

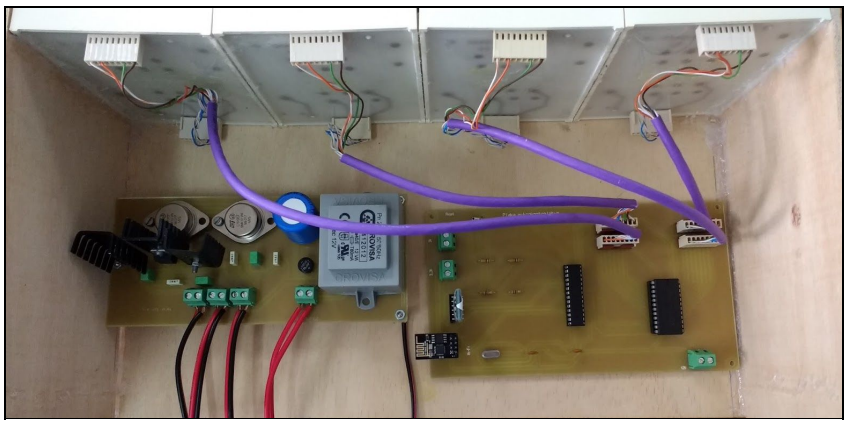
2. Puesta en marcha

Hemos acabado de soldar, el siguiente paso es hacer funcionar nuestro arduino, como el Arduino Shield.

En nuestro caso, hemos comprobado todos los posibles fallos de la placa. Percatandonos que uno de los path de unas pistas específicas, estaban mal fresadas, pero en como en nuestro centro disponemos de una Trimer no tuvimos que hacer la placa otra vez, ya que utilizamos la Trimer para arreglar los path-s de las pistas.

3. Montaje del prototipo

En esta fase hemos juntado todos los anteriores pasos para conseguir un proyecto uniforme. Este proyecto uniforme lo vamos a meter en una caja de madera, para que no se vea ningún cable, para seguir la estética de las sidrerías.



Montaje de la caja y soporte

Para seguir la estética de las sidrerías y que no se vean mucho los cables, hemos pensado en meter todo en una caja de madera. Para hacer la caja, hemos calculado todas las medidas y las hemos escrito en un papel. Para hacer los cortes necesarios de la madera, nos han facilitado una sierra caladora y con ella cortamos la madera en las medidas que calculamos.

Para poder juntar las paredes de la caja hemos usado cola termofusible.

Después de hacer la caja, hemos hecho una media barrica con la madera que nos sobró, para darle una buena imagen al prototipo y la gente vea como va a quedar en las kupelas. Hemos utilizado una madera fina para hacer la medialuna y hemos hecho un soporte para que la caja se sostenga.

Pensamos que era conveniente tener dos maderas diferentes, una fina para las paredes y la tapa superior y una más gorda para la base de la caja, ya que la base tiene que soportar todo el peso del prototipo.

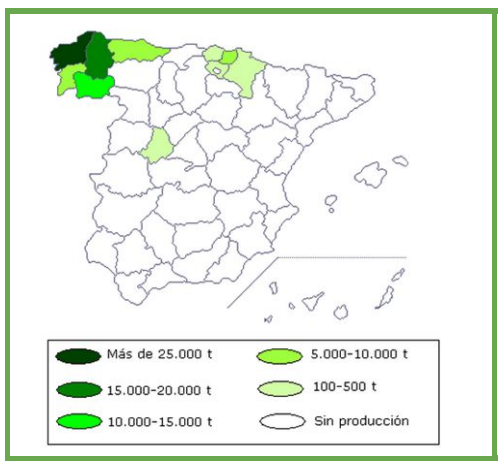
Liderazgo Emprendedor e Innovación

Características del sector

La sidra constituye un emblema dentro de la gastronomía y la cultura del norte de España. En concreto, la producción de manzana y la elaboración de sidra constituyen una de las principales actividades económicas de la agricultura de Gipuzkoa, formando parte de los productos agroalimentarios más importantes en volumen de negocio (unos 78 millones de euros al año).

Desde sus inicios, allá por la época romana, el producto ha ido evolucionando hasta convertirse en un elemento insustituible en la promoción turística como en la fortaleza socioeconómica de la región. Sin embargo, su producción y comercialización aún siguen siendo muy tradicionales.

El cultivo del manzano de sidra se localiza principalmente en las regiones de la cornisa cantábrica (Asturias, Galicia, País Vasco y Navarra).



