SISTEMAS EMBEBIDOS

Obligatorio 2

Univerisdad ORT Uruguay
Docente: Diego Sáez

Ignacio Lanzani - 143289
Ariel Mordetzki - 222523
Maria Noel Trombotti - 186840

Contenido

Introducción

Basándonos en el primer obligatorio que funcionaba con un diagrama “servidor-placa”, para esta segunda entrega se agregó una nueva dimensión: paralelismo entre placas.

Es decir, ahora existe más de una placa Raspberry que toma medidas y actualiza bases de datos. Esto complejiza ciertas decisiones de diseño, que se explicarán a lo largo del documento. Además, se agrega un nuevo sensor de temperatura.

Cabe destacar que muchas de las explicaciones y fundamentaciones se mantienen igual que para la primera entrega de este sistema.

Descripción general del sistema

En esta sección procederemos a describir el funcionamiento del sistema de medida de temperatura y luz, cuyas funcionalidades deben de ser:

- Medir temperatura y luz mediante un termistor, un LDR y un sensor de temperatura digital.
- Generar persistencia en los datos de medida
- Existencia de una interfaz web capaz de visualizar valores de temperatura y luz instantánea, así como acceder a medidas anteriores y poder manipular parámetros claves del sistema.
- Permitir la conexión segura de hasta 3 placas Raspberry Pi 2, cada una con sus medidas independientes.

Diseño de bloques

El ordenamiento lógico del sistema se puede dividir en 2 bloques fundamentales: cliente y servidor. Estos bloques y funcionan paralelamente entre sí, transmitiendo información y ejecutando funciones que permiten el correcto funcionamiento del sistema.

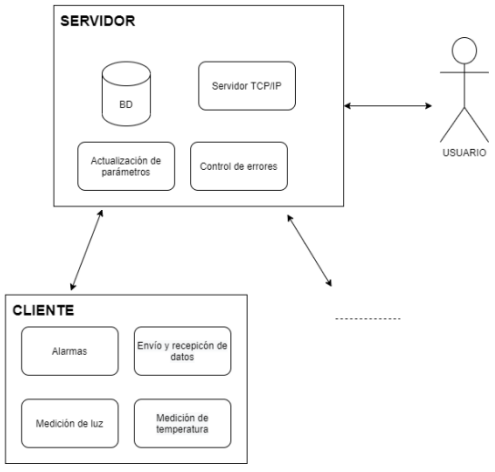
A su vez, cada bloque tiene distintas funciones:

Cliente (corre en las placas Raspberry):

- Toma de datos: toma datos del medio ambiente
- Sistema de alarmas: encargado de enviar alarmas al observador

Servidor:

- Encargado de coordinar el correcto funcionamiento de los bloques mencionados
- Control de errores: encargado de filtrar datos ingresados por el usuario
- Base de datos
- Servidor Web



En líneas generales, el sistema debe ser capaz de poder captar, procesar y guardar correctamente, medidas de 3 placas provenientes del medio ambiente (pueden no estar en el mismo espacio físico ni en la misma red LAN) así como garantizar la seguridad de la conexión.

Servidor

El bloque central de este sistema es el servidor, que tiene las funcionalidades de proveer al sistema de todas las herramientas necesarias para la ejecución de las funcionalidades (creación y adecuación de las bases de datos, disparo de alarmas, tratamiento de la información dentro de las bases de datos y totalidad del manejo web) así como permitir el flujo constante y en paralelo de procesos.

Cliente

Mediante un par de convenientes circuitos que será explicados en profundidad más adelante, las placas son capaces de detectar valores de temperatura y luz provenientes del medio ambiente, además allí también corre el sistema de alarmas y la comunicación con el servidor.

Profundización sobre los bloques

En esta sección se profundizará acerca de bloques mencionados anteriormente.

Bloque servidor

Estudiaremos las funcionalidades que posibilita el bloque servidor

Bases de datos

Dentro del ordenamiento lógico que consideramos “servidor”, se especifican las ordenes de creación de las bases de datos, creándose tres esquemas de base de dato: una para valores de temperatura, otra para valores de temperatura digital y otra para valores de luz.

Las tres bases de datos tienen igual formato, guardaran valores de luz y temperatura, así como la fecha y hora en donde fueron tomados y de que Raspberry provienen estos datos.

Estas bases de datos son creadas con los siguientes criterios:

- Contenido de tipo *string* con bandera “nullable” en falso (lo que significa que en cada celda deberá si o si existir un valor)
- Columna de “date created” en la cual figura el tiempo del sistema, permitiendo guardar la hora y la fecha en la cual se tomó una medida
- Columna de Raspberry en la cual se indica de que placa provino la medida
- Columna de contenido en la cual se indican los valores.

