Intelligent O internet of Things
2º Curso de Master de
Ingeniería en Informática
Área de Ciencias de la Computación e
Inteligencia Artificial
Departamento de
Tecnologías de la Información

PRÁCTICA 2 V2.0

Imeter: un cargador de Vehículos eléctricos

Objetivos

La segunda parte de la práctica tiene como objetivo diseñar un sistema con multiples productores y consumidores de información que representen el esquema básico de un sistema de IOT distribuido y extensible.

La documentación debe tener en cuenta la maquina de estados implícita y representar los estados del sistema para que ninguna situación se quede en un estado del que no sea posible retornar o no haya respuestas a un evento que necesite una acción.

El servidor MQTT y la configuración de los mensajes debe ser documentada de manera formal para que alguien que tuviese que actualizar el sistema pudiese interpretar y modificar los mismos.

Los eventos y módulos presentados son una propuesta, los nombres, cantidades y orden de proceso pueden ser modificados si no resultan en una merma de funcionalidad y son justificados adecuadamente.

Enunciado de la práctica

El objetivo general del proyecto será la de automatizar la carga de un coche eléctrico en base a los precios de la electricidad por horas y la potencia contratada.

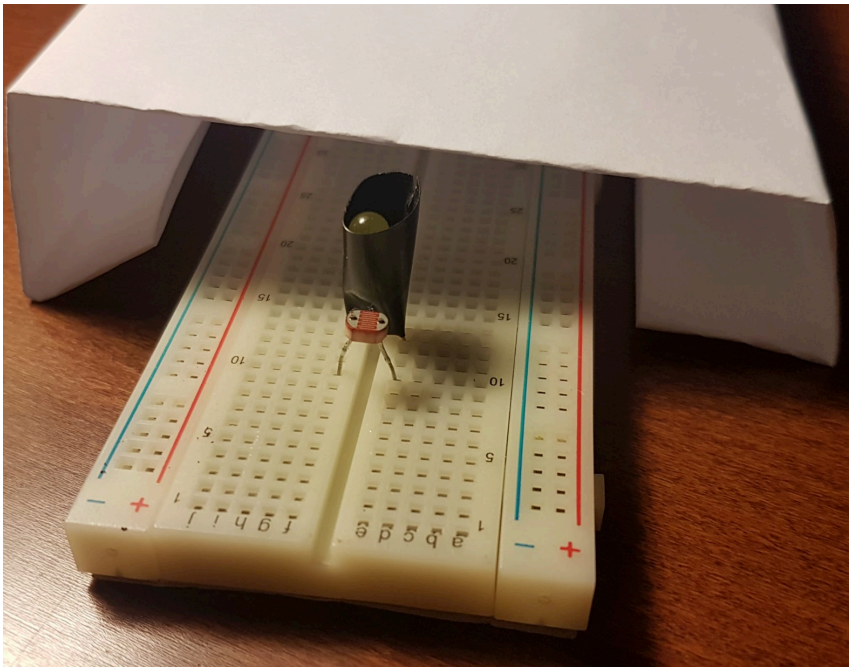
Fase I

En una primera fase desarrollaremos un cargador inteligente que detecta cuando el coche está aparcado mediante el sensor de luz y a partir de una hora planifica las horas de carga según los precios. El cargador comunica el estado de carga, hora de inicio , fin , energía consumida y precio de la energía al principio (estimación) y final de la carga (suma real).

Detección de presencia del vehículo

El sensor de luz debe diferenciar las situaciones de coche aparcado, tanto de día como de noche , por lo que se habrá una pequeña iluminación que refleja una luz determinada cuando el coche está aparcado y se mantiene durante un periodo suficiente de tiempo (ventana de medidas). A plena luz o a completa oscuridad detectará la plaza vacía.

Para simular la plaza, se deberá situar el sensor de luz mirando hacia arriba y con un led junto a él encendido con un pequeño cono de papel. Encima de este sensor a una distancia determinada se tapará con una pequeña caja hecha con hoja de papel que impida luz lateral (hacer una pequeña caja). Esta caja simulará el vehículo.



Ejemplo de disposición de sensor de luz y led.

El sensor deberá estar calibrado (puede ser un programa previo o ser parte de una secuencia de arranque) para establecer los limites entre los cuales se detecta la presencia del coche y diferenciarlo de la oscuridad de la noche o pleno día.

