Máster Universitario en Sistemas Electrónicos para Entornos Inteligentes

Sensor IoT basado en Raspberry Pi

Práctica final de la asignatura Microprocesadores Empotrados

Manuel Lorente Almán
12-23-2018

CONTENIDO

1. Introducción

En esta práctica final de la asignatura Microprocesadores Empotrados se ha desarrollado un prototipo de sensor de temperatura IoT, utilizando como sistema principal un ordenador de placa simple basado en la arquitectura ARM como es la placa Raspberry Pi 3.

Este sensor IoT tiene como objetivo realizar mediciones de temperatura cada cierto periodo de tiempo definido y presentarlas en una página web accesible mediante HTTP. Por otra parte, el sistema incluye un led de monitorización de temperatura, el cual varía su intensidad en función de la temperatura de la sala. Finalmente se ha añadido una funcionalidad de envío de mensaje de alarma de manera asíncrona mediante un protocolo no orientado a conexión como es UDP.

Para ello, se ha dividido el proyecto en tres funcionalidades independientes:

- Lectura periódica de la temperatura y presentación en formato HTML.
- Modulación de la intensidad del led en función de la temperatura medida.
- Generación alarma asíncrona.

A lo largo del presente documento se irán introduciendo los diferentes objetivos de la práctica, cómo se ha logrado alcanzarlo, las pruebas de validación de cada funcionalidad y posibles mejoras sobre la implementación propuesta.

2. Objetivos

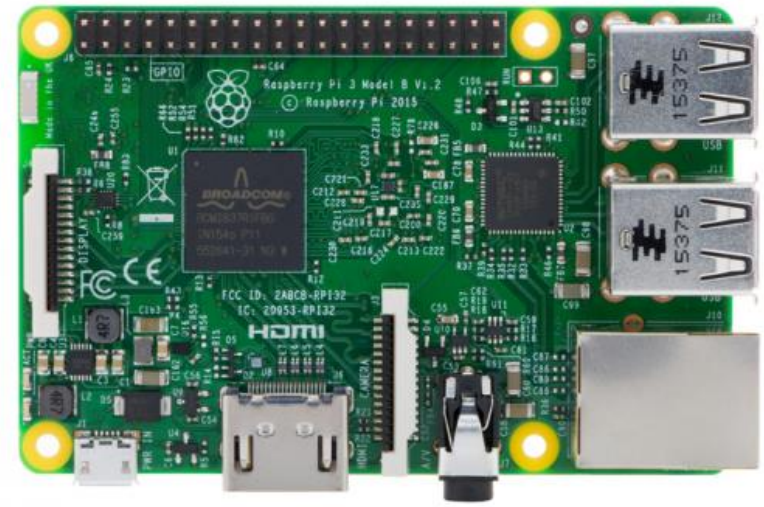
Los objetivos de este sensor IoT son los definidos en el enunciado de la práctica final como resultados de aprendizaje:

- Crear un sensor IoT básico
- Crear un sensor IoT de temperatura
- Leer una entrada digital de forma simultánea
- Activas la intensidad de un led en función de la temperatura.

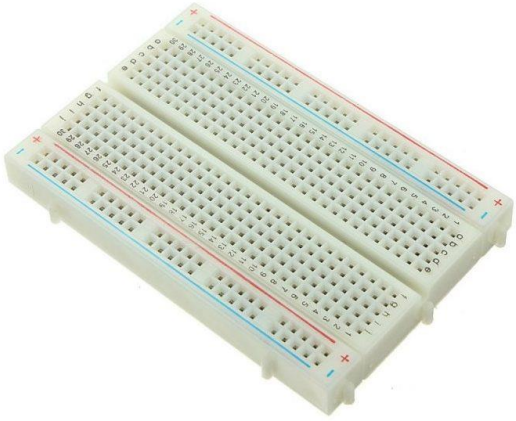
Para cumplir estos objetivos se ha tenido que hacer uso de técnicas de diseño aprendidas en la asignatura como la implementación de hilos, mecanismos de exclusión, métodos de sincronización entre tareas y mecanismos para "dormir" una determinada tarea. Todo esto se explicará en detalle a lo largo de la sección de desarrollo.