Existen 3 bases de datos: Luz, temperatura analógica (medida con el termistor) y temperatura digital. La página web permite filtrar por identificador de placa, para mostrar el histórico de las medidas de una sola de ellas.

Además, el servidor se encarga de tomar un promedio ponderado de los últimos 5 valores de la base de datos (función “circular”). Esto tiene como finalidad el poder eliminar una posible medida fuera de lugar que ocasione un disparo de alarma, y utiliza el hecho de que la temperatura ambiente tiende a variar de forma lenta lo cual permite que al ponderarla con sus medidas “vecinas” no se pierda información vital.

Ejemplo:

Sea un conjunto de medidas ABCD, en el instante en que se tome la medida “E”, se promediará con las medidas ABCD, que, a su vez, cada una son promedios de las anteriores y así sucesivamente.

Consideraciones:

- La primera medida que se guardará en la base de datos tardará un tiempo 5*(tiempo de medida), la segunda tardará 4*(tiempo de medida) y así sucesivamente hasta obtener 4 medidas, para que la 5ta medida pueda ser obtenida y ponderada correctamente, Las medidas sucesivas serán tomadas a tiempo correcto.
  Este aparente enlentecimiento del sistema de medida es poco trascendente, debido al carácter no volátil de la base de datos: las medidas que se enlentecerán serán las primeras 4 medidas en la base de datos y no en las primeras 4 medidas de una sesión. Es necesario especificar que la primera vez que se instale el sistema de medida, se deberá esperar un tiempo transitorio aproximadamente "14 tiempos de medida" para que el sistema comience a funcionar correctamente.
- Se genera una relación de dependencia entre medidas al ponderarlas entre sí. Como fue mencionado anteriormente, debido a que la temperatura ambiente varia de forma muy lenta, esta dependencia no genera una pérdida de verosimilitud, pero si el sistema es cambiado de ambiente (por ejemplo, pasado de una habitación fría a una caliente) se deberán optar por tomar dos caminos para mejorar la precisión: limpiar completamente la base de datos, o esperar un tiempo transitorio para lograr que las medidas anteriores comiencen a perder peso en el promedio ponderado de las nuevas medidas.
- Para tiempos de medida muy altos (por ejemplo, 1h) el sistema no logra detectar variaciones de temperatura muy finas, debido a que se promedia con valores tomados con diferencias de 1h.

En resumen, el promedio ponderado entre medidas permite eliminar una posible señal de alarma por fallo de medida, pero al precio de una perdida de lentitud del sistema que no parece afectar muy negativamente, dado que la temperatura ambiente varia lentamente.

Manejo web

El servidor se encarga de la total creación de la página web, así como permitir todas las acciones necesarias.

Primero se define a la página con los métodos "POST" y "GET", que indica que la pagina permite el flujo bilateral de información, es decir, podemos mandar y recibir datos.

La página web lee desde archivos txt los parámetros más importantes: temperatura umbral alta, temperatura umbral baja, tiempo de medida y tiempo de alarma, destino (para dirección mail) y fechas (para mostrar datos). También es capaz de modificarlos, si el retorno del "request.method" es "POST", se abre el archivo de texto correspondiente, se sobrescribe en el mismo la información brindada desde la página web y se cierra el archivo, actualizando así de todos los parámetros indicados en las casillas.

Actualización de parámetros

El sistema cuenta con una serie de archivos en formato .txt que contienen información sobre los parámetros de funcionamiento (temperaturas umbrales, tiempo de alarma, tiempo de medida, etc). Procedemos a definir cada uno de ellos:

- **Temperatura alta:** temperatura máxima que debe tolerar el sistema, si la temperatura medida supera este valor, se deberá enviar una alarma.
- **Temperatura baja:** temperatura mínima que debe tolerar el sistema, s la temperatura medida es menor a este valor, se deberá enviar una alarma.

- **Tiempo de medida:** tiempo entre dos medidas, este tiempo deberá ser mayor a 5 constantes de tiempo del sistema R-C mencionado en el capítulo de medición.
- **Tiempo de alarma:** tiempo mínimo entre dos alarmas, definido para no saturar al usuario con alarmas innecesarias una vez que se hayan superado cualquiera de los umbrales de temperatura.
- **Destino:** dirección mail de destino para enviar informes de excesos o defectos de temperatura.
- **Fecha desde y fecha hasta:** fechas en formato "AAAA-MM-DD" que indican umbrales para filtrar el mostrado de datos.

