**Taller de Proyecto II**

**Informe Final**

**Sistema de monitoreo y control de temperatura en tiempo real utilizando con MQTT y GPRS (G2)**

**Grupo de Desarrollo**

* Dehan, Lucas – 565/1
* Duarte, Víctor – 1055/7
* Kleinubing, Hernán – 1614/6
* Palacio, Constantino – 1806/2

**04/12/23**

# **1.- Descripción General del Proyecto**

**Enfocar la descripción mayormente en términos de objetivos/funcionalidad. Identificar a) qué fue propuesto originalmente, b) qué fue propuesto/modificado a partir de las evaluaciones de la cátedra y c) qué fue determinado por otras decisiones de desarrollo hasta completar el proyecto tal como se entrega. No detallar lo que luego estará en secciones subsiguientes de este informe, sino dejar las indicaciones correspondientes en esta sección para ser luego explicadas en el resto del informe.**

## ***Introducción***

En un mundo cada vez más interconectado, el monitoreo preciso de la temperatura desempeña un papel fundamental en numerosos aspectos de la vida cotidiana y en diversas industrias.

Sin embargo, el acceso a estos datos en tiempo real ha sido un desafío constante, especialmente en ubicaciones remotas o en áreas donde la comunicación es limitada. La falta de una solución efectiva para la adquisición y visualización de datos en tiempo real ha llevado a ineficiencias operativas, pérdida de productos sensibles a la temperatura y, en algunos casos, incluso a situaciones de seguridad crítica.

El presente proyecto se centra en abordar esta problemática mediante la creación de un dispositivo electrónico capaz de medir con precisión la temperatura en diversos entornos y transmitir esos datos a través de una conexión GPRS (*General Packet Radio Service*) haciendo uso de la red de telefonía celular. La visualización de estos datos se realizará de manera sencilla y accesible por medio de, por ejemplo, una aplicación web.

## ***Objetivos Principales***

Originalmente se propuso como objetivo principal del proyecto la realización de un prototipo como prueba de concepto que sea verificable en el ámbito del aula con la placa Arduino de un sistema capaz de realizar distintas tareas como:

1. Medir la temperatura de un ambiente.
2. Usar las redes de telefonía celular mediante un módulo GPRS para conectar la placa Arduino a un servicio de IoT. En este proyecto se usará el servicio EMQX con el protocolo MQTT para enviar los sensados.
3. La aplicación web será una interfaz de usuario la cual obtendrá y mostrará los datos proporcionada por EMQX, proporcionando gráficos y mensajes de texto al usuario de las temperaturas y una interfaz que le permitirá establecer dicha temperatura dentro de un rango máximo y mínimo.

Respecto al desarrollo de este objetivo, no hubo cambios significativos en el sistema propuesto y se completó el desarrollo.

## ***Objetivos Secundarios***

Al inicio del desarrollo, se propusieron algunos objetivos secundarios como mejoras o ampliaciones al cumplir los objetivos principales. A continuación se nombran las mejoras propuestas junto con el avance realizado por el grupo:

### 1.3.1. Adición de módulos controlados por el sistema

Podrían incluirse, una vez probado el funcionamiento del prototipo, dispositivos que reflejen el acondicionamiento de la temperatura cuando la misma se encuentra fuera de rango. Estos podrían ser

Inicialmente se propuso la inclusión de dispositivos que reflejen el acondicionamiento de la temperatura cuando la misma se encuentra fuera de rango. Ejemplos:

* + - * Un ventilador para cuando la temperatura es mayor al límite superior indicado
      * Una resistencia para cuando la temperatura es menor al límite inferior indicado

Debido al tiempo que requeriría la implementación de dispositivos como un ventilador, se decidió hacer uso del LED interno del Arduino UNO para responder a comandos enviados desde un cliente conectado al servidor MQTT. De está forma se puede realizar una comunicación desde un cliente en cualquier lugar hacia el sistema.

### 1.3.2. Adición de módulo RTC

Dado que se debe registrar la fecha y la hora de las mediciones, resultaría más preciso que el mismo módulo Arduino haga estas marcas temporales, puesto que las comunicaciones (y más aún las inalámbricas) conllevan un retardo considerable. Para este propósito se incluiría un módulo RTC (*Real-Time Clock*) y se adosaría esta marca temporal al paquete de datos transmitido al bróker MQTT. Estos módulos generalmente llevan una pila para mantener el circuito de reloj activo aun cuando se corta la alimentación del circuito principal y suelen tener una vida útil prolongada.

No se implementó está mejora debido a que no se obtuvo un RTC a tiempo para implementar la funcionalidad.

### 1.3.3. Adición de módulo de alimentación por batería

Como una posible mejora se propone "independizar" el sistema en funcionamiento de una computadora o incluso de la red eléctrica agregando una batería. Podría agregarse al software una verificación del nivel de carga y que se envíe una notificación por el mismo canal que las mediciones a los suscriptores informando que la batería de alimentación debe ser recargada o reemplazada.  
No se implementó está mejora debido a que no se consideró necesaria para este prototipo.

