Water Balance Saver

MEMORIA FINAL

Duarte Traseira Santín
Miguel Ángel San José Alarcón
Rafael Alvear González
Marta del Olmo Collado
Mónica Jiménez Muñoz
Manuel Pérez García

ÍNDICE

1.- INTRODUCCIÓN

Es muy probable que en numerosas ocasiones tu factura del agua se haya ido un poco más allá de tus expectativas o de tu media mensual. Además, es posible que alguna vez tengamos una pérdida de agua en algún lugar de la casa y tardemos demasiados días en darnos cuenta. Por ello, desde WATER BALANCE lanzamos el nuevo WATER BALANCE SAVER.

El objetivo del producto es poder dotar al usuario de la capacidad de controlar el consumo doméstico de agua para evitar ese gasto de agua innecesario y, así, ayudar a la sostenibilidad de nuestro planeta. Sin duda, la gran ventaja que aporta nuestro producto respecto a otros competidores es el fácil acceso a los datos a través del teléfono móvil así como el aviso instantáneo en caso de que se produzca alguna pérdida de agua no deseada en nuestro domicilio.
El nuevo WATER BALANCE SAVER está formado por un caudalímetro acompañado del hardware necesario para ser capaz de enviarnos cierta información a través de una app. Dicho hardware está formado por un sensor de temperatura, una batería, una pantalla LCD, un ESP-32 con bluetooth y una caja que permite contener y aislar adecuadamente todos los componentes. La información, que se envía a través del bluetooth del ESP-32 a nuestro móvil, se puede visualizar a través de una app y muestra, además del consumo de agua en l/min, otro tipo de información como puede ser la temperatura del agua.

Nuestro producto está diseñado para poderse instalar en las diferentes habitaciones de una vivienda donde se consuma agua (normalmente cocina y baño). En la cocina, por ejemplo, nuestro caudalímetro puede estar conectado directamente a la red por lo que no necesita batería. Además, si se usa en un electrodoméstico, se colocaría por la parte de atrás, así que al no estar visible también podemos prescindir de la pantalla.

Sin embargo, el producto está más concretamente diseñado para controlar nuestro consumo de agua en la ducha. En ese caso sí debemos contar con todo el hardware disponible (pantalla y batería) para obtener una mayor experiencia de usuario, de tal forma que se pueda visualizar el consumo de agua y, como valor añadido a nuestro producto, la pantalla mostrará también la temperatura del agua y será capaz de notificarnos mediante un diodo LED si sobrepasamos cierto límite de consumo de agua establecido en 45 litros, que es el gasto medio de agua que utiliza una persona al ducharse.

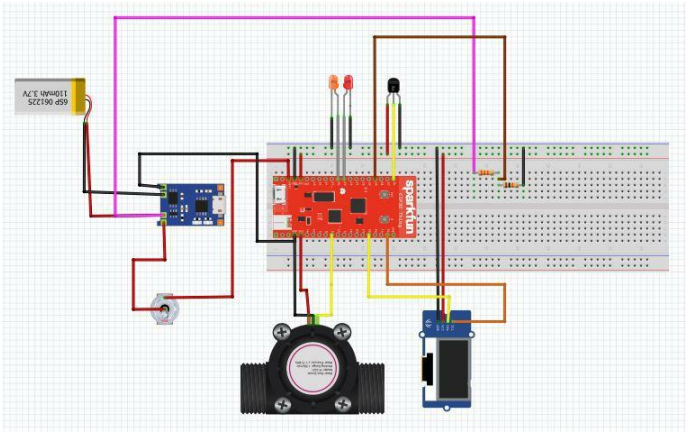
2.- PROTOTIPO

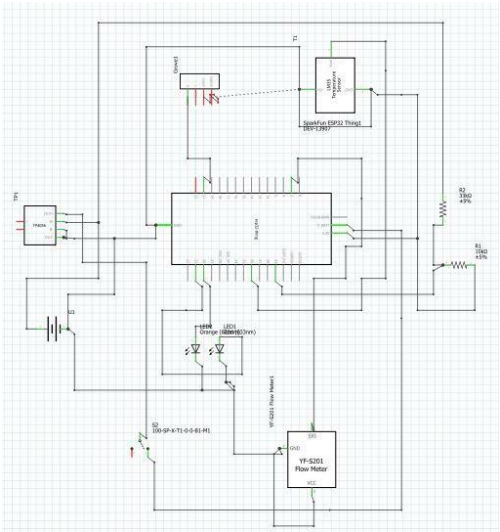
2.1 FUNCIONALIDAD

La funcionalidad de nuestro prototipo se ha visto ciertamente limitada. Actualmente disponemos de un único prototipo capaz de….