Estos parámetros deben de ser leídos utilizando las funciones "readLine()" y un correcto tratamiento del retorno de la misma (ya que readLine() no siempre devuelve el tipo de variable querido). Un primer acercamiento podría ser definir variables globales que contengan la información de estos txt, para luego ser utilizadas en el flujo del programa, pero esto arrastra un error grave: estos parámetros pueden ser actualizados de forma asincrónica, es decir, el usuario puede actualizar estos valores en cualquier momento. La solución es sencilla: dado que desde la web se actualizan solo los .txt mencionados anteriormente, los parámetros son leídos cada vez que se los vaya a utilizar, para asegurarnos de que estamos utilizando el valor más reciente del parámetro, cubriéndonos de la posibilidad de que un parámetro sea actualizado, y por el flujo mismo del programa se utilice su valor anterior. La desventaja de esto es que se necesita invocar a la función readLine() cada vez que se quiera evaluar si una temperatura pasó el umbral, este es un precio a pagar por intentar mantener el paralelismo.

Paralelismo y programación concurrente

En el "main" del bloque servidor, el mismo se encarga de utilizar la función "threading.Thread" de pyton para disparar hilos que se ejecutarán en paralelo, estos varían desde la creación del servidor, medidas de temperatura y luz así como evaluaciones de la alarma. Hay ciertos hilos que solo pueden ejecutarse cada cierta cantidad de tiempo (por ejemplo, tomar una medida analógica), para solucionar esto, estos hilos se ponen en espera mediante un "sleep" cuyo parámetro varía (en caso de las medidas, este parámetro será "tiempo de medida), en otros casos, este parámetro es una constante elegida por el equipo para asegurar que los procesos no se ejecuten constantemente, sino con un mínimo umbral de ejecución.

Es así que obtenemos dos servidores en simultáneo, uno para el tratamiento de la página web, y otro que únicamente se comunica con las tres placas vía TCP/IP.

Control de errores

Existe una función llamada "genericError()" que se encarga de la totalidad del control de errores que el usuario pueda cometer en el ingreso de los nuevos parámetros. Esta función puede ser pensada como un "filtro" ya que descarta datos mal ingresados antes de que estos ingresen al sistema. (es decir, antes de que estos sean escritos en los archivos .txt)

La función recibe dos parámetros: "tipo" y "dato", donde "tipo" es un identificador utilizado por el programador para poder clasificar que tipo de error se desea evaluar y "dato" es el dato en string puro que proviene del método "GET" de la página web, la función retorna "1" si hubo error y 0 si no hubo error.

Lo primero que se evalúa en todos los casos es si el largo del string "dato" es menor a 0, en caso de serlo, la función termina y se retorna 1 ya que no se aceptan campos vacíos. En caso de no serlo, se procede a separar el análisis dependiendo del parámetro "tipo".

Control de temperatura umbral alta

Para este caso, el parámetro “tipo” será igual a “th”.

Dentro de un bloque try-catch, se convierte el parámetro “dato” a entero y se compara con el valor de temperatura umbral baja del sistema. Si el usuario intentó ingresar un valor de temperatura umbral alta menor al valor de temperatura umbral baja, la función retorna 1. Si el usuario ingreso un texto que no es un entero, el bloque try-catch falla en la conversión a entero y se retorna 1 también.

Control de temperatura umbral baja

Análogo al control de temperatura umbral alta, se compara el valor ingresado con la temperatura umbral alta del sistema, y si el usuario ingresó un valor de temperatura umbral baja mayor al valor de temperatura umbral alta, entonces la función retorna 1. La misma estrategia se utiliza para verificar que el usuario haya ingresado un entero.

Control de tiempo de alarma

Dentro de un bloque try-catch, se evalúa si el tiempo de alarma ingresado es menor a 10 segundos, en caso de serlo, la función retorna 1. En caso de que el usuario ingrese un tiempo de alarma que no es un entero, falla el bloque try-catch y la función retorna 1.

Control de tiempo de medida

Idéntico al control de tiempo de alarma, solo que se compara con el valor 5 segundos en vez de 10.

Control de actualización de destino

Es imposible controlar que la dirección de correo electrónico de destino que ingrese el usuario sea existente, ya que se debería tener acceso a las bases de datos de Gmail, Hotmail,Outlook, etc. No obstante, lo que si podemos es descartar ciertas direcciones invalidas, estas son las que no contienen ni puntos ni arrobas. Seguro que una dirección debe contener al menos un punto y al menos una “@” (por “.com” o “.uy” y por las “@” características de las direcciones).