# 

# **2.- Presentación Esquemática del Proyecto**

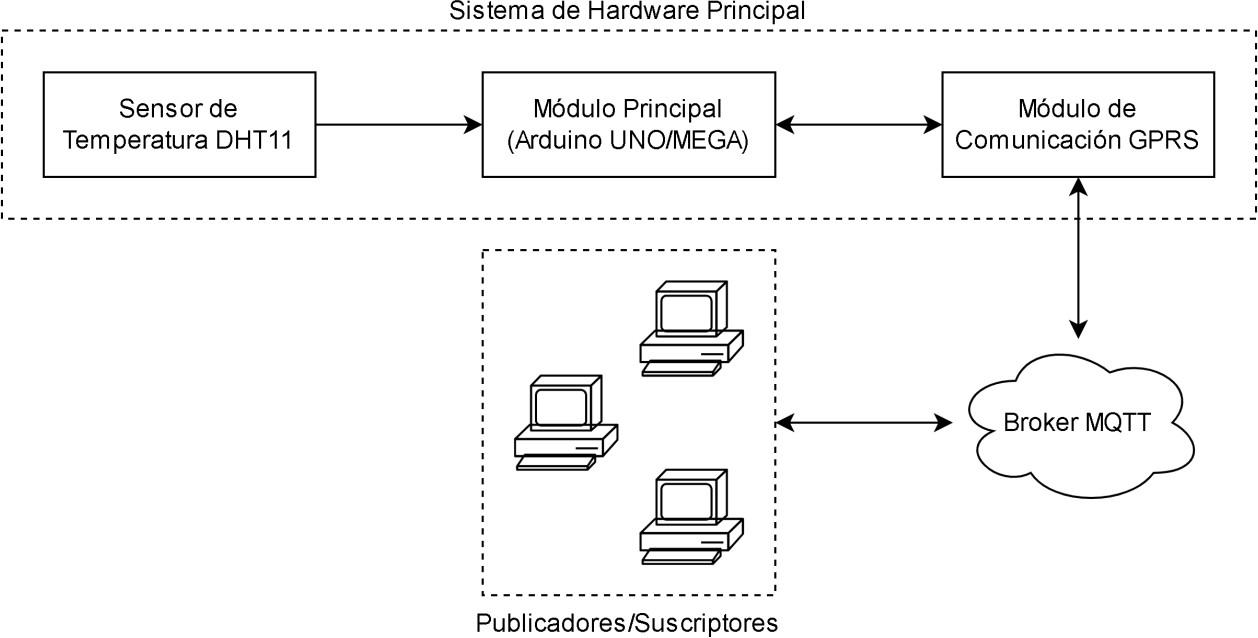
**Actualización completa de proyecto tal como se entrega, en dos esquemas gráficos:**

**1.- Gráfico de procesos y funciones del proyecto, identificando cuáles fueron implementadas por el grupo de desarrollo y cuáles son utilizadas de otras fuentes. Consultar con cada docente el nivel de detalle requerido. Dependiendo del docente y del proyecto en particular, pueden requerirse gráficos de casos de uso relacionándolo con el gráfico de procesos y funciones del proyecto.**

**2.- Esquemático de conexiones del hardware utilizado para construir el sistema. Todos los módulos/”breakouts” utilizados deben identificarse de manera explícita en el gráfico.**

## ***2.1. Gráfico de procesos y funciones del proyecto.***

El siguiente diagrama de bloques ilustra lo descrito anteriormente en términos generales.



**Figura 1.** Diagrama de bloques del sistema

De acuerdo a la figura 1, la placa Arduino recibiría los datos de temperatura del módulo de sensor DHT11, para luego empaquetarlos y transmitirlos a través de la red celular usando el módulo GPRS hacia un servidor MQTT. A éste se conectarán usuarios suscriptores que acceden a los datos del sensor y los almacenarán localmente, quienes a su vez podrían comunicarse en sentido inverso con la placa Arduino para establecer el rango de temperatura del sistema. Un aspecto omitido del diagrama anterior es la alimentación del sistema.