Para alcanzar dichos objetivos se han empleado los siguientes recursos:

- Raspberry Pi 3 model B V1.2



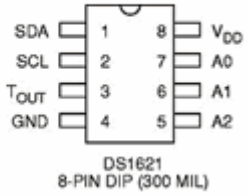
- Protoboard y cables conectores



- Interruptor, led y resistencia de 1k.

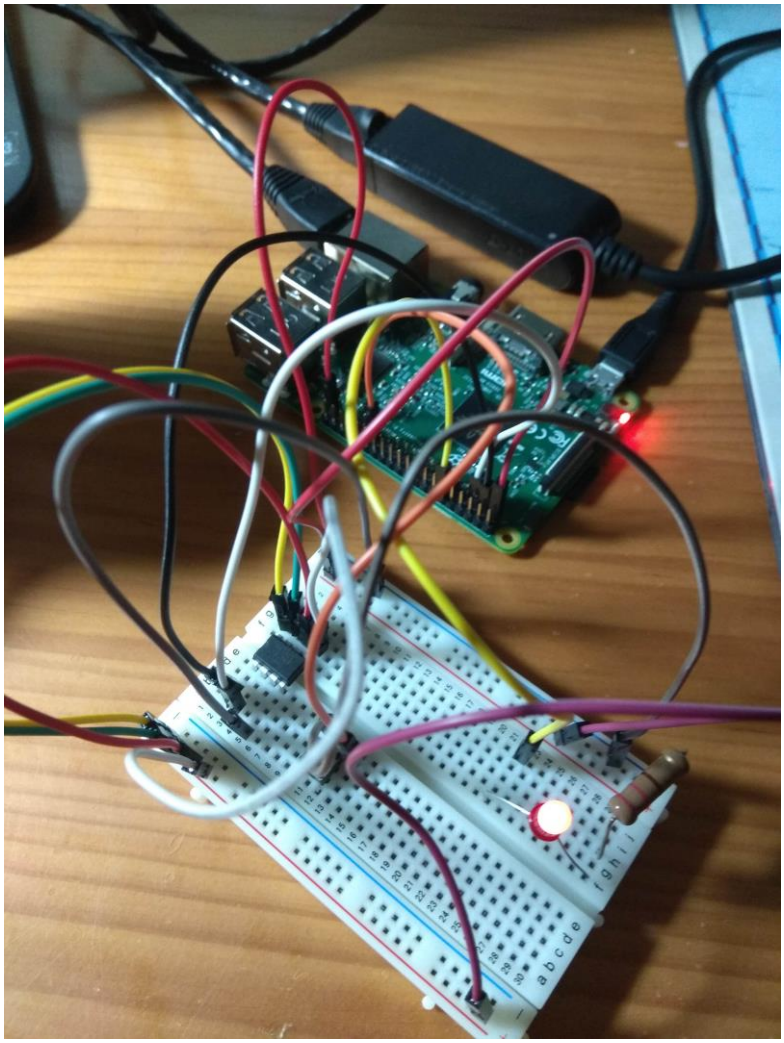


- Sensor de temperatura DS1621

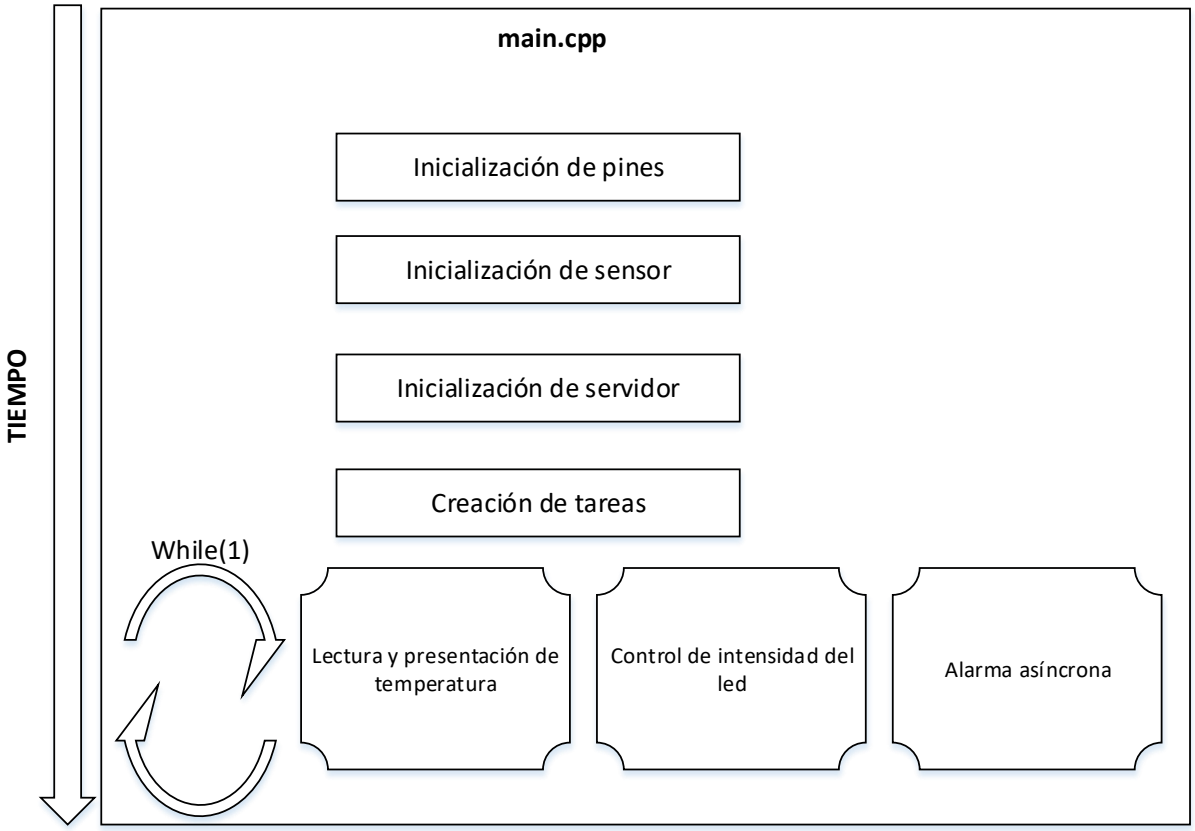


3.Desarrollo

El montaje ha sido algo diferente al propuesto en la memoria inicial debido a que he decidido utilizar una salida PWM diferente a la propuesta, en la siguiente imagen se puede apreciar el montaje final.



Básicamente, la idea del sistema queda esquematizada en el siguiente dibujo, donde se puede apreciar una rutina secuencial de inicialización del sistema para posteriormente crear varios hilos en paralelo que ejecuten las funcionalidades mencionadas.



3.1. LECTURA DE TEMPERATURA

El objetivo de esta funcionalidad consistía en leer la temperatura cada 10 segundos y obtener la temperatura media horaria, guardar las últimas 24 horas y presentar los valores de las últimas 2 horas en la página web.

Por falta de tiempo únicamente se ha cubierto la lectura de temperatura cada 10 segundos y presentación en la página web.

Para ello se ha creado debido configurar el sensor de temperatura correctamente con las instrucciones que proporcionaba el datasheet, identificando además la dirección i2c que le correspondía, 0x4F en mi caso.

```
#define DEV_ID              0x4F                // Direccion del sensor DS1621