Para relevar esto, utilizamos el comando “not in” de strings de Python para evaluar si “.” y “@” se encuentran presentes en el texto ingresado, en caso de que no se encuentren ambos, la función generic error retorna 1.

Control de ingreso de fechas

Para controlar que las fechas fueron ingresadas en el exacto formato “AAAA-MM-DD”

Para ello se recorre el texto ingresado y se cuenta la cantidad de separadores, si esta es menor a 2 quiere decir que se ingresó mal el formato de primera (no es necesario controlar el caso mayor a 2 ya que queda contemplado en los controles de correcto ingreso de años, meses y días). Luego se separa el string por separadores, utilizando la función “split()” y pasándole como parámetro el carácter “-“ separador, y obtenemos 3 substrings que corresponden a año, mes y día. Luego, en un bloque de try y catch se convierten a enteros estos sub strings y se evalúa que sean meses, años y días válidos. Por cualquier error posible del usuario al ingresar fechas, la función retorna 1.

Este filtro de errores de usuario es llamado cada vez que se pretende sobreescribir un archivo txt. Si la función retorna 1, entonces el archivo no se sobrescribe y el dato ingresado por el usuario se descarta. Se incluyeron mensajes de error que aparecen en la consola y solo un administrador/programador podrá visualizar.

Funcionalidades extras

Se agregó en la página web ventanas de muestra de la temperatura de la ciudad de Montevideo a modo de referencia, y un graficador de humedad y temperatura instantánea.

Bloque Cliente

Como mencionamos anteriormente, en el cliente se ejecutan varias funciones que se detallarán a continuación.

Cliente TCP/IP :

Las placas están en función de cliente, con lo cual intentarán conectarse permanentemente al servidor correspondiente, ubicado en este caso en una PC Windows.

Las funcionalidades que expresa durante las ejecuciones son:

- **Conexión**: establece una conexión por handshake con el servidor, este reconoce la placa por su dirección IP.
- **Reconexión**: si la conexión se pierde o el server no estaba en funcionamiento al momento de iniciar el programa, el cliente intentará reconectarse.
- **Envío**: envío de datos vía TCP, pero además codificamos la información de tal manera que pueda ser reconocida del otro lado por el server.
- **Recepción**: recibe la información desde el servidor, está puede estar encadenada por ejemplo “L20A30M50T3600: , por lo cual decodificar es parte de la tarea de este bloque.

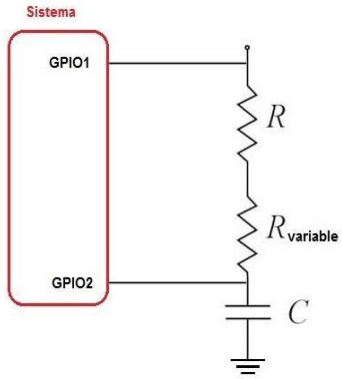
Estas funciones son accesibles para el programa principal que las usara acordemente para enviar los datos al servidor.

Medidas analógicas:

Para medir las temperaturas, se recurre a la medición de tiempos de descarga de un sistema R-C convenientemente definido.

Ejemplificaremos con un diagrama:

El algoritmo consiste en poder medir la resistencia variable en función del tiempo de carga del capacitor. Si podemos obtener un valor de resistencia variable, entonces podremos obtener el valor de temperatura o luz, ya que esta resistencia varia con estas magnitudes.

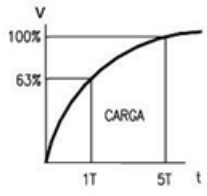


La constante de tiempo del sistema será entonces:

$\tau = (R + R_{variable})C$ para cumplir el requerimiento de que el mínimo tiempo de medida sea de 5s, alcanza con elegir valores de R y C tal que $5 \quad \tau < 5s$.

El sistema debe encargarse de medir cuanto tiempo se tarda en llegar a los 5τ, para luego poder despejar sencillamente el valor de $R_{variable}$

Para lograr este objetivo, se setea GPIO1 como salida y GPIO2 como entrada, luego se dispara un “1” lógico. Esto causará que la red R-C comience a cargarse, adjuntamos una imagen de la carga de una red R-C:



Cuando la entrada GPIO2 detecte un “1” lógico, podemos asumir que habrán transcurrido 5τ, despejamos $R_{variable}$ y con eso podemos despejar el valor de temperatura. Luego, debemos esperar un tiempo para que el capacitor se descargue. La duración de este proceso deberá ser menor a 5 segundos (tiempo mínimo entre medidas) para poder tomar medidas cada 5 segundos como mínimo.