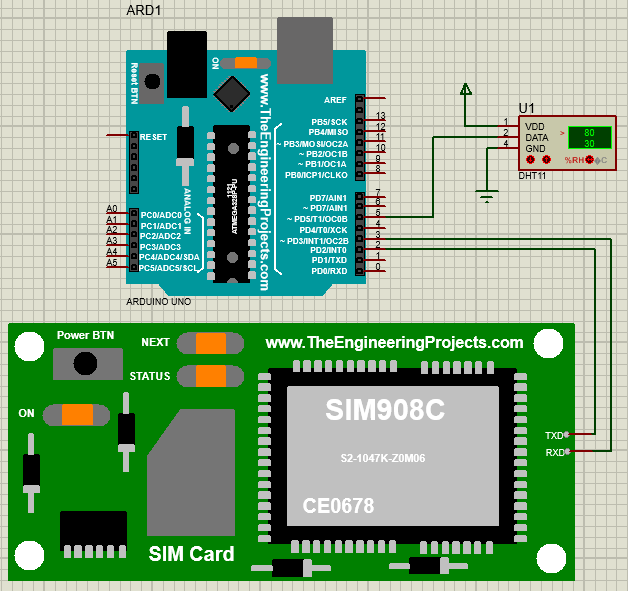
## ***2.2. Esquemático de conexiones de hardware***

Se realizaron las conexiones de todos los componentes electrónicos que conforman el sistema según lo especificado en la siguiente tabla:

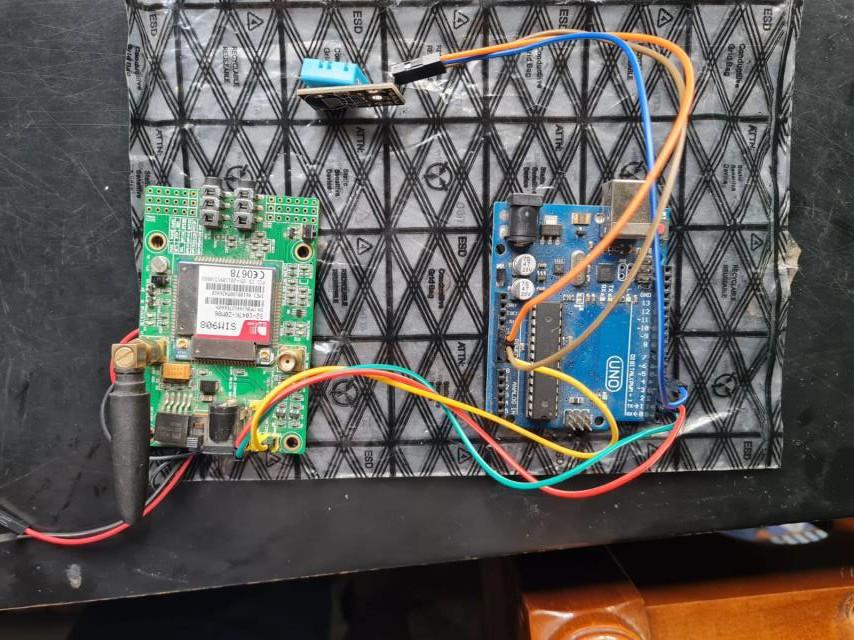
| Componente | Pin Componente | Pin Arduino |
| --- | --- | --- |
| DHT11 | VCC | 5V |
| GND | GND |
| Data | 5 |
| SIM908C | VCC | - |
| GND | GND |
| GPRSTXD | 2 |
| GPRSRXD | 3 |

**Tabla 1.** Conexiones entre la placa Arduino UNO y los demás componentes

La alimentación del SIM908C se realiza mediante una fuente de 9V/2A. A continuación, la figura 2 muestra el esquema de conexiones mientras que la figura 3 presenta el conexionado físico del sistema.



**Figura 2**. Diagrama de conexiones del sistema.



**Figura 3**. Esquema de conexiones físicas del sistema.

En la figura 3 podemos observar los 3 componentes: a la izquierda inferior se encuentra el SIM908C, a su derecha el Arduino UNO y en la parte superior el sensor de temperatura DHT11.

# **3.- Documentación del Software del Proyecto**

**Junto con este informe deben entregar todo el código fuente y configuraciones implementadas durante el desarrollo del proyecto. En este informe se debe indicar para cada carpeta/directorio y archivo/fichero su contenido y explicación de lo implementado y/o configurado en el mismo, no incluir código fuente en este informe a menos que se considere estrictamente necesario (corroborar con el docente a cargo). Se sugiere incluir en el código los comentarios necesarios para asociar la descripción hecha en esta sección en el código fuente propiamente dicho. En el caso de archivos de código fuente (.c, .cpp, .py, etc.), se deben listar las funciones y el objetivo de cada una de ellas.**

**Para los proyectos desarrollados con hardware Arduino-NodeMCU-ESP8266-ESP32:**

**- Identificar la instalación de manejadores de placa/s instaladas en el IDE de Arduino.**

**- Identificar la versión de manejador de placa utilizada en el IDE de Arduino y la configuración de nombre de placa con la cual se genera y carga el binario en la placa desde el IDE de Arduino.**

**- Identificar todas las bibliotecas/librerías que tuvieron que ser instaladas, que deben ser entregadas junto con el código fuente, dado que en un paso posterior otro grupo deberá reconstruir el proyecto para su continuación/mejora/etc. Además, aclarar si se pudieron utilizar de manera directa como parte del proyecto o tuvieron que ser instaladas en el IDE de Arduino.**