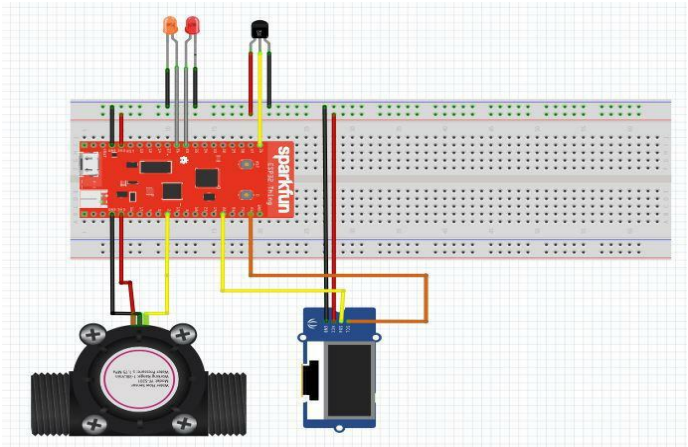
2.2 ESQUEMA ELÉCTRICO

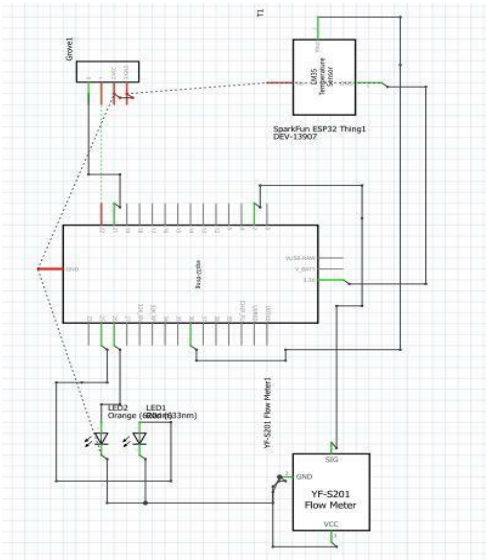
- **PROTOTIPO CON BATERÍA**





- **PROTOTIPO ENCHUFADO**



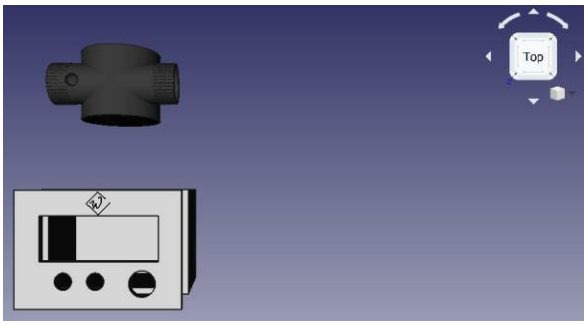


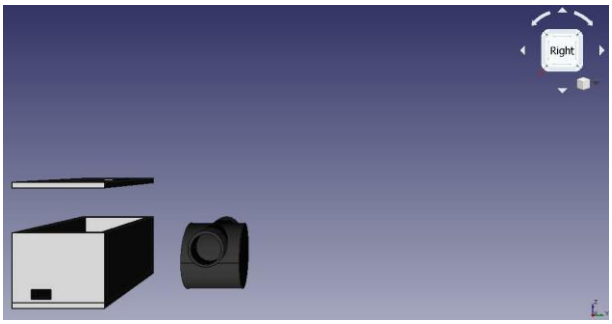
2.3 PLANOS, PIEZAS, DISEÑOS…

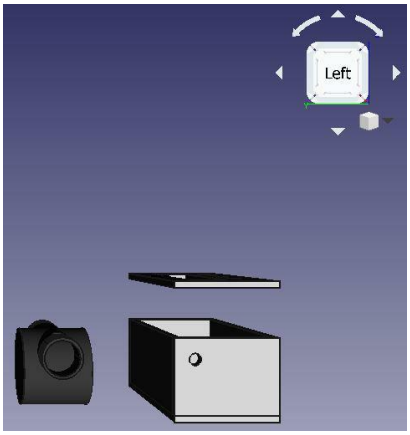
Como en todos estos apartados dividiremos los puntos en dos debido a que tenemos dos tipos de prototipos.