Para poder medir este tiempo, simplemente se toma el tiempo del sistema antes de comenzar a cargar, luego se toma el tiempo del sistema cuando la GPIO2 detecte el 1 lógico y se realiza la diferencia de estos valores.

Estos métodos fueron utilizados para medir valores de temperatura y luz, con la única diferencia en el despeje de temperatura o de luz, una vez obtenido el valor de $R_{variable}$ que se puede despejar de la ecuación de carga del circuito RC:

$$R_{variable} = \frac{5\tau}{log\left(1 - \frac{V_{ref}}{V_{final}}\right) * C} - R$$

Donde:

- $R_0 =$ Resistencia del termistor a la temperatura T_0
- $5\tau =$ Tiempo que tarda el circuito en cargarse
- $V_{ref} =$ voltaje de referencia inicial (para este caso, fue puesto en 1.10V)
- $V_{final} =$ voltaje de final de carga, otorgado por la GPIO1 (3.27V medidos)
- $C =$ Capacitor

Esta resistencia obtenida, podrá variar en función de la temperatura o de la luz, una vez obtenido este valor, alcanza con despejar el valor de temperatura o luz de las ecuaciones características del termistor o el LDR.

Nota: se agregaron factores de corrección heurísticos correctores, debido al hecho de que las placas presentan un error intrínseco, dado que desconocemos como el sistema operativo asigna recursos. El factor elegido fue de "0.10", constante que mejora las medidas, elegido por el equipo a prueba y error

Temperatura:

$$R(T) = R_0 e^{\left(\frac{1}{T} - \frac{1}{T_0}\right)}$$

Donde:

- $R_0 =$ Resistencia del termistor a la temperatura T_0
- $T_0 =$ Temperatura de calibración, suele ser típicamente 25°C (298°K)
- $T =$ Temperatura en °K
- $B =$ Índice de sensibilidad

Basta con despejar la temperatura en grados Kelvin de esta ecuación.

Luz

$$R(L) = R_0 \left(\frac{L_0}{L}\right)^{\alpha}$$

Donde:

- $R_0 =$ Valor de R_L un punto dado
- $L_0 =$ Nivel de luz utilizado para hallar el modelo
- $\alpha =$ Constante del material
- $L =$ Luminosidad en lux

Análogamente, basta con despejar el valor de lux de esta ecuación.

Para nuestro caso, $L_0 = 300$, nivel de luz ambiente, $\alpha = 0.8$ y $R_0 = 1000$ (resistencia medida con un tester a luz ambiente).

Los resultados son enviados utilizando la función "sendall" de sockets, una vez seteados, bindeados y aceptados correctamente con el servidor.

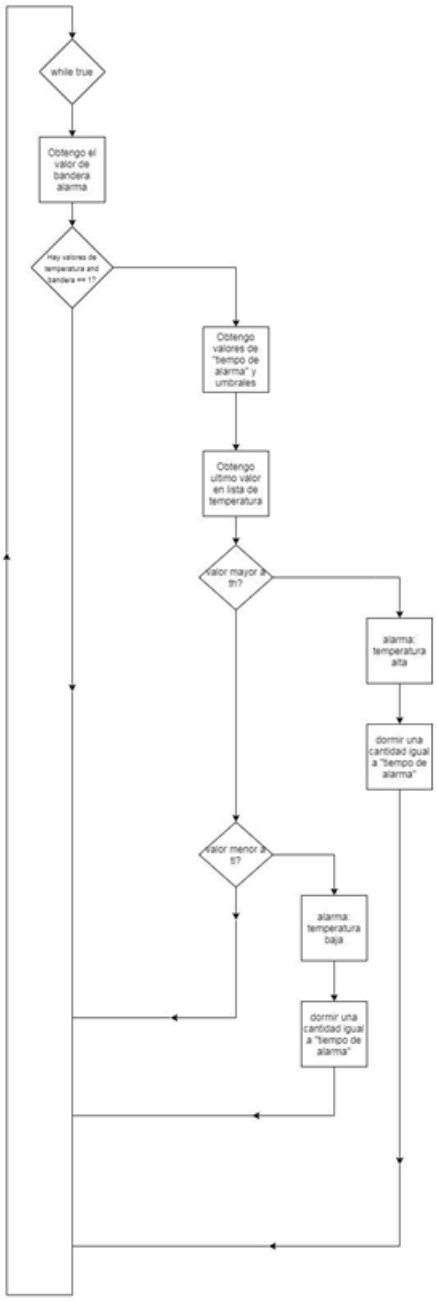
Medidas digitales:

Importando la librería "W1ThermSensor" y creando una estructura "sensor = W1Thermsensor()" podemos pedir valores de temperatura utilizando el método "sensor.get_temperature()". Esta temperatura será enviada al servidor utilizando a la función "Envio", que recibe un dato y un tipo de datos, y envía en un conjunto de bytes la secuencia "tipo -dato" donde dato es el valor de temperatura y "tipo" es un encabezado. Utilizando la función "sendall()" de sockets, se envía la secuencia "tipo-dato" que construyo la función "Envio".

