15-11-2023

Trabajo Práctico

Colillero

Docente:

- Ing. Valverde, Franco

Alumnos:

- Baldo, Facundo
- Bono, Valentino
- Giorda, Juan Cruz

INDÍCE

1.Introducción …… 2

1.1 Propósito ……...2

1.2 Alcance ..……... 2

1.3 Visión general del documento ……... 2

2.Descripción general ….…… 2

2.1 Perspectiva del sistema...….……...2

2.2 Funciones del sistema.....…….... 3

2.3 Características de los usuarios …...……... 3

3.Requisitos específicos …… 3

3.1 Funciones ……...3

3.2 Requerimientos no funcionales ..……... 4

3.3 Arquitectura ……...……... 4

3.4 Apéndices ……...……... 5

4.Arduino ……...…… 6

5.Links ....……...…… 12

1 introducción

1.1 Propósito

El propósito del presente documento es establecer el alcance y la funcionalidad requerida para la integración del sistema de colilleros en la facultad. Esta iniciativa tiene como objetivo principal mejorar la gestión de residuos de cigarrillos a través de la implementación de sensores en los colilleros. Estos sensores permitirán detectar el nivel de llenado de los colilleros, contar las colillas que ingresan, y transmitir esa información a una base de datos en Django, vinculada con una página web o móvil donde se muestran los colilleros en Vue. La finalidad es proporcionar a los encargados de mantenimiento notificaciones en tiempo real cuando un colillero esté próximo a llenarse, facilitando así la planificación eficiente de actividades de vaciado.

1.2 Alcance

La integración del sistema de colilleros abarcará la instalación de sensores en los colilleros distribuidos en la facultad, la transmisión de datos al servidor central, y la implementación de una plataforma web o móvil para notificar a los encargados de mantenimiento.

1.3 Visión general del documento

En la Sección 2 se presentará una descripción general de la integración del sistema de colilleros, incluyendo la perspectiva del producto, funciones, características de los usuarios, restricciones generales, suposiciones y dependencias relevantes. Posteriormente, se detallarán los requisitos específicos del sistema en la Sección 3, que incluirán funciones, requisitos no funcionales, y la arquitectura propuesta para la integración.

2 Descripción general

2.1 Perspectiva del sistema

El sistema de colilleros desarrollará como una aplicación autónoma que no depende de otros sistemas externos para su funcionamiento.

2.2 Funciones del sistema

El sistema de colilleros se centrará en las siguientes funciones claves:

-Detección del nivel de llenado de los colilleros mediante sensores.

-Transmisión eficiente de datos al servidor central.

-Notificación automática a través de una plataforma web o móvil cuando un colillero esté próximo a llenarse.

-Registro y seguimiento del historial de llenado

2.3 Características de los usuarios

Los usuarios que interactuarán con el sistema de colilleros son:

Encargados de mantenimiento: Accederán a la plataforma web o móvil para recibir notificaciones y realizar un seguimiento del estado de los colilleros.

Administradores: Contarán con funciones avanzadas para gestionar usuarios, configurar parámetros del sistema y acceder a informes detallados.

3 Requisitos específicos

3.1 Funciones

RC1 - El sistema de colilleros debe ser capaz de detectar de manera precisa el nivel de llenado de los colilleros.

RC2 - El sistema debe transmitir los datos del nivel de llenado al servidor central de manera eficiente y segura.

RC3 - La plataforma web o móvil debe notificar automáticamente a los encargados de mantenimiento cuando un colillero esté próximo a llenarse.

RC4 - El sistema debe permitir el registro y seguimiento del historial de llenado de cada colillero.

RC4 - El sistema debe poder ver la ubicación del correspondiente colliero que está próximo a llenarse.

3.2 Requerimientos no funcionales

Rendimiento: El sistema de colilleros debe tener una respuesta rápida para proporcionar notificaciones en tiempo real.

Tolerancia a fallos: El sistema debe ser capaz de manejar interrupciones momentáneas en la comunicación sin pérdida de datos críticos.

Seguridad: Se deben implementar medidas de seguridad para proteger la integridad de los datos transmitidos y almacenados en la plataforma.

3.3 Arquitectura

Utilizaremos el modelo cliente-servidor, en este, el sistema se divide en dos partes principales: el cliente y el servidor. A continuación, presentamos una descripción del funcionamiento para el sistema:

Servidor:

En el sistema de colillero, el servidor sería la parte central encargada de gestionar y procesar las solicitudes relacionadas con la recolección y gestión de colillas de cigarrillos. Este componente manejaría la lógica de negocios, incluyendo la gestión de ubicaciones de colilleros, el seguimiento del nivel de llenado, la coordinación de la recolección y el almacenamiento de datos asociados.

