UNIVERSIDAD DE COSTA RICA

TCU 629: Aplicación de soluciones automatizadas o robóticas en MiPyMEs e organizaciones sociales en el contexto del movimiento industrial 4.0

Estudiante: Andrés Castro Núñez

Carné: B21604

Proyecto: Automatización de la monitorización de temperaturas de un calentador solar de agua.

Documentación Técnica del Proyecto

Introducción

El proyecto desarrollado consiste en un sistema integral que permite monitorear, de manera automatizada, el funcionamiento de un calentador solar de agua. Dicho sistema consta de un dispositivo físico con la capacidad de medir la temperatura del agua del calentador y, a través de internet, enviar la información. El sistema consta además de una solución de Software que puede recibir y manipular la información del dispositivo, así como reaccionar a ella para generar alertas y reportes.

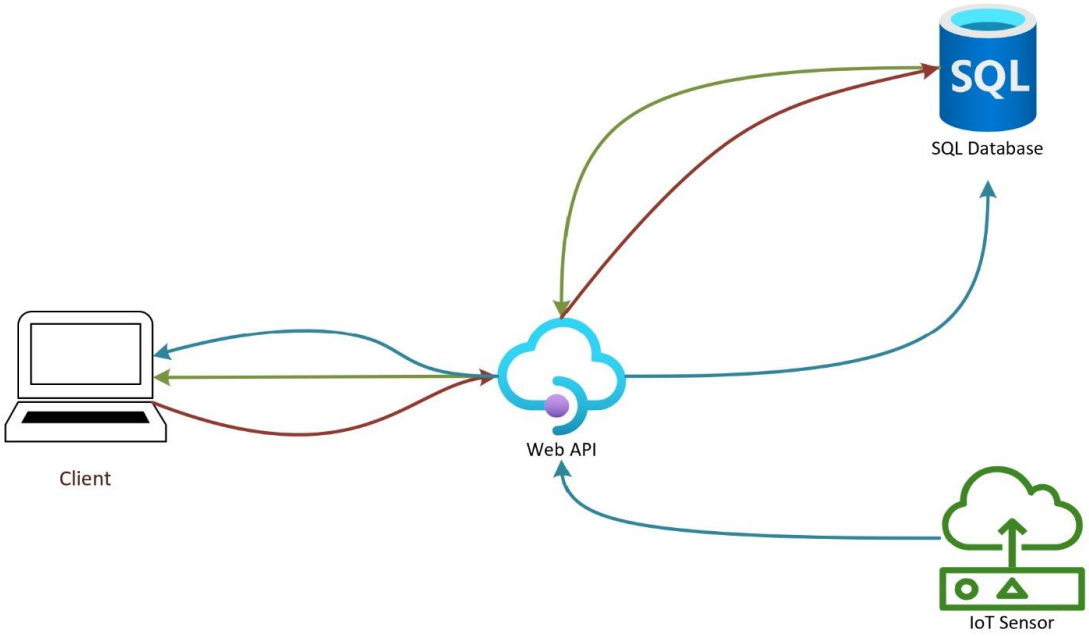
La creación de este sistema nos abre muchas puertas en el uso del Calentador Solar de agua. La automatización de las mediciones nos permite analizarlo en detalle sin necesidad de invertir tanto tiempo. Tener la información recopilada nos permite generar análisis estadísticos que facilitan la toma de decisiones.

Dentro de las posibilidades, podemos analizar la eficiencia del sistema en distintas condiciones. Podemos encontrar información útil de su funcionamiento, como, por ejemplo, a que hora del día se alcanza la temperatura deseada para un baño con agua caliente o tibia, según se desee. Nos permite mejorar la seguridad y la comodidad, al poder monitorear, en tiempo real, la temperatura del dispositivo. Esto puede ayudarnos a evitar quemaduras o choques térmicos en los usuarios.

Software

Arquitectura General

La solución desarrollada consta de varios componentes de Software con funciones específicas que actúan en conjunto. A continuación, se presenta un diagrama general de la solución y se procede a explicar cada parte de esta.



API

El corazón de la solución es una web API. Este elemento corresponde a la parte del software que se encanta de interconectar las demás piezas. A su vez, posee la lógica necesaria para generar dichas conexiones.

Es decir, se encarga de procesas la información recibida y tomar las decisiones de que hacer con ella. Por ejemplo, le envía al usuario la información que solicita o guarda las mediciones que recibe del dispositivo que monitorea la temperatura del agua.

El código de la API se puede encontrar en este repositorio.

Base de Datos

El sistema usa una base de datos relacional (SQL) que se encuentra alojada en un servicio de la nube Azure. El enlace a la base de datos y las credenciales se encuentran al final del documento.

La base de datos se encarga de almacenar la totalidad de la información de la aplicación. Guarda los sensores y dispositivos registrados, los tipos de dispositivos, todas las mediciones realizadas por los sensores y más.