Puesta en hora

El dispositivo llamará al servicio de NTP para obtener la hora en tiempo real . Hay múltiples ejemplos de código para sincronizarse con servidores NTP desde una ESP8266 o Arduino con ethernet.

Precios PVPC Tarifa 2.0DHA

Para extraer los precios de la tarifa nocturna en dos tramos (2.0DHA) exite un api en https://www.esios.ree.es/es/pagina/api , donde hay que solicitar un token por correo . Para esta práctica podeis usar:

TOKEN "b88ea74d42178303a2a76c2a167e7d17007fc1131513cacb9e3c935dfa1c2e2e"

Existen varias implementaciones en Python (por ejemplo : https://bit.ly/3qYJ8dp, que ya incluye en token) , y alguna en arduino que tiene incluidas llamadas a NTP y código

de lcd y otros sensores que habría que limpiar para su uso (https://github.com/rgrastur/ArduinoEsios) .

Esta parte es opcional en la fase 1, se puede poner un precio a mano.

Descripción del proceso de carga Inteligente

El proceso tendrá en cuenta las siguientes variables/constantes:

- Precio de la luz
- Hora actual
- **P**: Potencia del cargador : estimar 4kw
- **N** : Numero de horas de carga : 4

Si la hora es mayor de las 21:00 y no tenemos los precios actualizados, descargará los precios de la web . Se calcula el periodo de N horas continuas
Se sacarán las N horas más baratas (N es una cte). Para ello se recorre el array de horas desde las 00:00 hasta las 10:00am sumando intervalos de 4 horas, Guarda la Hora de inicio y fin programada (Hinicio, Hfin), calcula la carga y el precio y los muestra por pantalla (serie):

```
Horario de carga y precios:

Inicio: 2:00
Fin: 6:00
Carga programada: 16kwh
Precio:
       2:00  4 cts
       3:00  4 cts
       4:00  3.5 cts
       5:00. 2.5cts
Total   56 cts
```

Cuando llegue la hora de carga sistema activará el cargador si el coche está aparcado y se ha llegado a esa hora enviando un mensaje a pantalla:

```
2:00 comienza carga
```

Si no está el coche aparcado, se muestra un mensaje de error:

```
2:00 Coche no está preparado para carga
```

Si el coche llega dentro del periodo se pondrá a cargar ,sin necesidad de recalcular periodo .

```
2:35 comienza carga
```

Si se va dentro del periodo, se desconecta y muestra un mensaje :

```
2:50 Se detiene la carga , no hay vehículo.
```

Cada cierto tiempo se muestra el estado de carga y tiempo cargando

```
2:45 10 minutos 0,6kwh
2:55 20 minutos 1,2kwh
```

Cuando se llegue a la hora de fin de carga se mostrará el mensaje:

```
Carga Terminada
2:35 Inicio de carga
6:00 Fin de carga

Carga programada: 16kwh
Precio:
       2:00  4 cts
       3:00  4 cts
       4:00  3.5 cts
       5:00. 2.5cts
Total   56 cts
```

Cuando llega/sale el coche independientemente del estado de carga se informa de la hora:

```
   5:00 Sale el coche
```

Fase II

En esta fase se va tener una distribución de las funciones en distintos módulos que se comunican a través de un servidor MQTT. Es necesario tener un ordenador externo (recomendable Raspberry pi) para alojar con un servidor de colas y probablemente varios de los modulos que producen o consumen información.

Esta fase se describirá en detalle en una segunda versión del documento, pero se adelanta la estructura para que el código de la fase 1 ya tenga en cuenta la información a envíar/recibir en fase II:

Modulo Cargador Inteligente:

Se suscribe a :
- Hora de inicio/fin de carga
- Orden de comienzo de carga (inicia aunque no sea la hora si hay coche)
- Orden de fin de carga
- Orden de reinicio de carga (sólo afecta si esta en periodo)
- Pregunta de potencia (para que el servidor calcule las horas de carga)

Produce
- Comienzo de carga
- Fin de carga
- Coche llega/sale

Periódicos:
- Desconectado/cargando (energía acumulada, tiempo de carga acumulado)
- Coche aparcado/no está
- Potencia instantánea (4kw)

Proceso:

El cargador recibe la orden de programar de una hora a otra o inmediatamente..
Si el vehículo está presente , se enciende .Si no lo esta para.

Sensor Coche:

- Produce : % carga batería y tamaño total batería.

Sensor electrodoméstico:

- Produce: Potencia instantánea consumida.