El sensor digital consta de 3 "patas". Una es la tierra, otra es vcc y la del medio es la que se conecta a la placa dada.

Alarmas

El sistema de alarmas fue implementado como una única función dentro del código del bloque servidor, no obstante, conviene ser pensada como un bloque aparte.



Existe una función llamada “alarma TEMP” que respeta el siguiente flujo:

La función provee al sistema de la capacidad de detectar valores de temperatura fuera de los rangos umbrales y disparar alarmas siempre que se cumpla que el tiempo entre alarmas sea mayor al tiempo mínimo especificado por el sistema.

Esta función es representada por un hilo en paralelo, que se está siempre ejecutando paralelamente al resto del funcionamiento del programa, de ahí la necesidad de definir un bloque “alarma”.

El sistema de alarmas primero obtiene el valor de “bandera alarma”, que es una bandera que vale “1” si la alarma está encendida o “0” si la alarma está apagada. Este valor se encuentra grabado en un archivo en formato “.txt” debido a que el usuario puede apagar el sistema de alarma desde la página web de forma asincrónica y cuando lo hace, la página web actualiza el archivo que contiene esta bandera.

Al obtenerlo, cada vez que evaluamos si disparar una alarma o no, nos aseguramos de que el estado de las alarmas es exactamente el deseado por el usuario en ese instante de tiempo.

Luego, evaluamos si existen valores tomados de temperatura (condición que casi siempre será verdadera) y si la bandera especificada anteriormente está en “1” (es decir, si la alarma esta encendida). En caso que sea verdadero, procedemos a leer de los archivos .txt correspondiente los valores umbrales de temperatura (guardados en txt por la misma razón que la bandera de alarmas), y por último leemos el último valor en la lista de las temperaturas (último valor medido).

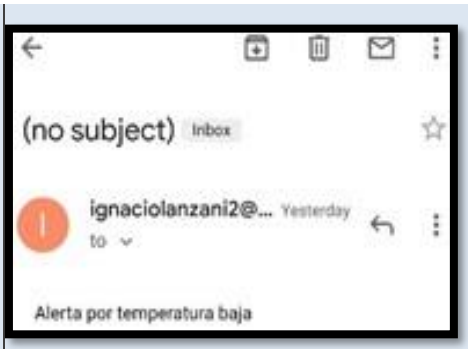
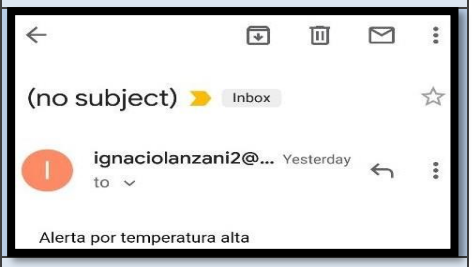
Más tarde comparamos el último valor medido con los valores umbrales: si el valor se encuentra fuera del umbral, se dispara la alarma correspondiente y luego se pausa el hilo durante un tiempo igual a “tiempo de alarma”. Esto no frena al resto del programa ya que este flujo corre en un hilo en paralelo, de ahí la definición de “bloque”. La necesidad de pausar el hilo radica en no mandar alarmas en un tiempo menor a tiempo de alarma, es decir, una vez mandada una alarma, no se volverá a enviar ninguna alarma hasta que no haya pasado el tiempo de alarma. Al terminar cada uno de estos sub-flujos, se retorna al loop infinito inicial y se repite el flujo. Cabe destacar que el diagrama asume que todos los parámetros están ingresados

correctamente. Si se deseara documentar el flujo del algoritmo agregando control de errores, se perdería la idea principal.