**A menos que se haya acordado con el docente a cargo o el proyecto determine otra configuración, todos los proyectos deben tener configuradas sus placas de WiFi como Access Point de la red WiFi.**

**Detallar al menos un caso de uso o un diagrama de secuencia utilizando como punto de partida una función del proyecto de las que se describieron en la Sección 2, de manera tal que se pueda asociar la funcionalidad del proyecto con el software y hardware involucrados. En caso de duda, consultar y acordar con el docente a cargo del proyecto.**

Los siguientes componentes forman el sistema de software:

### 3.3.1. Firmware del Sistema

Para la programación del firmware se utilizaron las siguientes herramientas, las cuales están adjuntas en este informe tanto el código fuente con sus respectivas librerías en la carpeta “arduinoClient”:

* Arduino IDE
* Lenguaje C
* Librerías de Arduino para el manejo de los componentes conectados:
  + TinyGsm: Libreria para el manejo de dispositivos GSM como el SIM908C.
  + PubSubClient: Librería para soporte de comunicación MQTT.
  + DHT: Librería para el manejo del sensor de temperatura DHT11
  + SoftwareSerial: Librería para habilitar la comunicación serial en pines digitales además de los designados por defecto por Arduino (pin 0 y 1 para Arduino UNO)
  + ArduinoJSON: Para el envío de datos en formato JSON.

A continuación, se describe en forma de pseudocódigo el firmware del sistema, tomando como referencia el estilo de programación usado en Arduino IDE:

void setup () {

Inicializar la comunicación serie entre Arduino y módulo GPRS

Configurar el pin correspondiente al LED interno de Arduino como salida

Inicializar el sensor de temperatura DHT11

Inicializar el módulo GPRS Conectar a la red telefónica Conectar a la red GPRS

Configurar los datos del servidor MQTT

Configurar la función a llamar en caso de recibir mensajes por MQTT.

}

void loop () {

Loop para reconectarse en caso de desconectarse de la red telefónica y/o GPRS

Si no está conectado al servidor MQTT, conectarse al servidor.

Caso contrario, si se cumplio 10 segundos desde la anterior comunicación/inicio de programa:

Leer los datos del sensor

Si no ocurrieron errores al obtener los datos del sensor: Se crea un objeto JSON y se publica como mensaje MQTT. Caso contrario, se transmite un mensaje informando del error

Llama un método loop para que no se corte la conexión al servidor.

}

El firmware utiliza dos canales para la comunicación MQTT:

* Arduino/temp: En este canal el firmware publica las medidas de temperatura.
* Arduino/control: El firmware está suscrito en este canal para recibir mensajes de control por parte de un usuario.

Cuando se recibe un mensaje por el canal de control, el firmware procesa el mensaje según el pseudocódigo a continuación, por medio de la función mqttCallback:

void mqttCallback () {

Imprime el mensaje recibido en el terminal de Arduino.

Alterna el estado del led para simular la parte de control.

}

### Documentación del Firmware

El código desarrollado para la placa Arduino está implementado en entorno Arduino IDE diferentes librerías disponibles para la gestión de cada uno de los módulos integrados en el archivo “arduinoClient.ino”. En este archivo se encuentran las siguientes variables de configuración:

* GSM\_PIN: Las tarjetas SIM tienen una clave PIN asignada que puede usarse para desbloquear la tarjeta en caso de bloquearse.
* apn[]: APN de la red telefónica.
* gprsUser[]: Usuario para conectarse a la APN.
* gprsPass[]: Contraseña para conectarse a la APN.
* broker: Dirección del servidor MQTT. Puede ser tanto la IP como una dirección URL.
* topicTemp: Nombre del canal donde el firmware enviará los censados de temperatura.
* DHTPIN: Variable de configuración de la librería DHT para identificar a qué pin está conectado el pin DATA
* DHTTYPE: Variable de configuración de la librería DHT para identificar el modelo de sensor de temperatura.

El equipo desarrolló las siguientes funciones:

mqttConnect(): se conecta al servidor MQTT especificado en la variable broker por con las credenciales especificadas para este proyecto.