El servidor proporcionaría una interfaz de programación de aplicaciones (API) que permitiría a los clientes, que podrían ser aplicaciones móviles o web, interactuar con el sistema. A través de esta API, los clientes pueden enviar solicitudes al servidor para informar sobre la ubicación de colilleros

llenos, solicitar servicios de recolección, o consultar estadísticas sobre la cantidad de colillas recogidas.

Cliente:

El cliente sería una aplicación móvil o web que permite a los usuarios consultar la ubicación de colilleros llenos, solicitar servicios de recolección y recibir información sobre la gestión de colillas en su área. Los usuarios interactúan directamente con esta interfaz para contribuir al proceso de recolección y mantener limpios los espacios públicos.

Funcionamiento general:

Los sensores detectarán el nivel de llenado de los colilleros cuando alcancen un umbral específico y contarán las colillas entrantes.

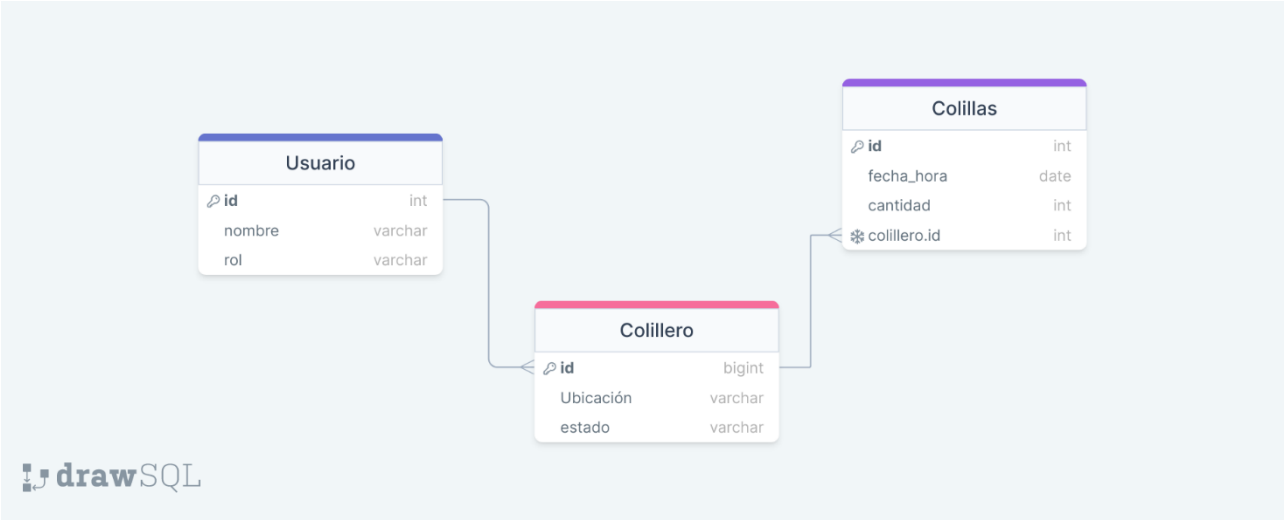
El usuario utiliza la aplicación para consultar al servidor sobre la ubicación de un colillero lleno.

Una vez que se completa la recolección, el servidor actualiza la base de datos, confirma la recolección al usuario y puede proporcionar estadísticas actualizadas sobre la cantidad total de colillas recogidas en la zona.

Este sistema ayudaría a gestionar de manera efectiva la recolección de colillas, optimizando los recursos y fomentando la participación de la comunidad en la mejora del entorno.

3.4 Apéndices

Diagrama de clases



4 Arduino

En el proyecto utilizamos una placa 'ESP-WROOM-32' la cual cuenta con Wi-Fi para poder enviar los datos generados por los sensores a la base de datos.

Necesitamos dos sensores infrarrojos 'E18-D80NK'.

- Sensor 1: Es para el conteo de las colillas, el cual está programado para que cuando pase una colilla por el mismo a la distancia definida envié una señal a la base de datos de que ha ingresado una nueva colilla.

- Sensor 2: Es para saber cuándo se ha llegado al almacenamiento completo del colillero, aquí también se enviará una señal a la base de datos para cambiar su estado a lleno, lo que permitirá avisar al servicio de limpieza de que es hora de limpiar el colillero. Una vez sea vaciado el sensor lo detectara y mandara nuevamente una señal para avisar que ya fue limpiado.