```

Luego se han creado dos hilos principales, uno para la lectura de temperatura y otro para atender peticiones a modo de servidor y poder representar la información:

```
pthread_t idHilo_temperatura;
pthread_t idHilo_atenderPeticion;
```

Para la representación de la información se ha seguido el ejemplo propuesto en la memoria con ligeras modificaciones, debidamente comentadas en el código.

3.2. CONTROL DE INTENSIDAD DEL LED

El objetivo de este control del led era ajustar la intensidad del led en función de la temperatura. Para esta aplicación nos encontrábamos con dos problemas; escalar la temperatura para poder ajustar el valor de salida de PWM y el acceso a un recurso compartido como es el bus i2c.

Para el primer problema se ha definido un umbral de temperatura entre 25 y 35ºC de manera que como la salida PWM es de 10 bits, teniendo 1024 valores posibles había que hacer una conversión analógico-digital del dato para poder escribirlo en el pin de salida. Todo esto se ha implementado dentro del hilo `void *ledPWM(void *t).`

Por otra parte para solucionar el acceso a un recurso compartido se ha utilizado un mutex:

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;   // Mutex para acceso al bus
i2c
```

3.3. ALARMA ASÍNCRONA

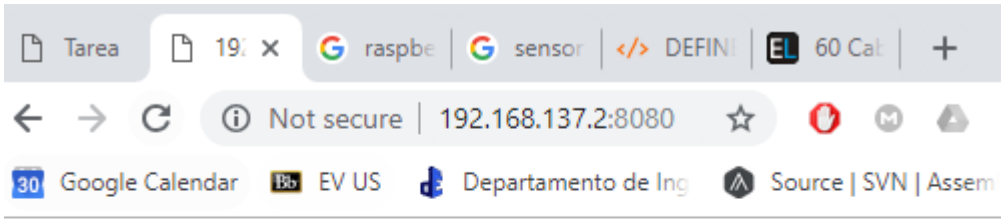
Para ello se ha creado otro hilo que manda mensaje de alarma por UDP, basado en la ayuda propuesta en la memoria.