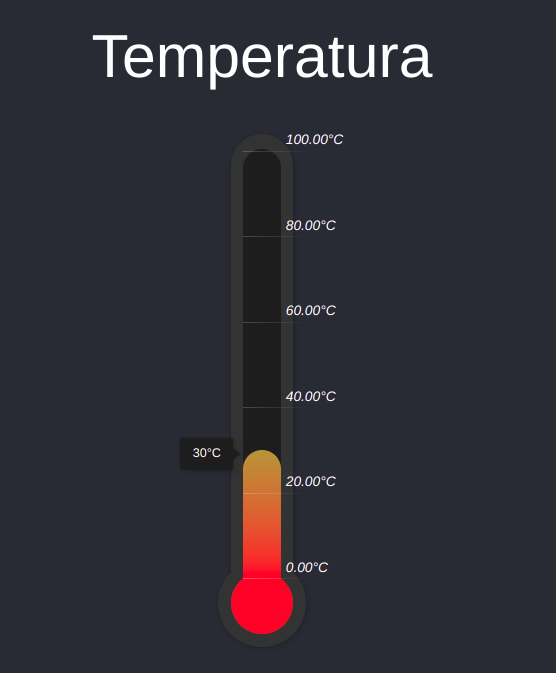
mqttCallback(char\* topic, byte\* payload, unsigned int len): Maneja los mensajes MQTT recibidos por la placa Arduino.

## ***3.4.1 Aplicación Web temperatura***

Para poder visualizar el valor actual de temperatura se optó por desarrollar una aplicación React ignorando la idea principal de hacerlo con grafana, se tomó esta decisión debido a la complejidad de Grafana y a la facilidad que ofrece React para implementar la solución.

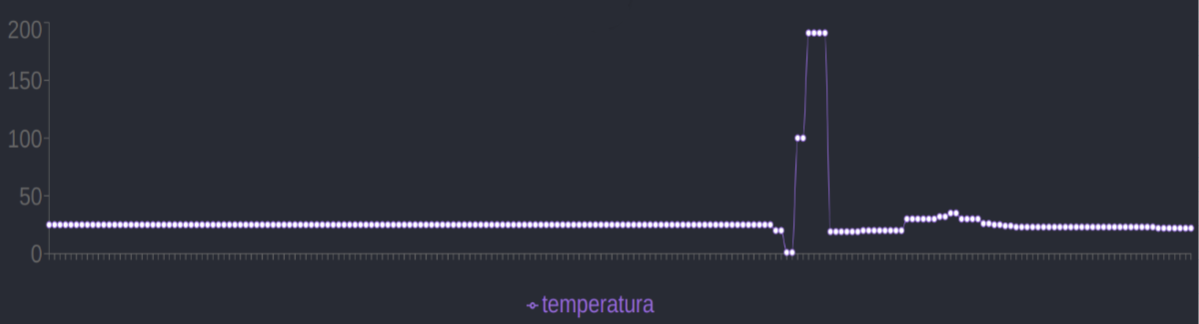
Nos basamos en un ejemplo de la comunidad que utiliza la librería mqtt desde el front para poder suscribirse al tópico “**Arduino/temp**”, luego para mostrar la información recibida desde el broker utilizamos la librería “**react-thermometer-component**“

Se realizaron pruebas en tiempo real, enviando mensajes al tópico mencionado y se puede comprobar que el refresco del valor del termómetro se realiza con éxito.



**Figura 8.** Aplicación Web en donde muestra la temperatura del arduino en tiempo real.

Se realizó la adición de un gráfico de datos históricos de temperatura (ver Figura 9). Este gráfico se implementó utilizando la biblioteca "Recharts", reconocida por su facilidad de uso y versatilidad para la representación de distintos tipos de datos gráficos. En particular, optamos por el componente "LineChart" de Recharts para visualizar la evolución de la temperatura a lo largo del tiempo. La información necesaria para alimentar este gráfico se obtiene directamente desde un endpoint específico en nuestro backend, diseñado para proveer estos datos históricos en un formato adecuado para su visualización. Esta integración asegura que el gráfico siempre refleje los datos más recientes y completos disponibles.



**Figura 9.** Gráfico XY de temperatura

# ***3.4.2 Pseudocodigo***

1. **Establecer Conexión con el Broker MQTT:**

Para poder conectar el broker MQTT primero se obtienen las credenciales necesarias para establecer la conexión como: protocolo, host, clientId, puerto, usuario y contraseña. Luego se procede a conectar al broker MQTT y actualizar el estado de conexión.

Definir configuración()

Obtener protocolo, host, clientId, etc.

1. **Manejo de Eventos de Cliente MQTT:**

Cada vez que se produce un evento en el objeto cliente, se ejecuta una acción:

**Establecer conexión()**

Siempre que se produzca un cambio en el objeto “cliente”():

-Si se realizó la conexión de forma exitosa:

Actualizar estado de conexión a 'Connected' y registrar en consola.

-Si se produjo un error:

Manejar y registrar errores de conexión.

-Si se está reconectando:

Actualizar estado de conexión a 'Reconnecting'.

-Si se recibe un mensaje:

Actualizar el estado payload con el mensaje recibido y registrar en consola.

1. **Suscribirse a Tópicos MQTT:**

Utiliza el cliente mqtt para suscribirse a un tópico, actualizar estado de suscripción una vez suscrito al tópico.