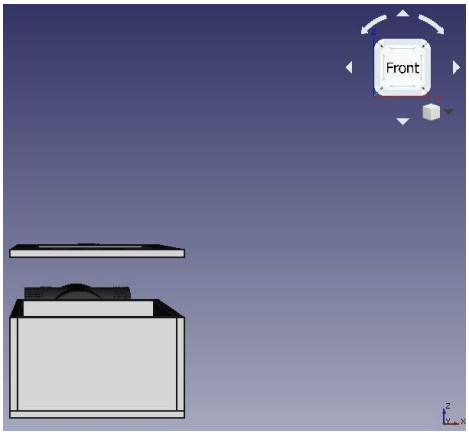
- **PLANOS**

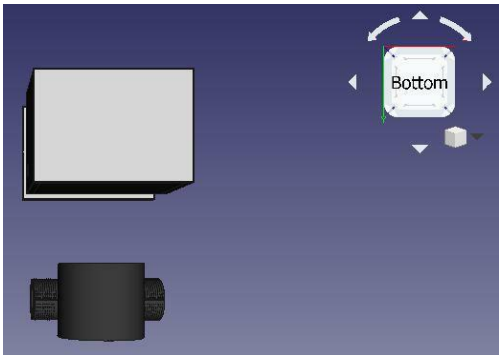
➔ Prototipo con batería

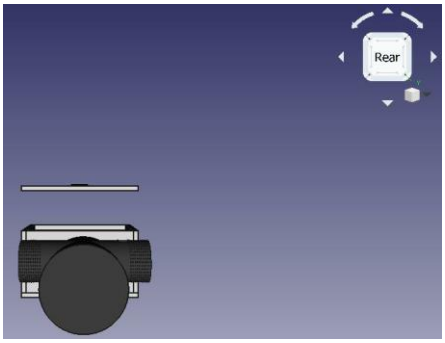




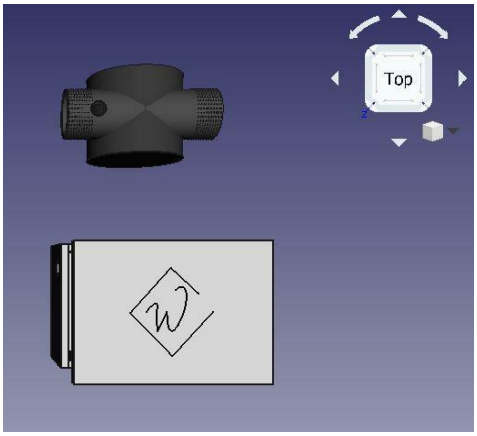


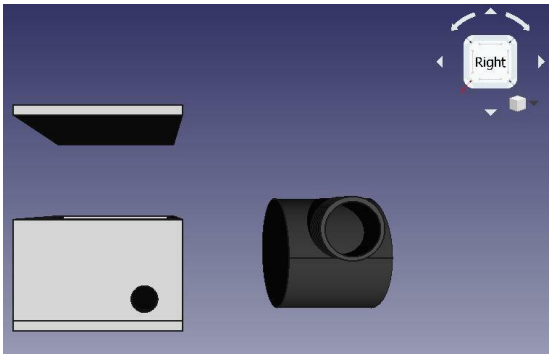


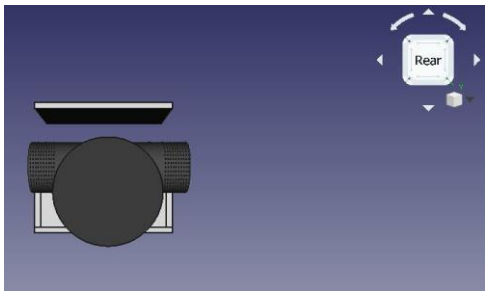


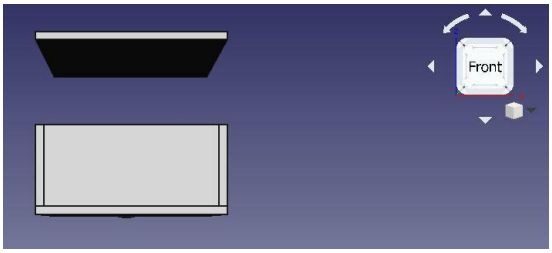


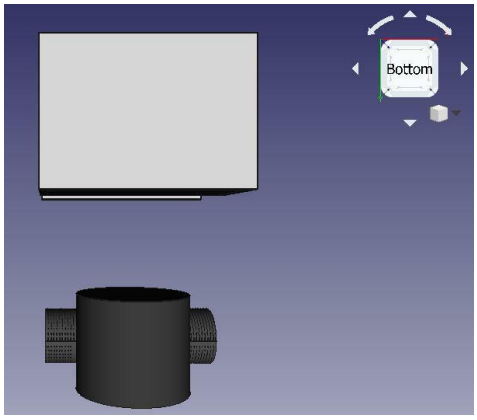
➔ Prototipo enchufado

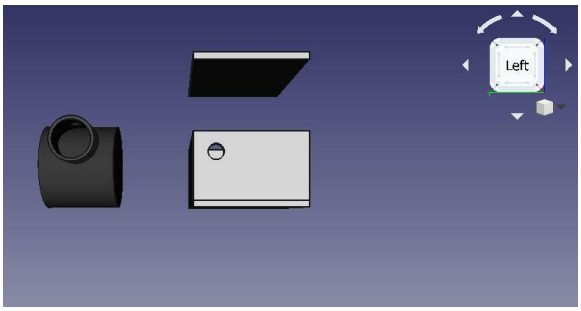










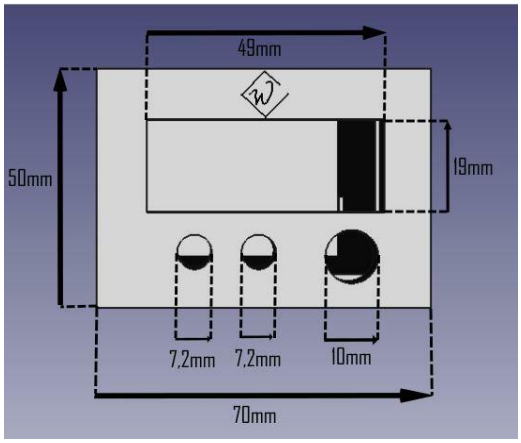


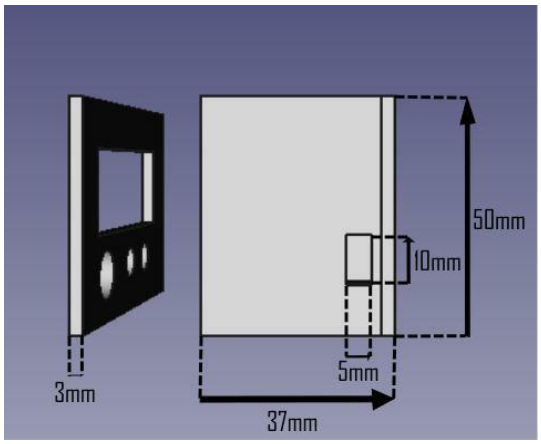
- **PIEZAS**

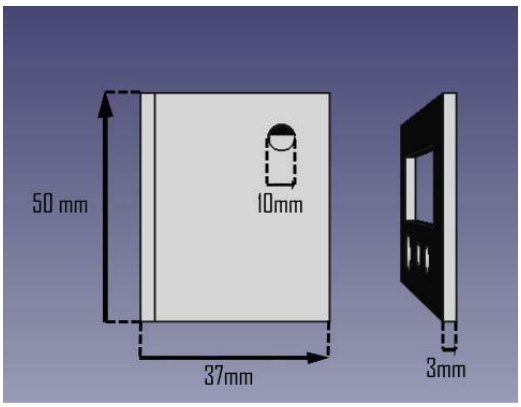
En ambos prototipos se representa perfectamente en los planos que están formados por dos piezas unicamente.
La primera pieza esta formada por la unión de la base y todos los laterales. Se imprimirá mediante una impresora 3D como una única pieza. Dentro de ese “recipiente” colocaremos con especial cuidado toda la electrónica que hemos tenido que utilizar para nuestro producto final.Por otro lado imprimimos del mismo modo la segunda pieza que es la tapa.
Una vez que ya tenemos todo colocado sellaremos las dos piezas para que no entre agua dentro de nuestro prototipo, y además taparemos las conexiones necesarias del sensor de temperatura y del caudalímetro para evitar también que pueda entrar agua.