Cliente o Aplicación Web

Esta pieza del software corresponde a la interfaz que se utiliza para configurar y configurar la información del sistema. Es acá donde los usuarios finales interactúan con el resto de la solución.

En esta aplicación es posible registrar nuevos dispositivos y sus configuraciones, así como consultar toda la información asociada a los mismos. Es decir, acá es donde se analizan las temperaturas, sus estadísticas, etc.

Se está trabajando en un manual que explica del uso y las capacidades de la interfaz. Consultar al respecto al encargado del TCU. El código para la interfaz gráfica se puede consultar en este repositorio.

Dispositivo IoT

Como se verá más adelante, se creó un dispositivo con capacidad de envío de datos. Este posee programación diseñada para que envíe la información en el formato y en la frecuencia correctas para funcionar con la API. El código del dispositivo se puede encontrar en este repositorio.

Tecnologías

Front-End

La tecnología principal del Front-End es **React**. Para mantener un mejor orden y control en el código, se está utilizando **TypeScript.** Para la parte de los estilos y manejo en general del CSS se está utilizando **Tailwind.** De la mano con esto, se utilizan las librerías **FluentUI** y **ReactEcharts**. FluentUI nos aporta elementos visuales como botones mientras que ReactEcharts se utilizó principalmente para la parte de generación de gráficos y otros elementos visuales.

Back-End

El Back-End está basado en el Framework .**NET.** Se utiliza **ASP.NET Core** para la creación de la API. El lenguaje utilizado es **C#**. Se adiciona **Entity Framework Core** para manejar la comunicación con la base de datos y la creación de tablas. Por último, se agrega **Identity** para el manejo de usuarios, roles y autenticación.

Arquitectura Back-End

En la sección “Arquitectura General” se explican los componentes individuales que componen la solución. En esa sección vamos a adentrarnos en el componente “API”, de modo que se pueda clarificar su funcionamiento y su comunicación con los demás elementos involucrados.

La API como tal se conforma de 3 proyectos individuales que actúan de manera conjunta. Juntando los proyectos podemos diferenciar 3 capas dentro de la API. Estas capas son: Web API, Servicios y Acceso a Datos. Uno de los proyectos no se considera una capa si no un proyecto de apoyo para la transferencia de datos. A continuación, se describen dichos proyectos y capas.

API

El proyecto API se encarga de la comunicación con el Cliente (Front-End) y el dispositivo IoT. Su principal tarea es recibir las peticiones que estos 2 elementos soliciten. Por ejemplo, el dispositivo le envía a la API las mediciones de temperatura. La API debe recibir esta petición y trasladarla a la siguiente capa. Una vez que se tenga un resultado, se lo comunica de vuelta al dispositivo.

Similarmente, toda la información que se envía o recibe en el Cliente pasa por la App. Desde guardar un dispositivo, verificar un usuario o ver las temperaturas, la API es la encargada de recibir y enviar la información correspondiente. Cabe destacar, que la tarea de la API es esa comunicación y no es realmente dueña de la información ni se encarga de procesarla, si no nada más de transmitirla.

Servicios

Dentro del proyecto API existe una capa de servicios. La capa API se comunica con la capa de servicios para solicitarle lo que necesite. Por ejemplo, la API podría requerir todas las temperaturas de un determinado sensor o guardar un nuevo sensor. La API le comunica esto a la capa de servicios y es acá donde se decide que hacer con la información. Si, por ejemplo, hay varias bases de datos, la capa de servicios debe decidir con cual base de datos necesita comunicarse o, si, por el contrario, la información debe solicitarse de otra fuente. También, si la información requiere procesarse previamente antes de ser guardada (o de vuelta a la API), ese procesamiento se realiza en esta capa.

Es importante mencionar que la capa de servicios no es la que lee o guarda en la base de datos, si no que decide hacia donde se redirige la información procesada. En caso de que la información deba enviarse o leerse de la base de datos, nos comunicamos con la siguiente capa.

Acceso a Datos

La capa de Acceso a datos es la capa encargada de la comunicación directa con la base de datos. Una vez que la capa de servicios le indique a esta capa lo que necesita, la capa de acceso posee la lógica necesaria para extraer dicha información de la base de datos. Naturalmente, posee también la lógica para guardar la información en la base de datos.

DTO

El proyecto DTO no es una capa, si no una herramienta. DTO hace referencia a “Data Transfer Object”. Estos son clases cuya estructura define piezas de información que se desean mover entre distintas capas de la aplicación. Son similares a cajones encargados de mover información con una forma específica. Por ejemplo, cuando un usuario nuevo se registra, es necesario un email, un nombre de usuario y una contraseña. Existe un DTO cuya función es permitir que esos 3 datos en conjunto se movilicen entre las capas, según sea necesario.