1. **Renderizar Componente:**

Renderizar el componente Receiver pasando el mensaje transformado en un json como parámetro, para así poder mostrar el componente.

***3.4.3 Guia de Uso***

* **Prerrequisitos**:
  + [Node LTS](https://nodejs.org/) version v14.21.3 (recomendado usar nvm)
* **Dependencias**:
  + npm install (para instalar las dependencias)
* **Ejecutar el proyecto**
  + Levantar primero el backend.
  + Ejecutar el comando en el directorio raíz del proyecto
    - **npm run start**

# ***3.5.1 Aplicación Backend para persistencia de datos en la DB***

El backend desarrollado en Python se usa para la gestión y procesamiento de datos de temperatura. Realiza las siguientes operaciones:

1. **Recepción y Manejo de Datos de Temperatura:**

Para poder recibir la información del broker se implementan callbacks específicos que se activan al conectarse al broker MQTT y al recibir mensajes. En el primer caso indica en un archivo de log que la conexión fue exitosa, en el caso de recibir un mensaje también se hace un log de el mensaje recibido y también se persiste en la base de datos el dato obtenido.

**Recibo Mensaje del Broker**():

insertar en la base de datos

1. **Almacenamiento y Gestión de Datos:**

Para persistir y recuperar datos de temperatura se una conexión con una base de datos MySQL

**Insertar Temperatura()**

Obtiene las credenciales de la DB

Establecer conexión con la DB

Insertar información recibida en la DB.

**Si Hay Error:**

Registrar error

**Por Último:**

Cerrar conexión

1. **Exposición de Datos a través de un API REST:**

Se usa la librería de Python Flask (un micro framework de Python) para exponer un API REST.

Haciendo uso de esta librería se expone el endpoint “/temperatura”que permite recuperar los datos históricos de temperatura almacenados, retornando en formato JSON para la aplicaciones cliente.

**Obtener Temperatura()**

Obtiene las credenciales de la DB

Establecer conexión con la DB

Obtiene información de la DB.

Revuelve el resultado en formato JSON.

**Si Hay Error:**

Registrar error

**Por Último:**

Cerrar conexión

1. **Registro y Monitoreo:**

Se usa un archivo de logging para llevar un registro detallado de todas las operaciones, incluyendo conexiones, inserciones de datos y errores.

**Aclaración:** La credenciales se encuentran directamente en el código, idealmente se obtendrían en tiempo de ejecución utilizando variables de entorno.

# ***3.5.2* Guia de uso:**

#### Prerrequisitos:

* Python, en el presente proyecto se utiliza la versión 3.8.

#### Dependencias:

* Para instalar las librerías de python se utiliza pip:

***pip install paho-mqtt***

***pip install mysql-connector-python***

***pip install Flask***

***pip install Flask-CORS***

#### Ejecutar el proyecto:

* Para poder levantar el proyecto se utiliza el siguiente comando en una consola de linux:

**python backend.py**

* Debería devolver el siguiente mensaje:

\* Serving Flask app 'backend'

# **4.- Otra Documentación Relacionada**

**Incluir fotos de las partes físicas ya construidas del sistema. Incluir capturas de pantalla de la/s interfaz/ces web que se haya/n construido. En todos los casos, indicar a qué parte/s corresponden de las que se describen en las secciones anteriores.**

**Crear un video de no más de 2 minutos (se sugiere 1 minuto o menos) con al menos la funcionalidad explicada en la sección anterior. Este video no reemplaza nada de lo anterior ni le agrega información a lo anterior, sino que lo complementa. Incluir en esta sección el enlace al video en algún sistema de almacenamiento de videos (youtube, vimeo, etc.) o archivos (dropbox, Mega, etc.) desde donde se pueda ver y/o descargar, asegurándose que no sea necesario ningún tipo de información o tarea extra como clave, registro (login) en el sistema, etc. Se recomienda que el formato sea .mp4 en el caso de usar almacenamiento de archivos como Dropbox o Mega.**

**Agregar en esta sección el enlace a la bitácora, donde han documentado los avances/experimentos/etc. de manera cronológica y al sistema de almacenamiento y versionado de código fuente desde donde se pueda obtener todo el código del proyecto.**