Casos de prueba

Funcionamiento sin errores

Caso de prueba	Respuesta deseable	Respuesta obtenida
Medir un valor de temperatura ambiente	Temperatura medida y registrada correctamente	Temperatura medida y registrada correctamente
Medir un valor de luz ambiente (habitación Ignacio)	Valor de luz medido y registrada correctamente	Valor de luz medido y registrada correctamente
Actualizar parámetro “tiempo medida”	El sistema actualiza el archivo “TiempoDeMedida.txt”	El sistema actualiza el archivo “TiempoDeMedida.txt”
Actualizar parámetro “tiempo de alarma”	El sistema actualiza el archivo "tiempoDeAlarma.txt"	El sistema actualiza el archivo "tiempoDeAlarma.txt"
Actualizar parámetro “temperatura umbral alta”	El sistema actualiza el archivo: "AlarmaTEMPHig.txt"	El sistema actualiza el archivo: "AlarmaTEMPHig.txt"
Actualizar el parametro "Temepratura umbral baja"	El sistema actualiza el archivo: "AlarmaTEMPlow.txt"	El sistema actualiza el archivo: "AlarmaTEMPlow.txt"
Actualizar el parámetro “destino”	El sistema actualiza el archivo “FechaDesde.txt”	El sistema actualiza el archivo “FechaDesde.txt”
Actualizar el parámetro "fecha desde"	El sistema actualiza el archivo "FechaHasta.txt"	El sistema actualiza el archivo "FechaHasta.txt"
Actualizar el parámetro “fecha hasta”	El sistema actualiza el archivo “Destino.txt”	El sistema actualiza el archivo “Destino.txt”

Disparo de alarma por temperatura baja	El sistema envía un correo electrónico indicando la situación	
Disparo de alarma por temperatura alta	El sistema envía un correo electrónico indicando la situación	
Solicitar valores de medida entre dos fechas	Sistema muestra valores de medida entre dos fechas	Sistema muestra valores de medida entre dos fechas
Desactivar alarmas	El sistema no emite alertas de alarma	El sistema no emite alertas de alarma
Refrescar la medida manualmente desde web	Sistema actualiza valores mostrados en la web	Sistema actualiza valores mostrados en la web
Encender el led manualmente	Sistema enciende el led	Sistema enciende el led
Encender el led desde la página web	Sistema enciende el led	Sistema enciende el led

Funcionamiento con errores

Caso de prueba	**Respuesta deseable**	**Respuesta obtenida**
Ingresar un valor de temperatura alta menor a temperatura baja	El sistema ignora este valor y alerta la situación	Error! Intentaste ingresar un th de 2 ºC cuando el tl es 5ºC
Ingresar un valor de temperatura baja menor a temperatura alta	El sistema ignora este valor y alerta la situación	Error! Intentaste ingresar un tl de 240 ºC cuando el th es: 80 ºC
Ingresar un tiempo de alarma menor a 10 segundos	El sistema ignora este valor y alerta la situación	Error! Intentaste ingresar un tiempo de alarma de: 3s cuando los tiempos de alarma no deben ser menores a 10s

Ingresar una dirección de correo de destino no valida	El sistema ignora este valor y alerta la situación	Error! Intentaste ingresar la dirección: Me Llamo Ariel, que no es válida
Ingresar un tiempo de medida menor a 5 segundos	El sistema ignora este valor y alerta la situación	Error! Intentaste ingresar un tiempo de medida de: 3 s cuando los tiempos de medida no deben ser menores a 5s
Ingresar campos vacíos	El sistema ignora este valor y alerta la situación	Error! No puedes ingresar campos vacíos!
Ingresar la fecha: "ABCDEFG"	El sistema ignora este valor y alerta la situación	Error! Debe ingresar la fecha con separadores
Ingresar la fecha: "ABCD-EF-GH"	El sistema ignora este valor y alerta la situación	Error, se esperan enteros
Ingresar un año no entero	El sistema ignora este valor y alerta la situación	Error, se esperan enteros
Ingresar un mes no entero	El sistema ignora este valor y alerta la situación	Error, se esperan enteros
Ingresar un día no entero	El sistema ignora este valor y alerta la situación	Error, se esperan enteros
Ingresar un año entero pero inválido	El sistema ignora este valor y alerta la situación	Ingrese un año válido
Ingresar un mes entero pero inválido	El sistema ignora este valor y alerta la situación	Ingrese un mes válido
Ingresar un día entero pero inválido	El sistema ignora este valor y alerta la situación	Ingrese un día válido
Ingresar cualquier combinación de los errores de fecha mostrados	El sistema ignora este valor y alerta la situación	El sistema ignora este valor y alerta la situación

Conclusiones

El poder de procesamiento de las Raspberry permite crear sistemas embebidos diversos, en gran parte al hecho de que pueden correr sistemas operativos. Esto permitió a los clientes realizar varias tareas en paralelo utilizando Python de manera rápida, sencilla y eficiente.

Sin embargo, desarrollar en esta versión de Raspberry hizo que trabajar directamente sobre la placa no fuese una opción, dada la lentitud general del sistema. Por este motivo, optamos por usar Pycharm como IDE principal y pasar el código conectándonos de manera remota a la placa. Estas decisiones de trabajo nos permitieron acelerar la velocidad al momento de desarrollar y debugear el sistema.