Sensor Usuario:

Representa al usuario y sólo tiene un correo electrónico. Si no se sabe enviar correo electrónico desde proceso, este proceso tendrá una subscripción a "aviso"

Se suscribe:
aviso: "mensaje de aviso"
Solicitud de correo electrónico

Produce:
- Orden: cargar/parar
- CorreoElectrónico "correoelectronico para avisos"

Panel de mandos consola/web/APP:

Se suscribe a :
- Carga del vehículo
- Estado de carga
- Hora de Comienzo
- Hora de Fin
- Energía acumulada/precio
- Precios de la noche (de 22 a 8)

Modulo de control:

Se suscribe a:
- Orden directa de usuario: Parar/Comenzar carga
- Carga actual de vehículo
- Potencia cargador
- Potencia consumida instantánea

produce:
- Orden de carga/parada
- Precios
- Hora comienzo/fin
- Pregunta potencia cargador

Proceso:

El servidor lee los precios a las 21:00, lee la carga del coche y decide cuantas horas necesita según la potencia del cargador entre las 00:00 y las 10:am. Se envía esta información (la utiliza el panel de datos , cargador y logger)
Se envía un mensaje de correo con el plan de carga y el precio total de la carga al usuario.

Para la carga de precios podéis utilizar el siguiente script en Python:
https://bit.ly/3qYJ8dp

Hay otros dispositivos conectados que consumen electricidad y producen un evento con cuanto están consumiendo (potencia instantánea). Si la suma de las potencias es superior a la potencia contratada (asumir 6 kw) emitirá al cargador una orden de parada , cuando vuelva a haber suficiente potencia se envía orden de reanudar la carga. No hay reajuste del numero de horas si esto ocurre, pero se mandará un mail con un aviso al usuario.

Base de datos y logger:
Se suscribe a todos los eventos.

Produce:
Información por eventos

Proceso:
Graba en una base de datos o fichero todos los eventos y su origen con fecha y contenido. Puede usarse por un usuario externo o bien por un módulo para saber si algo ya ha ocurrido en un periodo de tiempo.

Por ejemplo si el cargador ha caído y queremos saber si ya había una hora de carga programada podría preguntar por este evento al logger para recuperar la información. En la práctica supondremos que los sistemas no caen y sólo se usará como debug.

Implementación de módulos

El usuario, sensor electrodoméstico y coche se pueden realizar con una ventana conectad a mqtt suscritos a los topics adecuados y pegando los mensajes predefinidos en el momento adecuado o con un pequeño proceso que corre en segundo plano.

El cargador inteligente se implementa en esp8266 con un cliente MQTT

El modulo de control será un script que carga inicialmente todos los datos de configuración (kwh cargador, email, potencia contratada, etc…) . Esta parte incialmente se puede hacer con ctes y luego incluir la recopilación de esta información con pregunta/respuesta a los modulos.

Una vez cargados los parámetros iniciales corre en bucle descargando a las 21:00 los precios , mandando horas inicio-fin y controlando la potencia consumida para parar o rearrancar .

El panel de control se puede implementar con un proceso que lee la información y la muestra por consola o bien con aplicaciones Android o Web que permiten recibir información mqtt y mostrarla con gráficos de evolución (por ejemplo Linear MQTT dashboard o MQTT Dash). En las aplicaciones web (https://mqtttiles.flespi.io/#/, Thingsboard) es posible que el servidor esté localizado en la web, podéis utilizar entonces algún servidor externo gratuito como HiveMq o similar.

Documentación General

Para cada apartado se realizará una breve explicación del apartado de la práctica , capturas de pantalla y un video cuando haya que mostrar cambios de estado subido a youtube en modo privado (enviar link para su visualización. No es necesario hablar en el video, puede referirse a la explicación en el texto).
Todos los scripts y la documentación en un Word o similar deberá entregarse en un solo .zip o .rar.

Se valorará que se explique el porque de cada decisión refiriéndose a los apuntes. Por ejemplo " se elijen los valores limite por, la medida estadística porque…. , etc…" . Se permite que parte de las explicaciones estén directamente en el código si esto favorece su entendimiento.

No es imprescindible pero se valorará el uso de un dashboard gráfico que muestre la información histórica de la carga y estado actual , uno de los recomendados u otros como nodered, etc….

Fecha de entrega

13 de Mayo a las 16:00
Se expondrá el video y una pequeña presentación con el código y las diferentes explicaciones sobre las implementaciones.