Tipos de sidra

Se debe distinguir claramente dos líneas de productos:

- La sidra natural tradicional
    - Bebida fermentada de manzana proveniente de 22 variedades de manzanas
    - Aspecto amarillo (paja) intenso
    - Sabor intenso à fondo ácido con leve tono dulce
- La sidra gasificada
    - Gas endógeno que se consigue a través de la fermentación
    - Seca, tipo brut
    - Color amarillo pálido con burbuja fina
    - Más para celebraciones navideñas

Cifras de la producción

La producción total de sidra en España se acerca a los 70 o 75 millones de litros, de los que el 55% corresponden a la sidra natural y el resto a la espumosa. La producción se concentra fundamentalmente en Asturias, abarcando el 80% de toda la sidra elaborada en España. Aunque no es un producto exclusivo de Asturias ya que posee importantes competidores en el campo de la sidra natural, como es el País Vasco. El País vasco reúne a 70 productores de los cuales 58 se hallan en la Asociación de Sidreros de Gipuzkoa.

En el País Vasco se elabora Sagardoa (vino de manzana), una bebida fermentada de manzana natural, sin adición ni de azúcar ni carbónico. En la actualidad hay alrededor de 90 bodegas que elaboran en torno a los 12 millones de litros de sidra natural. Aunque

principalmente se elabora en Guipuzcoa hay sidrerías de elaboración en todas las provincias del País Vasco.

En cuanto, a los datos de la campaña 2014-2015 hay que destacar que se elaboraron 841.000 litros.

Hasta finales del siglo XIX se producía abundante sidra en las seis provincias vascas pero hoy en día, el 99% de las bodegas están situadas en Guipúzcoa (se produce algo en Vizcaya y Álava). La sidra o "sagardoa" que se elabora en Guipúzcoa, es muy similar a la de Asturias. Su mayor diferencia radica en el proceso de elaboración ya que en la vasca se trasiega menos. En cuanto al producto, la vasca es más ácida por las variedades de manzanas utilizadas y porque al consumidor vasco también le gusta con más "garra".

La producción de destina, casi toda, al autoabastecimiento aunque dentro de los hábitos de consumo, la sidra vasca se bebe más como acompañamiento de las comidas que bebida esporádica, bebida social o de "poteo" como es la asturiana. Por lo que uno de los objetivos de la sidra vasca, es lograr sacarla de las sidrerías y colocarla en la barra de los bares.

	HOGARES		RESTAURACIÓN COMERCIAL		RESTAURACIÓN COLECTIVA Y SOCIAL		TOTAL	
	CONSUMO	GASTO	CONSUMO	GASTO	CONSUMO	GASTO	CONSUMO	GASTO
Agua Envasada	2.345,20	496,70	837,60	336,20	105,40	62,60	3.288,20	895,50
Bebidas espirituosas	44,00	433,10	153,10	1.912,60	1,10	11,30	198,20	2.357,00
Refrescos y gaseosas	1.966,70	1.561,50	1.005,90	1.908,40	38,00	53,00	3.010,60	3.522,90
Cervezas	752,40	890,60	1.540,50	2.971,50	20,00	31,70	2.312,90	3.893,80
Sidra	13,80	23,60	11,60	18,80	0,10	0,30	25,50	42,70
Vinos	432,80	1.039,00	324,50	1.270,20	8,90	20,60	766,20	2.329,80
Zumo y néctar	526,90	463,40	85,70	170,00	28,60	32,20	641,20	665,60
TOTAL BEBIDAS	6.081,80	4.907,90	3.958,90	8.587,70	202,10	211,70	10.242,80	13.707,30

Millones de litros / Millones de euros

Consumo

Como se ve en la tabla, el grupo de productos suma un gasto total de 13.707 millones de euros (datos de 2015), de los cuales el 35,8% se concentra en el hogar y el 62,6% en la restauración comercial.

Hay bebidas cuya demanda se asocia al hogar como el agua, los refrescos y la cerveza. Está claro que la sidra tiene un consumo inferior al del vino o la cerveza, no obstante, aunque la sidra represente solo un 0,31% del consumo en bebidas respecto al total (25,5 millones de litros), que supone una cuota de mercado de 42,7 millones de euros de gasto total realizado por los españoles durante el 2011. Este gasto se concentra en Asturias, País Vasco, Cantabria y Navarra (las principales consumidoras).

Líneas futuras

- Crear estadísticas de los clientes de las sidrerías, mostrando la edad, el origen, el sexo, etc. con los datos recibidos por facebook (Para integrar esto en la aplicación es necesario solicitar permiso a los desarrolladores de facebook).
- Añadir más información sobre cómo llegar a las sidrerías (Autobús, andando, etc.).
- Añadir opción de compra online.
- Añadir opción de reserva online.

Bibliografía

- Repositorio de GitHub
- https://kupelike.herokuapp.com/
- heroku
- Pusher
- Cloud9
- Facebook Developers
- Google Developers
- Symfony
    - **Documentación**
    - **Libro de referencia**
- Bootstrap
- Filestack
- Pushbutton Bluetooth 1
- Pushbutton Bluetooth 2
- Pushbutton Bluetooth 3
- Bluetooth Hc-05
- Programa Bluetooth Slave
- Programa Bluetooth Master
- ESP8266 AT comandos
- Conexiones Arduino
- Soluciones de errores
- Primeras Conexiones
- Programas
- ESP8266 Alternativa