Estos sensores necesitan de 5V, un problema que se nos presentó es que está placa cuenta con salida de 3.3V, pero no de 5V (solamente de entrada). Pero al conectar la placa a la computadora también funciona como salida, por eso no es necesario una batería externa o un Arduino Mega o Uno.

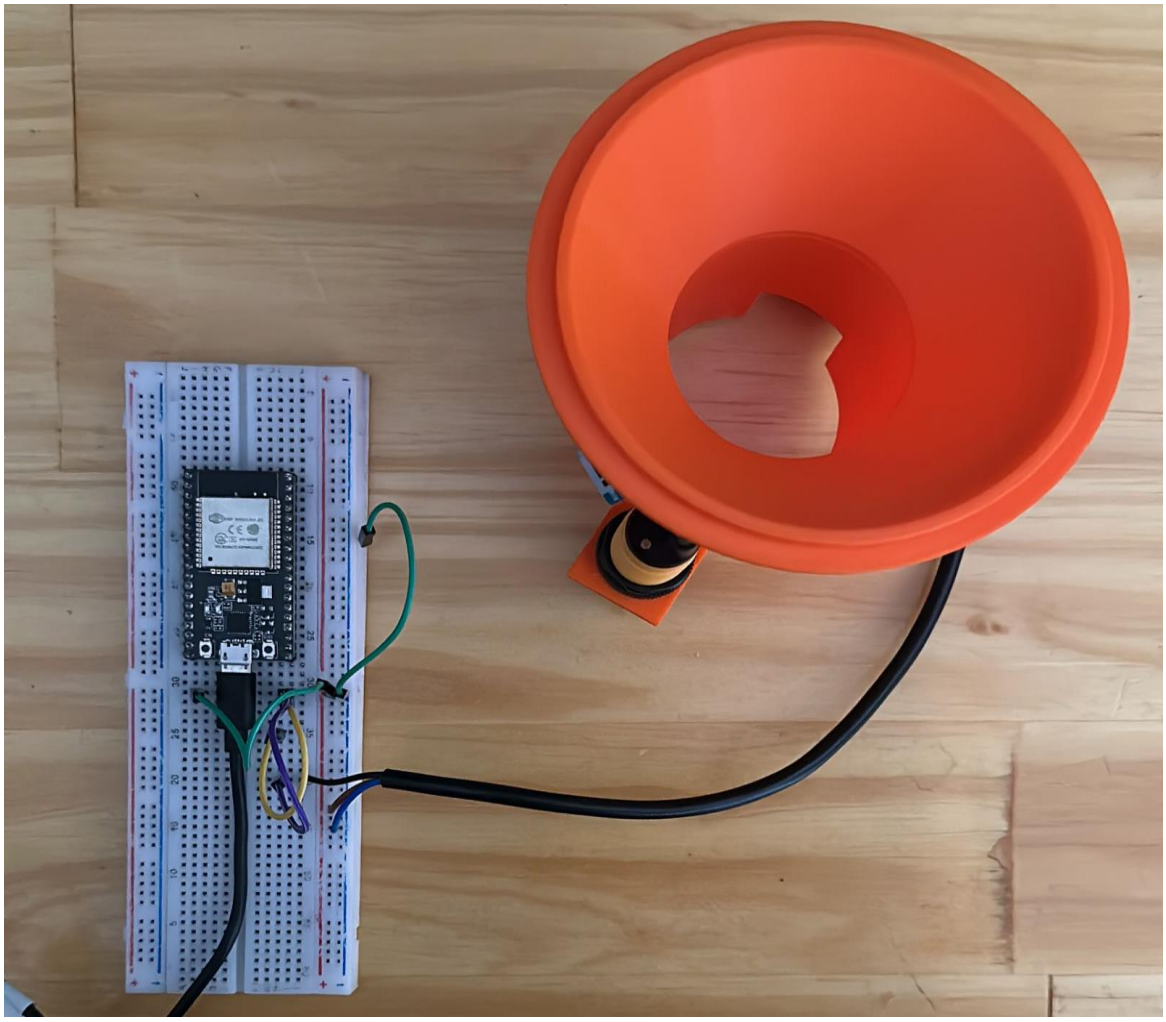
Errores típicos que nos cruzamos a lo largo del proyecto:

- Para el reconocimiento del ESP32 debemos de instalar la 'biblioteca' de esta placa desde el gestor de tarjetas.