- **MEDIDAS**

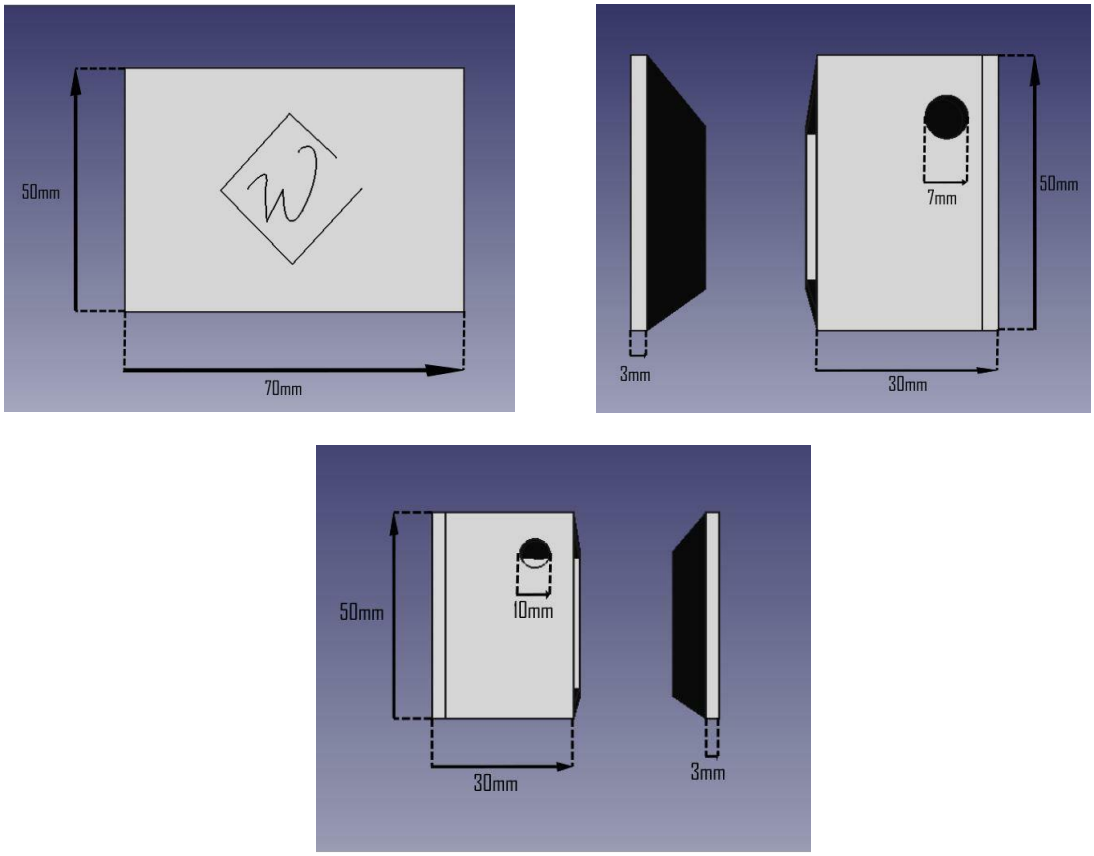
➔ Prototipo con batería







➔ Prototipo enchufado



2.4 FUNCIONAMIENTO

El funcionamiento del producto es relativamente sencillo, y lo dividiremos en dos partes, obtención de las medidas, la comunicación entre dispositivos y la comunicación vía bluetooth:

Obtención de medidas: Vamos a disponer de un caudalímetro y un sensor de temperatura. El funcionamiento del caudalímetro es relativamente sencillo, ya que, al cruzar agua a través de éste, hará girar el mecanismo rotatorio que tiene en su interior. Esta rotación generará un pulso de una cierta frecuencia, con la cual podremos calcular el volumen de agua y el caudal que estará atravesando el caudalímetro por según. Obtenemos las medidas sensor de temperatura mediante el método getTemperature(), que nos convertirá la lectura analógica del pin e grados Celsius. Por último, tendremos dos leds, los cuales nos avisarán si estamos consumiendo mucha agua, o si el nivel de la batería es bajo.

Comunicación entre dispositivos: Para comunicar un producto con otro, hemos utilizado un protocolo diseñado por la compañía Espressif para los esp32. Se trata del protocolo esp-now. Lo que hacemos es configurar los esp32 como si fueran una estación Wi-Fi, capaz de enviar y recibir datos en la banda de los 2.4 GHz. Este protocolo es muy útil ya que podemos enviar hasta 250 Bytes a una distancia de 220 metros. Nos hemos decantado por una configuración de esclavo-maestro, ya que lo que queremos conseguir es enviar todos los datos de los diferentes esp32 a través de un único dispositivo.

Comunicación vía bluetooth: Para enviar los datos obtenidos por el maestro y los esclavos, lo que haremos crear y configurar al maestro para que envié vía bluetooth los datos a la aplicación.

2.5 SOFTWARE

Vamos a comentar los métodos y las líneas de código más importantes de los caudalímetros, tanto del maestro como del esclavo:

typedef struct struct_message : estructura que debe de ser igual tanto en el maestro como en el esclavo. Esta estructura es muy importante ya que es la encargada de almacenar los datos que se envían al maestro.

void setup() : En este método lo que vamos a hacer es asignar los leds, el sensor de temperatura y el caudalímetro como entradas o salidas, vamos a inicializar la pantalla OLED, configurar la interrupción externa causada por el caudalímetro, configurar el esp32 como si fuese una estación Wi-Fi capaz de enviar y recibir datos mediante la línea de código **WiFi.mode(WIFI_STA),** inicializamos el protocolo esp-now, y en el caso del esclavo, registramos y añadimos la mac address del maestro, para así tener una dirección a la que enviar la información, y en el caso del maestro, lo que haremos será asignarle un nombre a nuestro Bluetooth.

void loop() : Haremos una llamada a lecturas() y a representaDatos(), para obtener y representar los datos cada 2 segundos. Aquí hay que remarcar dos cosas, ya que en el esclavo enviaremos los datos al maestro mediante la línea de código:

esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &misDatos, sizeof(misDatos));

Y en el caso del maestro, enviaremos la información cada dos segundos a la aplicación mediante las líneas de código **ESP_BT.print();**

void ISRCountPulse() : Método que actualiza el contador asociado a la rutina de atención a la interrupción.

float GetFrequency() : Es el método encargado de convertir el movimiento rotatorio del caudalímetro en una frecuencia para así poder calcular el resto de valores.

void SumVolume(float dV) : Método encargado de calcular el volumen de agua que cruza el caudalímetro por segundo.

float GetTemperature() : Método encargado de leer el sensor de temperatura y su conveniente conversión para hallar la temperatura en grados Celsius.

float GetBattery() : Método encargado de obtener el porcentaje de batería restante.

void lecturas() : Método encargado de obtener el caudal, y la temperatura. Las almacenaremos en flow_Lmin y temp, y estas dos variables son importantes ya que serán las encargadas de almacenar los valores, y posteriormente usados para enviarlos a través del protocolo ESP-NOW.

void representaDatos() : Es el método encargado de representar los datos por la pantalla OLED del producto.