```
pthread_t idHilo_alarmaUDP
```

4. Validación

4.1. LECTURA DE TEMPERATURA

Ello se ha hecho a través del navegador.

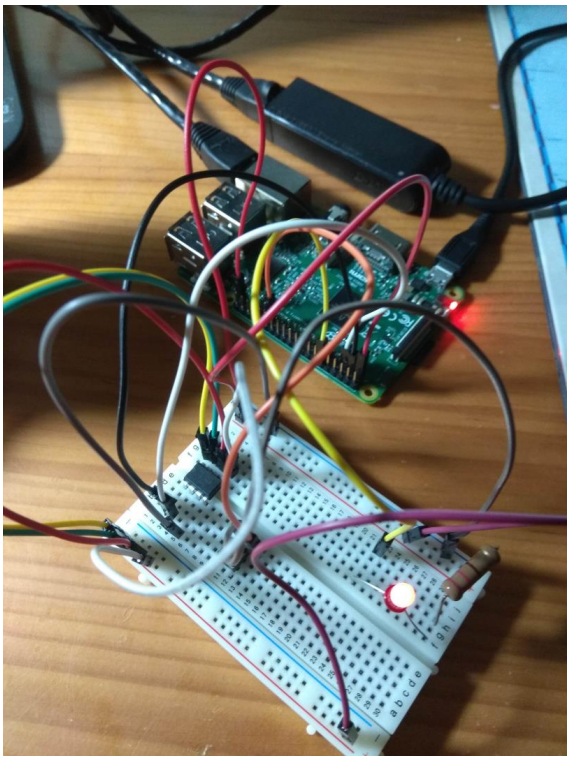


Valor del sensor = 24.000000 ºC

Estado del Pin de alarma = 1

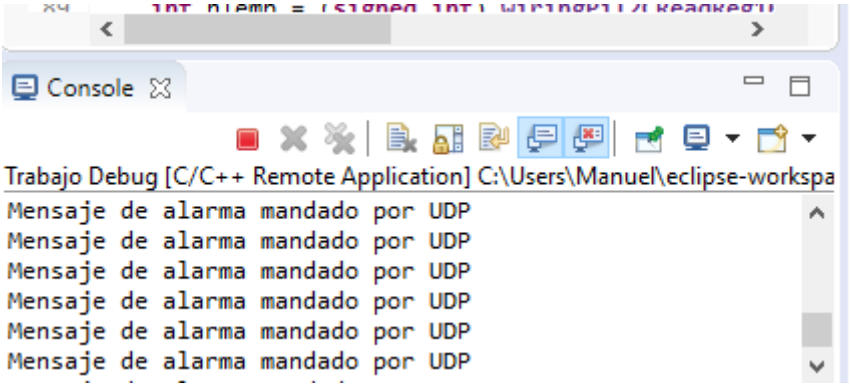
4.2. CONTROL DE INTENSIDAD DEL LED

Para ello se ha utilizado un secador del pelo y ver cómo varía la intensidad del led.



4.3. ALARMA ASÍNCRONA

Para su test se ha imprimido por pantalla un mensaje cada vez que se pulsaba el botón.


Console
Trabajo Debug [C/C++ Remote Application] C:\Users\Manuel\eclipse-workspa
Mensaje de alarma mandado por UDP
Mensaje de alarma mandado por UDP
Mensaje de alarma mandado por UDP
Mensaje de alarma mandado por UDP
Mensaje de alarma mandado por UDP
Mensaje de alarma mandado por UDP

5.Mejoras

Como mejoras se deja implementar la funcionalidad extra de lectura de temperatura almacenando el histórico de las últimas 24h tal y como se define en la práctica lo cual no se ha implementado por falta de tiempo en su realización.