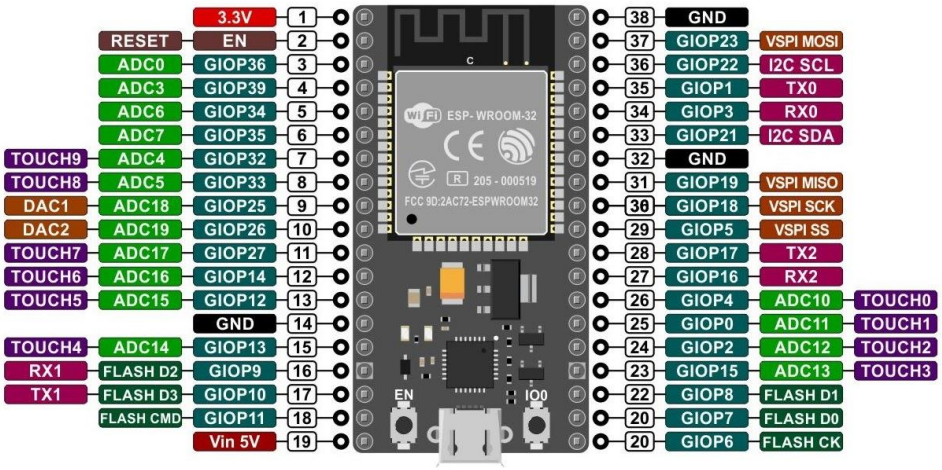
- Archivo JSON en preferencias.
- Velocidad a 115200 baudios.
- Velocidad a 40mhz.
- Nombre Driver: Silicon Labs CP210x USB to UART Bridge.
- Ver que está conectada a una placa ESP-WROOM-DA Module y no a la más típica 'Dev Module'.

Videos que sirven como guía para esta instalación:
Reconocimiento de placa: https://www.youtube.com/watch?v=I-NW9gTfIUY&t=194s
Driver: https://www.youtube.com/watch?v=az-X_1QSuOY&t=193s

(Conexión)



(Pines ESP32)



Código sensor – Contador de colillas y detección nivel de llenado de colillero:

```
#include <WiFi.h>
#include <HTTPClient.h>

//const int SENSOR_PIN = 0;   // Pin para el sensor de distancia  // Ajusta este umbral según sea necesario
const int NIVEL_PIN = 7;     // PROFUNDIDAD
const int CONTADOR_PIN = 6;  // CONTADOR
const char* ssid = "juli";
const char* password = "julis135";
const char* serverName = "http://192.168.241.155:8000/registro_estados/";

static unsigned long previousMillis = 0;

void llamada(char* estadoactual) {
  WiFiClient client;
  HTTPClient http;

  http.begin(client, serverName);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  String httpRequestData = "estado=";
  httpRequestData += estadoactual;
  // Send HTTP POST request
  int httpResponseCode = http.POST(httpRequestData);

  Serial.print("HTTP Response code: ");
  Serial.println(httpResponseCode);
```

```
  // Free resources
  http.end();
}

void connectToWiFi() {
  Serial.println("Conectándose a la red Wi-Fi...");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Intentando conectar a la red Wi-Fi...");
  }

  Serial.println("Conexión exitosa a la red Wi-Fi");
  Serial.print("Dirección IP asignada: ");
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(115200);         // Inicia el monitor serial a 115200 baudios
  pinMode(NIVEL_PIN, INPUT);    // Configura el pin del sensor como entrada
  pinMode(CONTADOR_PIN, INPUT);  // Configura el pin del sensor como entrada
  Serial.println("Prueba del sensor de distancia");
  Serial.println("");
  connectToWiFi();
}
void loop() {
  int valorContador = digitalRead(CONTADOR_PIN);

  if(valorContador){
    // avisar al backend
  }

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= 10000) {
    int valorSensor = digitalRead(NIVEL_PIN);
    if (valorSensor) {
      llamada("lleno");
    } else {
      llamada("vacio");
    }
  }
```

```
 delay(250);
}
```

Impresión

El laboratorio de electrónica nos imprimió un soporte para los dos sensores. En caso de necesitar generar un prototipo se puede utilizar la página de 'Tinkercad' la cual es muy fácil de utilizar y hay miles tutoriales en YouTube. Además, hay un foro donde se pueden utilizar prototipos armados por otras personas para acelerar el proceso de fabricación.

LINKS

Repositorio GitHub https://github.com/JuanCruzGiorda/colillero

-Fronted realizado con Vue
-Backend realizado con Django