Ahora vamos a ver las líneas de código propias del esclavo:

uint8_t broadcastAddress[] = {0xA4, 0xCF, 0x12, 0x9A, 0x02, 0x50}: Esta línea de código es de las más importantes ya que es la dirección del maestro al que enviaremos las medidas del caudal y de la temperatura.

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status): Este método nos imprime por la pantalla del terminal de Arduino si se ha enviado correctamente la información.

Ahora procederemos a ver las líneas de código propias del maestro:

BluetoothSerial ESP_BT : Creamos el objeto bluetooth para poder así enviar la información a la aplicación.

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len): Método que nos va a imprimir por pantalla los datos que nos han enviado, y la longitud del paquete en el que se envían.

3.- PRODUCTO

3.1 FUNCIONALIDAD

La funcionalidad del producto final, como ya se ha mencionado en la introducción, consiste en el control del consumo doméstico de agua a través de nuestro teléfono móvil.

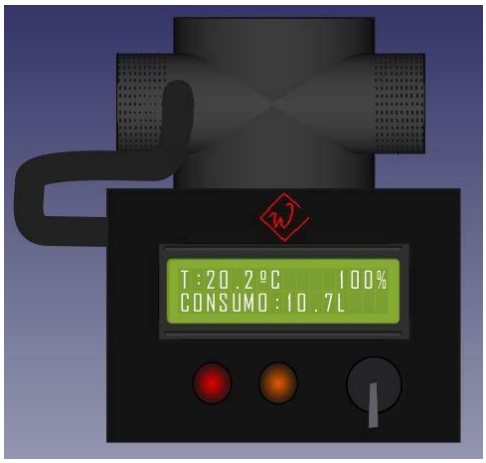
Es evidente que en una casa existen diferentes lugares o electrodomésticos que consumen agua. Sin embargo, el WATER BALANCE SAVER ha sido diseñado, en un principio, para ser instalado en una ducha o una bañera y por eso cuenta con una pantalla LCD que nos muestra los diferentes datos de manera instantánea. Así podremos conocer nuestro consumo de agua sin necesidad de consultar el teléfono móvil.

Si bien es cierto que el producto final se ha orientado al consumo de agua en la ducha o bañera, también disponemos de productos adaptados para electrodomésticos que estén en la cocina, por ejemplo. Es decir, se ha prescindido del hardware innecesario como puede ser la pantalla debido a que el WATER BALANCE SAVER no se encontraría en un lugar accesible para el usuario.

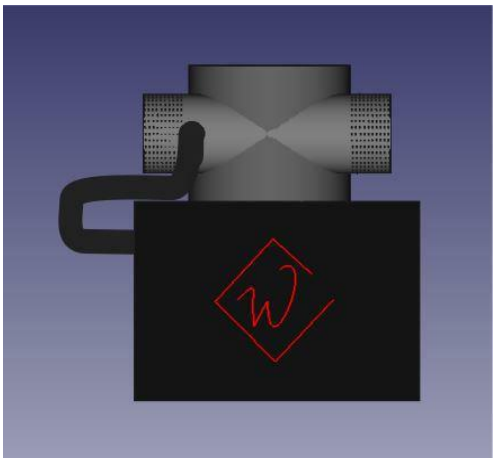
A pesar de ello, en nuestro teléfono móvil podremos recibir la información procedente de los diferentes dispositivos instalados en nuestra casa, tanto por separado, como en cómputo global.

3.2 ASPECTO FINAL

Nuestro producto dispone de dos tipos de cajas, uno con batería para poder instalar en las duchas y otro que va directamente enchufado a la corriente que medirá si existen pérdidas como por ejemplo en las lavadoras.



Caja con batería



Caja enchufada

3.3 FUNCIONAMIENTO PREVISTO

El funcionamiento del WATER BALANCE SAVER gira en torno a nuestro ESP32.
En primer lugar, recoge los datos recibidos a partir del caudalímetro físico, el cual es capaz de medir el agua que pasa a través de él hasta un caudal máximo de 30 litros por minuto. Por otro lado, es capaz de gestionar las diferentes actividades a realizar para ofrecer la información al usuario:

- Gracias a su conectividad bluetooth es posible enviar los datos a un teléfono móvil y así poder visualizarlos a través de nuestra app.
- También se encarga de enviar los datos a la pantalla LCD de manera instantánea.

Gracias a todas las acciones previamente descritas es posible conseguir una gran experiencia de usuario y así, satisfacer sus necesidades.

3.4 COSTE ESTIMADO

El coste estimado de nuestro producto se ha calculado como la suma del coste de todos aquellos materiales o dispositivos necesarios para la construcción del producto final. Cabe

destacar que los materiales para un solo producto final elevan el coste del mismo considerablemente. Estos son:

COMPONENTE	CANTIDAD	PRECIO/ UNIDAD	PRECIO TOTAL	ORIGEN
Caudalímetro	1	4,90 €	4,90 €	BricoGeek
ESP-32 (placa de desarrollo)	1	23,50 €	23,50 €	BricoGeek
Batería LiPo 300mAh	1	3,95 €	3,95 €	BricoGeek
Cargador de batería	1	1,50 €	1,50 €	BricoGeek
Pantalla 128x32	1	1,42 €	1,42 €	AliExpress
LM35 (sensor de temperatura)	1	0,30 €	0,30 €	AliExpress
Protoboard	1	1,95 €	1,95 €	BricoGeek
LEDs	2	0,01 €	0,02 €	AliExpress
Cable cargador	1	3,95 €	3,95 €	BricoGeek
Interruptor	1	0,90 €	0,90 €	AliExpress
Caja water resist	1	0,20 €	0,20€	Fabricante piezas 3D

Los enlaces correspondientes a los componentes del producto final se encuentran en el Anexo I.

