MEMORIA

PROYECTO DAD

Sergio Isidoro De Anca Garabito
Alberto Cabello García

Índice:

Introducción……...2
Base de datos: …….3
API Rest:.....4
MQTT: ……5
ESP·32: ……5
HARDWARE: ……5
Resultado final del proyecto:.....7

Introducción:

Al inicio del curso, la idea original de nuestro proyecto, era modificar un coche de juguete para darle ciertas funcionalidades propias de los coches de verdad.

Pensamos en funcionalidades como un sistema de frenos ABS, apertura automática de puertas al detectar que la “llave del coche” está cerca o un sistema que hace que el coche empiece a pitar si está muy cerca de un obstáculo, pero al final descartamos todo esto por ideas más sencillas.

Estas ideas eran añadir un sistema de frenos HAC(Asistencia de Partida en Pendientes), iluminación automática de los “faros” del coche según la luz de alrededor y la regulación automática de la temperatura dentro del coche.

Para el sistema de frenos, teníamos pensado usar un acelerómetro que detecta si el coche está en una rampa, si es el caso y el coche está parado, se activarán “los frenos” automáticamente siendo los supuestos frenos un servomotor que bloquearía la rueda.

Para regular la luz de los faros automáticamente, usamos un sensor de luz para detectar el nivel de luminosidad y en base a ese nivel iluminar unos leds que simulan los faros con más o menos potencia.

Por último, para simular el aire acondicionado automático usamos un detector de temperatura y un pequeño ventilador que se activaría dependiendo de si pasa o no de cierto umbral.

De estas tres ideas, al final descartamos el sistema de frenos por su mayor complejidad y nos quedamos únicamente con la luz y la temperatura.

Base de datos:

En el proyecto, tendremos que almacenar los datos que recogen nuestros sensores y los valores que tendremos que dar a los actuadores para que funciones correctamente, y para esto necesitamos crear una base de datos.

Usaremos las aplicaciones “MariaDB” y “HeidiSQL” para crear y gestionar nuestra BBDD.

Como explicamos en la introducción, al final usaremos dos sensores, para luz/temperatura y dos actuadores para luz(led)/temperaturas(ventilador) y por tanto vamos a crear una tabla para cada uno de estos, las tablas son las siguientes:

SensorTemperatura:

idtemp	INTEGER	Id del sensor que lee la temperatura
temperatura	DOUBLE	El valor de temperatura leído
timestampt	BIGINT	Momento en que se ha leído dicho valor
idP	INTEGER	Id de la placa donde está el sensor
idG	INTEGER	Grupo al que pertenece la placa

SensorLuz

idl	INTEGER	Id del sensor que mide la luz
nivel_luz	DOUBLE	El nivel de luz que ha leído
timestampl	BIGINT	Momento en que se ha leído dicho valor
idP	INTEGER	Id de la placa donde está el sensor
idG	INTEGER	Grupo al que pertenece la placa

ActuadorLed

idla	INTEGER	Id del actuador led
nivel_luz	DOUBLE	Nivel de luz con el que brillar
timestample	BIGINT	Momento en que el actuador se actualiza
idP	INTEGER	Id de la placa donde está el sensor
idG	INTEGER	Grupo al que pertenece la placa

ActuadorFan

idfa	INTEGER	Id del ventilador actuador
onoff	BOOLEAN	Indica si el ventilador debe estar apagado o encendido
timestampf	BIGINT	Momento en que el actuador se actualiza
idP	INTEGER	Id de la placa donde está el actuador
idG	INTEGER	Grupo al que pertenece la placa

API Rest:

Nuestra API REST sirve para conectar nuestro servidor con la BBDD descrita en el apartado anterior. En la API hemos creado funciones GET y POST por cada tabla para pasar datos entre el servidor y la BBDD.