**Agregar en esta sección el enlace al sistema de almacenamiento/repositorio de código utilizado.**

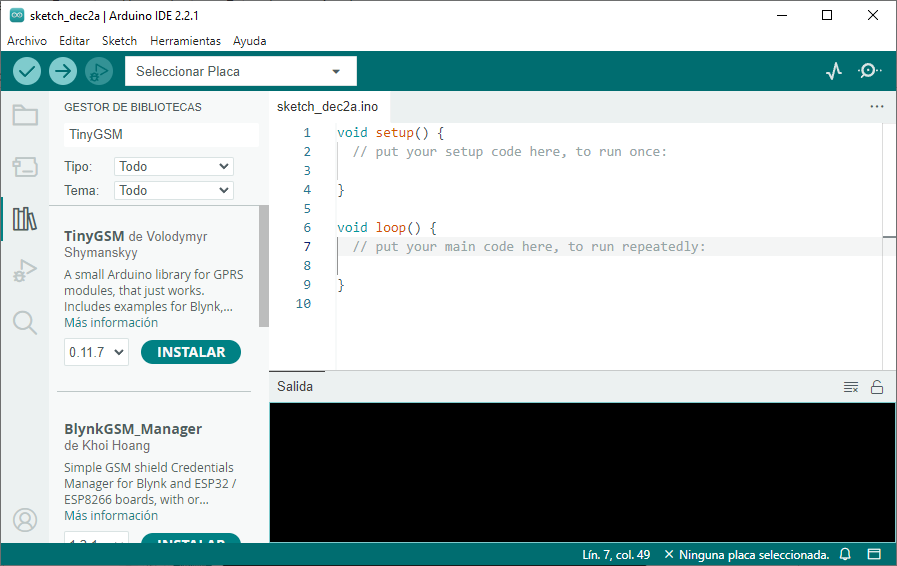
### 

### Guia de instalacion: Entorno Arduino

El firmware fue desarrollado y compilado en el entorno Arduino disponible para su descarga en: https://www.arduino.cc/en/main/software

Una vez descargado el IDE de Arduino, se deben descargar las librerías utilizadas para gestionar los módulos. Para ello, en el menú “Herramientas” seleccionar la opción “Gestionar bibliotecas…” y en dicha ventana buscar e instalar cada una de las librerías listadas a continuación. Las librerías que no son descargables desde el IDE, se agrega el link de descarga:

* TinyGsm
* PubSubClient
* DHT sensor library
* ArduinoJSON



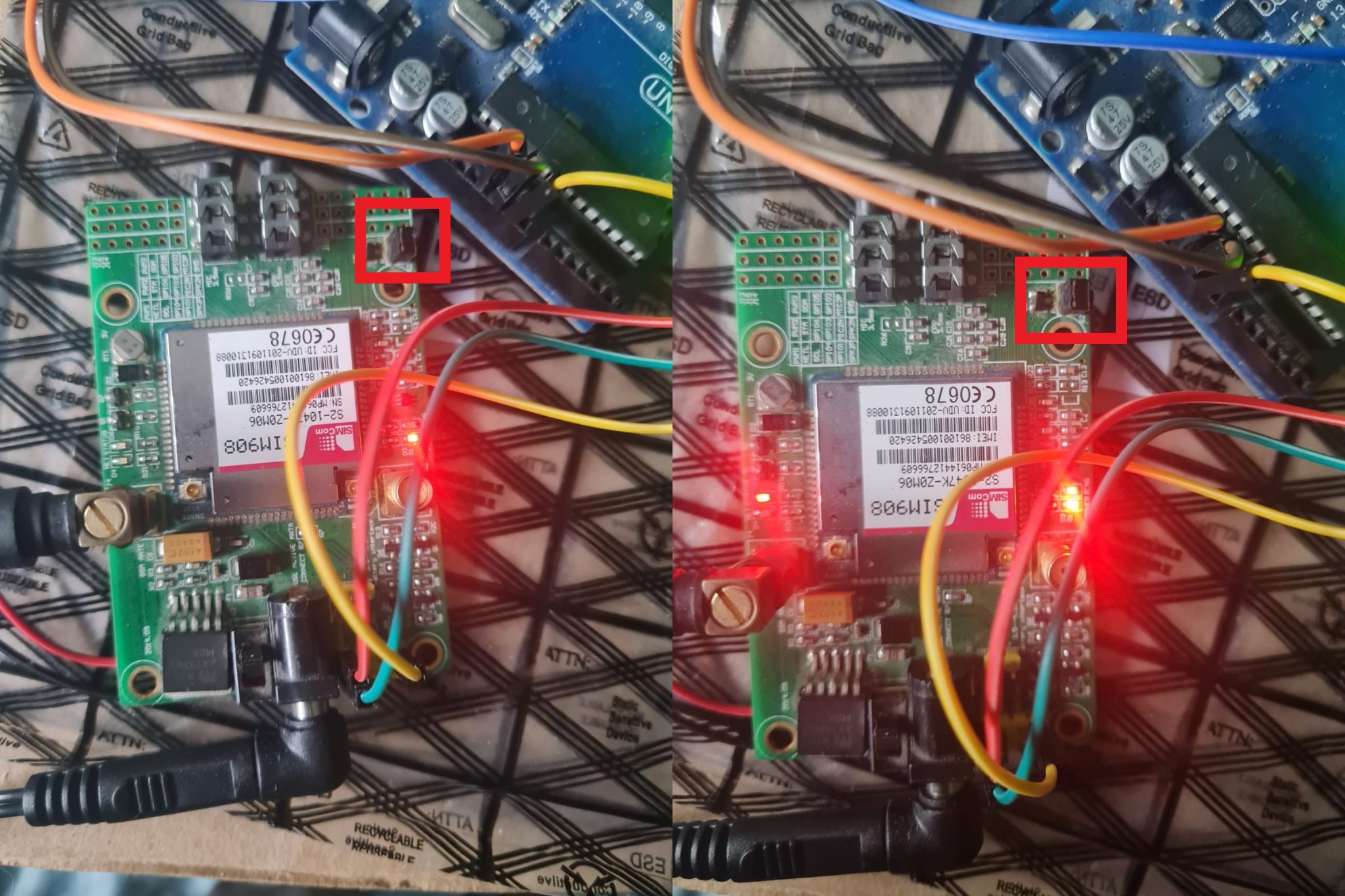
**Figura 4.** Ejemplo de instalación de librerías en Arduino