Los precios previamente detallados se corresponden a lo que costaría fabricar una sola unidad de Water Balance Saver. Sin embargo, en nuestro caso, como haremos una producción en masa, se comprarían los componentes en grandes cantidades, lo que haría que este coste se reduzca aproximadamente en un 30% siendo los siguientes costes los finales los especificados en la siguiente tabla.

MODELO DE PROTOTIPO	CON BATERÍA	SIN BATERÍA
COSTE TOTAL (1 UD)	42,59 €	37,14 €
COSTE TOTAL (PRODUCCIÓN MASIVA)	29,81 €	26 €

3.5 PRECIO DE VENTA ESTIMADO

Para establecer un precio de venta estimado para el producto es necesario basarse en el coste fijo de producción estimado en el apartado anterior. A este coste, es necesario añadirle un margen en el que se incluyen:

- Montaje.
- Programación software.
- Mantenimiento de la aplicación y página web.
- Gastos operativos (ventas, transporte, etc.).
- Beneficios de la empresa.

Los cuatro primeros puntos mencionados previamente provocan una subida de un 40% en el precio de venta respecto al coste estimado. Además, debemos sumarle un 20% que son los beneficios correspondientes que recibe la empresa por cada producto. Por ello, el precio final de venta aumentará un 60% respecto al coste de producción. Los precios de los distintos modelos serían los presentados en la siguiente tabla.

MODELO DE PROTOTIPO	**CON BATERÍA**	**SIN BATERÍA**
PRECIO FINAL	48 €	41,50 €

3.6 CANALES DE FABRICACIÓN, VENTA Y DISTRIBUCIÓN

La venta y distribución de nuestro producto se realizará por Internet a través de:

- Amazon: principal canal de venta y distribución.

- Página web propia de la empresa: http://waterbalance.emiweb.es/

4.- CONCLUSIONES

Debido a la situación en la que vivimos actualmente, se nos ha hecho muy difícil ya que no hemos podido profundizar más en la parte del hardware y del software, por falta de materiales que no han llegado a nuestro alcance. Por ello creemos que se podrán plantear ciertas mejoras:

1.- Mejora de la aplicación.

2.- Conexión del maestro a Internet para así poder tener a todos los dispositivos que estén conectados o tenga batería monitorizados.

Aún así se ha intentado realizar lo máximo posible con las herramientas que teníamos a mano (esquemas,planos,piezas,diseño…), para poder sacar nuestro producto y que se pudiera

mostrar la intención que teníamos, que no es otra que ayudar a la gente a poder controlar el consumo doméstico de agua y evitar el gasto innecesario del mismo.

ANEXO I

- Caudalímetro:
https://tienda.bricogeek.com/otros/936-sensor-de-flujo-yf-s201.html?search_query=caudalimetro&results=1

- Batería: https://tienda.bricogeek.com/baterias-lipo/1313-bateria-lipo-300mah-37v.html?search_query=BATERIA+&results=136

- ESP32:
https://tienda.bricogeek.com/wifi/892-sparkfun-esp32-thing.html

- Cargador de batería:
https://tienda.bricogeek.com/cargadores-de-bateria/1232-cargador-lipo-usb-tp4056.html?search_query=cargador+de+bateria+lipo&results=65

- Pantalla:
https://es.aliexpress.com/item/32716147339.html?spm=a2g0o.productlist.0.0.11e43eb3nHspTE&algo_pvid=3cd06324-148f-4ac9-a30a-c12628e4ebe2&algo_expid=3cd06324-148f-4ac9-a30a-c12628e4ebe2-7&btsid=0be376451588239973117 3713efe2d&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

- Sensor de temperatura:
https://es.aliexpress.com/item/4000109314442.html?spm=a2g0o.productlist.0.0.46922c48Zhaawx&algo_pvid=9998d973-291c-414a-bb57-2a00814bb339&algo_expid=9998d973-291c-414a-bb57-2a00814bb339-0&btsid=0ab50a5715882405086155645e6a4f&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

- Protoboard:
https://tienda.bricogeek.com/varios/1219-placa-de-prototipo-8x5cm-400-puntos.html

- Led:
https://es.aliexpress.com/wholesale?SearchText=led+5mm&d=y&origin=y&minPrice=&maxPrice=&isBigSale=n&isFreeShip=y&isNew=n&isFavorite=n&shipFromCountry=&shipCompanies=&CatId=0&g=y&SortType=price_asc&initiative_id=SB_20190507220308&filterCat=204006188,204002359,204005275&needQuery=n&groupsort=1

- Cable cargador:
https://tienda.bricogeek.com/cables/451-cable-usb-micro-b.html

- Interruptor:
https://es.aliexpress.com/item/33031693831.html?spm=a2g0o.productlist.0.0.544868e531QLx7&algo_pvid=743c84ec-e4ca-4509-b415-5111ff3d94ee&algo_expid=743c84ec-e4ca-4509-b415-5111ff3d94ee-0&btsid=0ab6d5951588693086735693eb1f5&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