Las funciones para el sensor temperatura son las siguientes:

getALLST → Obtiene todos los registros del sensor temperatura
getBySensorT → Obtiene los registros de un sensor en concreto
getLastBySensorT → Obtiene el último valor leído por un sensor
addSensorT → Introduce en la BBDD un nuevo valor de temperatura

Las del actuador ventilador son:
getALLF → Obtiene todos los valores de los actuadores
getByActF → Obtiene los valores de ha tenido un actuador en concreto
getLastByActF → Obtiene el último valor que ha tenido el actuador
addAF → Introduce en la BBDD un nuevo valor para el actuador

Las URLs del servidor son las siguientes:

"/api/temperaturas" → funciones “getAllST” y “addSensorT”
"/api/temperaturas/:idtemp" → funciones “getBySensorT” y “getLastBySensorT”

"/api/actfan" → funciones “getAllAF” y “addAF”
"/api/actfan/:idfa" → funciones “getByActF” y “getLastByActF”

MQTT:

Para hacer la conexión MQTT y poder pasar los datos entre cliente y servidor hemos usado el broker mosquitto para ejecutar el servidor y hemos utilizado el SoftWare MqttExplorer para comprobar que funcionaba correctamente y que los datos se transferían de manera adecuada.

El cliente (la placa) está suscrito al tópico “twmp”

FIRMWARE ESP 32:

Para implementar el firmware de este proyecto hemos usado el sistema de desarrollo Visual Studio y hemos usado el lenguaje c++

En dicho archivo hemos conectado la placa a una red wifi y hemos hecho un subscribe de la placa al servidor mqtt con el tópico “twmp”.

En el loop del firmware, se recogen los datos del sensor, se serializan con una función específica y se envían al servidor para que los añada a la base de datos con una petición post.

Cada vez que envía un dato, hay una función de callback de mqtt para recibir los datos de un “publish realizado en el servidor”, el callback se encargará de encender o apagar los actuadores.

HARDWARE:

Los componentes HARDWARE que hemos utilizado son los siguientes:

Placa → ESP32

Sensor Temperatura → DHT11

Sensor Luminosidad → BH170

Actuador 1 → Led normal

Actuador 2 → Ventilador 5V

Para poder usar el ventilador nos hemos encontrado con que necesitaba 12mA de intensidad como mínimo y la placa ESP32 no lo soporta por lo tanto hemos usado un transistor NPN 2N2222 para suministrar la potencia que requería el ventilador.

CONEXIONES:
Ventilador:

- Conecta el positivo del ventilador al VCC de 5V. (Cable rojo/naranja)
- Conecta el negativo del ventilador al colector del transistor (por ejemplo, un transistor NPN como el 2N2222). (Cable negro/verde)

Transistor:

- Conecta el emisor del transistor a GND.(Cable azul)
- Conecta una resistencia de 1kΩ entre la base del transistor y el pin GPIO 14 del ESP32.(Cable verde)
- Conecta el colector del transistor al ventilador.(Cable verde)

Led:

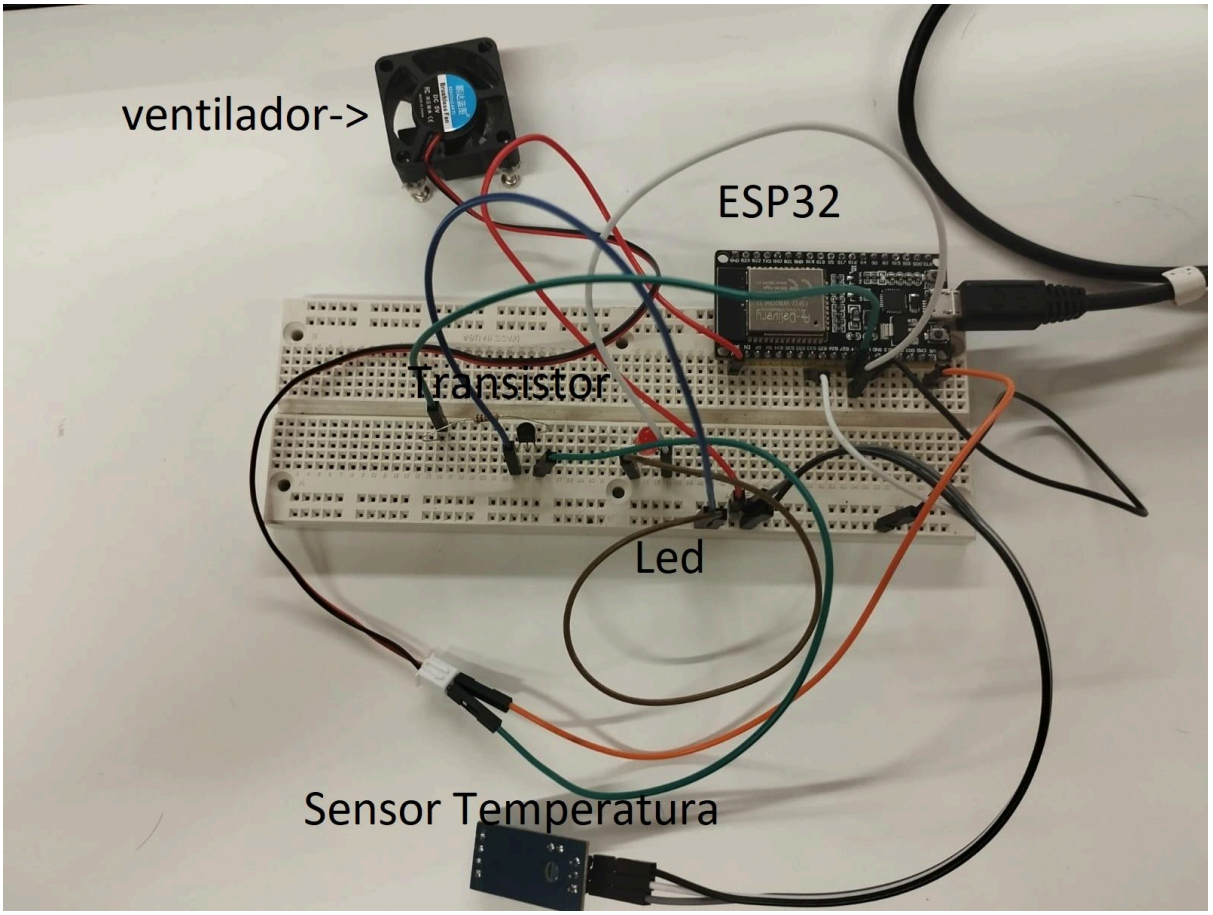
- Conecta el positivo al pin GPIO 14 del ESP32.(Cable blanco)
- Conecta el negativo a tierra (GND).(Cable marrón)

Sensor Temperatura:

- Conecta el negativo con GND.(Cable negro)
- Conecta el positivo con el pin de 3.3 voltios del ESP32.(Cable gris)
- Conecta el pin de transmisión de datos con el pin GPIO 33

ESP32:
- pin 3.3V : positivo sensor temperatura
- pin 5V : positivo ventilador
- pin GPIO 33:data sensor temperatura
- pin GPIO 14 :base transistor, positivo led
- pin GND: negativo sensor temperatura, negativo led,emisor transistor

Hardware completo:



Resultado final del proyecto:

Hemos tenido problemas con el sensor de luz que pensábamos poner ya que el sensor nos requería usar el bus I2C de la placa y después de intentarlo con varias librerías e incluso a mano y no hemos conseguido dar con ninguna que nos habilitase el bus I2C para que funcionara el sensor, por eso decidimos continuar sólo con el sensor de temperatura y como actuadores el ventilador y el led.

Quitando eso hemos conseguido todas las funcionalidades del proyecto que teníamos previstas, ya que hemos conseguido que cada 2 segundos(delay añadido por nosotros) el sensor mande datos, se guarden en la base de datos y además los actuadores se enciendan cuando la temperatura registrada es mayor a 30 grados.

Los videos demostrativos del funcionamiento final del proyecto están subidos a github al igual que todas las entregas del proyecto