Por el mismo motivo decidimos pasar el servidor web a una computadora en Windows (con mayor capacidad de procesamiento), para que las placas estuvieran más libres, y que el manejo de las múltiples funciones que involucra el Servidor no entorpeciera las tareas de cada uno de los clientes.

Por otra parte, Python como lenguaje es rápido, versátil y con una velocidad de desarrollo muy alta. Considerando que ninguno de los participantes de la tarea tenía experiencia trabajando con páginas web y bases de datos. Creemos que fue una buena experiencia.

La utilización de sensores analógicos, si bien relativamente más baratos que otras opciones, es en el caso de una Raspberry corriendo un sistema operativo que no es en tiempo real, una opción extraña pero que arroja resultados no muy distintos (en promedio) al de un sensor como el Dallas 18B20.

Pensamos también que estas tareas (cliente) podrían haberse implementado también con Arduino uno o un esp32, salteando algunos problemas.

Este proyecto nos permitió acceder al mundo de IoT, creando un sistema de medición completo de temperatura y luz, reflejar los valores en una base de datos todo para que sea presentado de manera eficaz a un usuario, en cualquier parte del mundo a través de una página web. Un sistema embebido que hoy por hoy, es muy deseado y cada vez más accesible.

Anexos

Observaciones:

La Raspberry cuenta con un sistema operativo que no es de tiempo real, ya que el procesador asigna recursos a cada una de las tareas siguiendo criterios de optimización propios, al medir valores de resistencia basados en tiempos de carga y descarga, no existe ningún tipo de garantía sobre la exactitud de la medida que uno está tomando. Este detalle no es tan significativo ya que el tiempo de medida mínimo es de 5 segundos y el error producido por la distribución de tareas del procesador es casi despreciable, pero si se desease aumentar la precisión, se debería explorar otras alternativas de medida que sean independientes del tiempo (por ejemplo, utilizando un sensor de medida digital, o estudiando la conversión A/D del sistema con algún otro acondicionamiento para el termistor/ldr)

Error de autocalentamiento del termistor

Los termistores se auto calientan debido a fenómenos eléctricos, generando un error en la medida, denominado error de autocalentamiento, procedamos a calcular el error de autocalentamiento del termistor.

El autocalentamiento del termistor esta dado por la potencia que disipa, y el peor de los casos (es decir, cuando disipa la mayor cantidad de potencia) está dado por el teorema de máxima transferencia de potencia y se da cuando $R(T) = R_{fija}$.

Calcularemos el autocalentamiento para este caso (que es el peor de los casos) para dar una cota superior al mismo, es decir, el autocalentamiento para una temperatura dada siempre será menor al autocalentamiento en el peor de los casos.

Utilizando estos enunciados podemos igualar:

$$P_{ntc} = \frac{\left({}^{V_{cc}}/_{2}\right)^2}{R(T)} = \frac{\left({}^{V_{cc}}/_{2}\right)^2}{R_{fija}}$$

La potencia sobre el termistor está relacionada con el autocalentamiento de la siguiente forma:

$$P_{ntc} = \delta \Delta T$$

Donde δ es un coeficiente de disipación térmica que depende del termistor y ΔT es el incremento de temperatura interna debido al autocalentamiento del termistor.

Para calcular el coeficiente de disipación térmica, debemos aplicar la siguiente formula:

$$P = \delta(T2 - T1)$$

Es decir, con una potencia suministrada P, a temperatura ambiente $T1$, elevando la temperatura del termistor a un valor de $T2$, el coeficiente será de la forma:

$$\delta = \frac{P}{T2 - T1}$$

Ahora, intentemos despejar el incremento de temperatura debida al autocalentamiento en función de los parámetros conocidos

$$\Delta \mathrm{T} = \frac{P_{ntc}}{\delta} = \frac{\left(\frac{V_{cc}}{2}\right)^2}{\delta * R_{fija}}$$

El valor de $\mathrm{V_{cc}}$ es el suministrado por la GPIO, que es de 3.3V y el valor de R_{fija} usado es de 9.1kΩ, el valor de δ depende de las condiciones de la media y la temperatura ambiente.

Para el termistor usado, a temperatura ambiente un valor estándar para δ es de 7mW/°K lo que desprendería un valor de error por autocalentamiento de 0.04 °C valor despreciable ya que es menor a la décima de grado. Para mejorar este valor se podría considerar aumentar el valor de R_{fija} (lo que conlleva un reajuste del valor del capacitor).

Diagrama de pines y circuito