ANEXO II

Código del maestro:

```
//ESTE ES EL CAUDALIMETRO QUE VA A ENVIAR LA INFO ALA APP
#include <BluetoothSerial.h>

#include <ETH.h>
#include <WiFi.h>
#include <WiFiAP.h>
#include <WiFiClient.h>
#include <WiFiGeneric.h>
#include <WiFiMulti.h>
#include <WiFiScan.h>
#include <WiFiServer.h>
#include <WiFiSTA.h>
#include <WiFiType.h>
#include <WiFiUdp.h>

#include <esp_now.h>

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//Caudalímetro
const int sensorPin = 2;
const int measureInterval = 2500;
volatile int pulseConter;
// YF-S201
const float factorK = 7.5;
float volume = 0;
long t0 = 0;
```

```
//VALORES DONDE SE ALMACENARÁN LOS VALORES DEL CAUDAL Y LA TEMPERATURA
float flow_Lmin;
float temp;

//Sensor de temperatura
const int sensorTempPin= 36;
float celsius = 0;
//LEDs
const int ledPinRojo = 25;        //Batería
const int ledPinAzul = 26;        //Consumo

//Batería
const int batPin = 39;
float bateriaBaja = 20;
float porcentajeBateria = 0;

//Pantalla
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// ESTRUCTURA DEL MENSAJE QUE SE VA A ENVIAR AL MAESTRO, DEBE DE SER IGUAL TANTO EN ELESCLAVO COMO EN EL MAESTRO
typedef struct struct_message {
  float temperatura;
  float caudal;
} struct_message;

//CREAMOS LA ESTRUCTURA A LA QUE LLAMAMOS MIS DATOS
struct_message misDatos;

//CREAMOS EL OBJETO PARA EL BLUETOOTH
BluetoothSerial ESP_BT;

// FUNCION PARA VER POR EL MONITOR SI ESTAMOS RECIBIENDO CORRECTAMENTE LOS DATOS
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&misDatos, incomingData, sizeof(misDatos));
  Serial.print("Bytes recibidos: ");
  Serial.println(len);
  Serial.print("Temperatura: ");
  Serial.println(misDatos.temperatura);
  Serial.print("Humedad: ");
  Serial.println(misDatos.caudal);
```

```
  Serial.println();
//MÉTODO QUE SE PODRÍA QUITAR
}

void setup() {

  Serial.begin(115200);

  //LE ASIGNAMOS UN NOMBRE A NUESTRO BLUETOOTH
  ESP_BT.begin("Water_Balance");

  //ASIGNACIÓN DE ENTRADAS Y SALIDAS DE NUESTRO PROYECTO
  pinMode(sensorPin, INPUT);
  pinMode(sensorTempPin, INPUT);
  pinMode(ledPinRojo, OUTPUT);
  pinMode(ledPinAzul, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(sensorPin), ISRCountPulse, RISING);
  t0 = millis();

  //CONFIGURACIÓN DEL DISPLAY QUEVAMOS A UTILIZAR
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 asignación fallida"));
    for(;;);
  }
  display.display();
  delay(1000);
  display.clearDisplay();

   //HACEMOS QUE EL DISPOSITIVO SEA UNA "ESTACION WI-FI"
  WiFi.mode(WIFI_STA);

  //INICIALIZAMOS EL PROTOCOLO ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error inicializando el ESP-NOW");
    return;
  }

  //UNA VEZ QUE EL PROTOCOLO SE HA INICIALIZADO, VEMOS ELESTADO DEL
PAQUETE TRANSMITIDO
  esp_now_register_recv_cb(OnDataRecv);


}

void loop() {

  lecturas();
```

```
  representaDatos();

  //ENVIAMOS LOS DATOS A NUESTRA APLICACIÓN
  ESP_BT.print(flow_Lmin);
  ESP_BT.print("|");
  ESP_BT.print(temp);
  ESP_BT.print("|");
  ESP_BT.print(misDatos.caudal);
  ESP_BT.print("|");
  ESP_BT.print(misDatos.temperatura);

  //LED CONSUMO
  if(volume >= 45) {                    //CONSUMO ALTO (DUCHA)
    digitalWrite(ledPinAzul,HIGH);
  }

  //LED BATERÍA
  if (porcentajeBateria <= bateriaBaja)  {
    digitalWrite(ledPinRojo, HIGH);
  }


  delay(2000);

}

//CONTADOR ASOCIADO A LA "RUTINA DE ATENCIÓN A LAS INTERRUPCIONES"
void ISRCountPulse(){
   pulseConter++;
}

//MÉTODO PARA OBTENER LA FRECUENCIA
float GetFrequency(){
   pulseConter = 0;

   interrupts();
   delay(measureInterval);
   noInterrupts();

   return (float)pulseConter * 1000 / measureInterval;
}

 //MÉTODO ENCARGADO DE HALLAR EL VOLUMEN DE AGUA QUE PASA POR EL CUADALÍMETRO CADA SEGUNDO
void SumVolume(float dV){
   volume += dV / 60 * (millis() - t0) / 1000.0;
   t0 = millis();
}
```

```
//MÉTODO PARA OBTENER LA TEMPERATURA
float GetTemperature(){
  int value = analogRead(sensorTempPin);
  float millivolts = (value / 4095.0) * 5000;
  return (float)millivolts / 10;
}

//MÉTODO PARA OBTENER EL PORCENTAJE DE LA BATERIA
float GetBattery(){
  int valorBateria = analogRead(batPin);
  float voltajeBateria = (valorBateria / 4095);
  return (float)voltajeBateria * 100;
}

void lecturas(){
  //OBTENEMOS LA FRECUENCIA
  float frequency = GetFrequency();

  //CALCULAMOS EL CAUDAL EN L/S
  flow_Lmin = frequency / factorK;
  SumVolume(flow_Lmin);

  //CALCULAMOS LA TEMPERATURA Y EL PORCENTAJE DE BATERIA
  temp = GetTemperature();
  porcentajeBateria = GetBattery();

}

void representaDatos(){
  //METODO ENCARGADO DE REPRESENTAR POR PANTALLA LOS VALORES DEL
SENSOR DE TEMPERATURA Y DEL CAUDAL
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.print(temp);
  display.setTextSize(1);  display.print((char)167);
  display.setTextSize(2);  display.println(F("C"));

  display.print(flow_Lmin);
  display.print(F(" "));
  display.println(F("L/s"));

  display.setCursor(96,0);
  display.setTextSize(1);
  display.print(porcentajeBateria); display.print((char)37);
```

```
  display.display();
}
```