A modo de resumen, analicemos que ocurre cuando se guarda una medida de temperatura:

1. El dispositivo IoT envía la información a la API. Esto incluye la temperatura, la fecha, un identificador para el sensor que envía y otros detalles.
2. La API recibe la información, valida que tenga la forma correcta y le indica a la capa de servicios que necesitar guardarse la medición en la base de datos.
3. En nuestro software, la información de la temperatura está asociada tanto al sensor que la mide como al microcontrolador al que está conectado. No obstante, la medición enviada solo indica el identificador del sensor. La capa de servicios tiene entonces que obtener la información faltante, antes de poder solicitar que se guarde la medición.
4. La capa de servicios solicita a la capa de acceso la información del sensor y del microcontrolador a la que está asociado dicho sensor.
5. La capa de acceso a datos se conecta a la base de datos para obtener esta información y se la retorna a la capa de servicios.
6. Ahora, la capa de servicios toma la información que necesitaba del microcontrolador y el sensor para adjuntarla con la información de la medición de temperatura recibida. Para ello, existe un DTO que está esperando exactamente esa información.

7. Al completar la información, la capa de servicios la envía a la capa de acceso a datos.
8. La capa de acceso a datos recibe la información completa y en el formado esperado (a través del DTO) y procede a enviarla a la base de datos. Terminando entonces el proceso de guardado de una medición de temperatura.

Dispositivo Físico

Para lograr el objetivo de monitorear las temperaturas, era necesario crear un dispositivo que pudiera tomar los datos de manera automática y enviar la información por internet. Para ello, se creó un dispositivo que consiste en un microcontrolador, un sensor de temperatura y una fuente de energía.

Partes y Materiales

Sensor de Temperatura

Se utiliza un sensor de temperatura DS18B20 debido a su bajo costo y fiabilidad. Lo más importante es que este es además sumergible, lo que permite la toma de temperatura directamente en el agua.

Microcontrolador

Para controlador, se elige usar una placa de desarrollo con un procesador ESP32. Se elige esta placa, no solo por su bajo costo, si no que además posee wifi integrado. Además, estas placas son compatibles con Python y Arduino, lo que facilita enormemente el proceso de desarrollo, al existir mucha información de acceso libre en internet.

Baterías

Existen varias opciones para energizar el dispositivo. Se desea que sea portátil y autónomo, de modo que no dependiera de tener una toma de corriente cercana. Se determinó que la manera más sencilla para resolver este tema era el utilizar una batería portátil para teléfono convencional (powerbank).

PLA

Se utilizó plástico PLA para crear una carcasa que permitiera tener todos los elementos físicos en juntos, facilitando además las conexiones externas. Dicha carcasa fue fabricada con impresión 3D. En la sección de enlaces se pueden encontrar los sólidos editables así como los archivos preparados para impresión.

Circuito Personalizado

El sensor en cuestión no se puede conectar directamente a la placa de desarrollo, si no que requería de la creación de un pequeño circuito para generar la conexión. Para ello, se utilizaron los siguientes materiales.

Placa Microperforada

Material común en electrónica que nos permite soldar elementos y conectarlos entre sí.

Resistencia

El circuito requería una resistencia de **4.7KOhms**.

Además, fue necesario el uso de soldadura y cables, para completar el circuito.

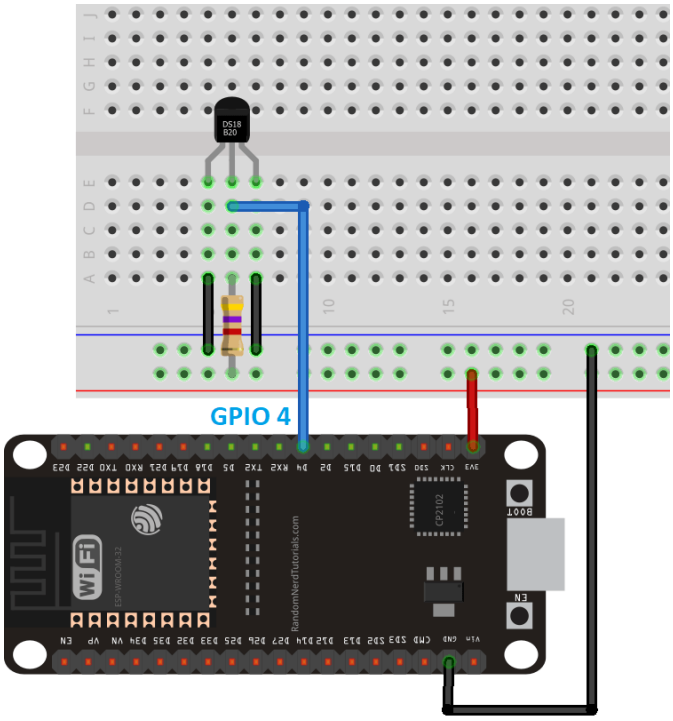


Diagrama de conexión del Sensor y la Placa.

Código

Para lograr que el dispositivo funcionara era necesario programarlo. El código puede encontrarse en este repositorio. Para más información, leer el archivo “Readme” del repositorio, donde se indica para que es cada archivo. Además, cada código está ampliamente comentado, de modo que se pueda estudiar, comprender y modificar fácilmente.