# En la figura 4 se muestra a modo ejemplo la búsqueda de la librería “TinyGSM”, para instalar la librería hay que presionar el botón “Instalar” que permitirá agregarla al entorno. Una vez instaladas las librerías, para poder compilar el código se debe completar los datos de la configuración como la dirección del broker MQTT y los datos de la APN de la tarjeta SIM colocada en el SIM908C y presionar el botón “Cargar” en el Arduino IDE.

### Guia de uso: SIM908C

En caso de desear probar el funcionamiento del módulo SIM908C, la libreria TinyGSM ofrece ejemplos que puede probar accediendo al menú “Archivo”, seleccionar “Ejemplos”, luego en “Ejemplos de bibliotecas personalizadas” seleccionar “TinyGSM”. Cabe destacar que estos ejemplos pueden ser complejos para alguien sin experiencia con el uso de placas GSM por lo que una alternativa para comprobar el funcionamiento es utilizar un código de pruebas desarrollado por el grupo llamado “test-sim800l.ino” que realiza pruebas utilizando comandos AT utilizados frecuentemente. Simplemente hay que descargar el archivo .ino y cargarlo a la placa utilizando el botón “Cargar”.

Cada vez que se enciende la placa, es necesario presionar un botón por un segundo o dos para iniciarla por completo, de otra forma no se conectará a la red móvil.



**Figura 5.** Estado de la placa al realizar el encendido correctamente.

En la figura anterior se observa la placa antes y después de presionar dicho botón. La parte izquierda de la imagen representa la placa al antes de presionar el botón, que está marcado con un cuadrado rojo. A la derecha, después de presionarlo, se puede observar que ahora hay más leds encendidos.

Siguiendo los mismos pasos anteriores, pero ejecutando el DHTtester en el Arduino IDE uno puede probar el funcionamiento del sensor de temperatura DHT11.

# **Apéndice A: Materiales y Presupuesto**

**Proveer la lista completa y actualizada (con los materiales ya acordados con la cátedra) de elementos de hardware necesarios para construir el proyecto, teniendo en cuenta los costos aproximados de cada uno de estos elementos. No incluir computadora de escritorio, computadora portátil o dispositivo móvil para acceso/uso de lo proyectado. Si el proyecto requiere servicios de la nube indicar el costo aproximado y si requiere suscripción anual o “por demanda/utilización de recursos”. Los valores de costos deben ser actuales (fechas dentro del período de desarrollo del proyecto) y las referencias respecto de dónde (enlace web) se obtuvieron los valores.**

Los siguientes materiales componen el sistema de hardware esencial para el funcionamiento del sistema. La tabla incluye los precios de cada componente:

| ***Componente*** | ***Precio*** |
| --- | --- |
| Placa Arduino UNO | $7600 |
| Módulo Sensor DHT11 | $1800 |
| Módulo GPRS SIM908C | $43989 |
| Chip Tuenti + Recarga | $1200 |
| Fuente de alimentacion 9V/2A | $5600 |
|  | |
| Costo total | $60189 |

**Tabla 1.** Presupuesto de los materiales obtenidos (Arduino UNO, DHT11, GPRS SIM 908C: mercadolibre, octubre 2023).

Componentes adicionales de hardware omitidos del listado anterior incluyen el equipo informático sobre el que funcionará el servicio MQTT, los equipos desde los cuales se conectarán los suscriptores y las fuentes de alimentación de cada uno de los componentes de hardware utilizados, especialmente la fuente externa para el módulo de comunicación GPRS.

**Tiempo y Forma de Entrega:**

**- Informe en formato .pdf enviado por correo a todos los docentes e integrantes del grupo.**

**- En ningún caso debe hacerse referencia al contenido de la bitácora (excepto algo explícito acordado con el docente a cargo).**

**- Tener en cuenta que los enlaces al código fuente, bitácora y video deben estar en el informe (en donde corresponda).**