Código del esclavo:

```
//ESTE ES EL CAUDALIMETRO QUE VA A ENVIAR LA INFO
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#include <ETH.h>
#include <WiFi.h>
#include <WiFiAP.h>
#include <WiFiClient.h>
#include <WiFiGeneric.h>
#include <WiFiMulti.h>
#include <WiFiScan.h>
#include <WiFiServer.h>
#include <WiFiSTA.h>
#include <WiFiType.h>
#include <WiFiUdp.h>

#include <esp_now.h>

//Caudalímetro
const int sensorPin = 2;
const int measureInterval = 2500;
volatile int pulseConter;
// YF-S201
const float factorK = 7.5;
float volume = 0;
long t0 = 0;

//Sensor de temperatura
const int sensorTempPin= 36;
float celsius = 0;
//LEDs
const int ledPinRojo = 25;        //Batería
const int ledPinAzul = 26;        //Consumo

//Batería
const int batPin = 39;
float bateriaBaja = 20;
float porcentajeBateria = 0;

//Pantalla
#define SCREEN_WIDTH 128 // ANCHO DEL OLED, EN PIXELS
```

```
#define SCREEN_HEIGHT 32 // ALTO DEL OLED, EN PIXELS

#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


//MAC ADDRESS DEL MAESTRO
uint8_t broadcastAddress[] = {0xA4, 0xCF, 0x12, 0x9A, 0x02, 0x50};

//VALORES DONDE SE ALMACENARÁN LOS VALORES DEL CAUDAL Y LA TEMPERATURA
float flow_Lmin;
float temp;

//Variable que nos dice si el envio de datos es correcto
String success;

// ESTRUCTURA DEL MENSAJE QUE SE VA A ENVIAR AL MAESTRO, DEBE DE SER IGUAL TANTO EN ELESCLAVO COMO EN EL MAESTRO
typedef struct struct_message {
  float temperatura;
  float caudal;
} struct_message;

//CREAMOS LA ESTRUCTURA A LA QUE LLAMAMOS MIS DATOS
struct_message misDatos;


// HACEMOS UNA LLAMADA CUANDO LOS DATOS SE ENVIEN CORRECTAMENTE
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio fallido");
  if (status ==0){
    success = "Envio exitoso";
  }
  else{
    success = "Envio fallido";
  }
}



void setup() {

  Serial.begin(115200);

  //ASIGNACIÓN DE ENTRADAS Y SALIDAS DE NUESTRO PROYECTO
  pinMode(sensorPin, INPUT);
```

```
  pinMode(sensorTempPin, INPUT);
  pinMode(ledPinRojo, OUTPUT);
  pinMode(ledPinAzul, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(sensorPin), ISRCountPulse, RISING);
  t0 = millis();

   //CONFIGURACIÓN DEL DISPLAY QUEVAMOS A UTILIZAR
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 asignación fallida"));
    for(;;);
  }
  display.display();
  delay(1000);
  display.clearDisplay();

  //HACEMOS QUE EL DISPOSITIVO SEA UNA "ESTACION WI-FI"
  WiFi.mode(WIFI_STA);


  // INICIALIZAMOS EL PROTOCOLO ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error inicializando ESP-NOW");
    return;
  }

  //UNA VEZ QUE EL PROTOCOLO SE HA INICIALIZADO, VEMOS ELESTADO DEL
PAQUETE TRANSMITIDO
  esp_now_register_send_cb(OnDataSent);

  // REGISTRAMOS EL MAESTRO
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // AÑADIMOS AL MAESTRO
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Error al añadir al maestro");
    return;
  }

}

void loop() {
  lecturas();
  representaDatos();
  misDatos.temperatura = temp;
```

```
  misDatos.caudal = flow_Lmin;
  //ENVIAMOS EL MENSAJE VIA ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &misDatos,
sizeof(misDatos));

  if (result == ESP_OK) {
    Serial.println("Envio de datos exitoso");
  }
  else {
    Serial.println("Error enviando los datos");
  }

  //LED CONSUMO
  if(volume >= 45) {                    //CONSUMO ALTO (DUCHA)
    digitalWrite(ledPinAzul,HIGH);
  }

  //LED BATERÍA
  if (porcentajeBateria <= bateriaBaja)  {
    digitalWrite(ledPinRojo, HIGH);
  }


  delay(2000);

}

//CONTADOR ASOCIADO A LA "RUTINA DE ATENCIÓN A LAS INTERRUPCIONES"
void ISRCountPulse(){
   pulseConter++;
}

//MÉTODO PARA OBTENER LA FRECUENCIA
float GetFrequency(){
   pulseConter = 0;

   interrupts();
   delay(measureInterval);
   noInterrupts();

   return (float)pulseConter * 1000 / measureInterval;
}

//MÉTODO ENCARGADO DE HALLAR EL VOLUMEN DE AGUA QUE PASA POR EL
CUADALÍMETRO CADA SEGUNDO
void SumVolume(float dV){
   volume += dV / 60 * (millis() - t0) / 1000.0;
   t0 = millis();
```

```
}

//MÉTODO PARA OBTENER LA TEMPERATURA
float GetTemperature(){
  int value = analogRead(sensorTempPin);
  float millivolts = (value / 4095.0) * 5000;
  return (float)millivolts / 10;
}

//MÉTODO PARA OBTENER EL PORCENTAJE DE LA BATERIA
float GetBattery(){
  int valorBateria = analogRead(batPin);
  float voltajeBateria = (valorBateria / 4095);
  return (float)voltajeBateria * 100;
}

void lecturas(){
  //OBTENEMOS LA FRECUENCIA
  float frequency = GetFrequency();

   ///OBTENEMOS LA FRECUENCIA
  flow_Lmin = frequency / factorK;
  SumVolume(flow_Lmin);

  //CALCULAMOS LA TEMPERATURA Y EL PORCENTAJE DE BATERIA
  temp = GetTemperature();
  porcentajeBateria = GetBattery();

}

void representaDatos(){
  //METODO ENCARGADO DE REPRESENTAR POR PANTALLA LOS VALORES DEL
SENSOR DE TEMPERATURA Y DEL CAUDAL
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.print(temp);
  display.setTextSize(1);  display.print((char)167);
  display.setTextSize(2);  display.println(F("C"));

  display.print(flow_Lmin);
  display.print(F(" "));
  display.println(F("L/s"));

  display.setCursor(96,0);
  display.setTextSize(1);
```

```
  display.print(porcentajeBateria); display.print((char)37);

  display.display();
}
```