**Taller de Proyecto II**

**Informe de Avance I**

**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

Contenido

[1. Introducción 3](#_Toc147492683)

[2. Objetivos 4](#_Toc147492684)

[2.1. Objetivos Principales 4](#_Toc147492685)

[2.2. Objetivos Secundarios 5](#_Toc147492686)

[2.2.1. Adición de módulos controlados por el sistema 5](#_Toc147492687)

[2.2.2. Adición de módulo RTC 5](#_Toc147492688)

[2.2.3. Adición de módulo de alimentación por batería 6](#_Toc147492689)

[3. Identificación de Partes 7](#_Toc147492690)

[3.1. Componentes de Hardware y Presupuesto 7](#_Toc147492691)

[3.2. Componentes de Software 7](#_Toc147492692)

[3.2.1. Firmware del Sistema 7](#_Toc147492693)

[3.2.2. Servidor MQTT (Interfaz EMQX) 7](#_Toc147492694)

[3.2.3. Comunicación entre Arduino y Servidor MQTT 8](#_Toc147492695)

[3.2.4. Aplicaciones y Procesamiento de Datos 8](#_Toc147492696)

[4. Grado de Avance 9](#_Toc147492697)

[4.1. Hardware y Firmware (TODO LO QUE ES ARDUINO) 9](#_Toc147492698)

[4.2. Servidor e Interfaz Web (CONFIG SERVER + GIT + DOCKER…) 9](#_Toc147492699)

[4.2.1. Servidor 9](#_Toc147492700)

[4.2.2. Docker 10](#_Toc147492701)

[4.2. Comunicaciones con la PC o dispositivo móvil 10](#_Toc147492702)

[4.4. Software para PC o dispositivo móvil 10](#_Toc147492703)

[4.5. Problemas más importantes y soluciones 10](#_Toc147492704)

[5. Documentación Relacionada 11](#_Toc147492705)

# 1. 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.

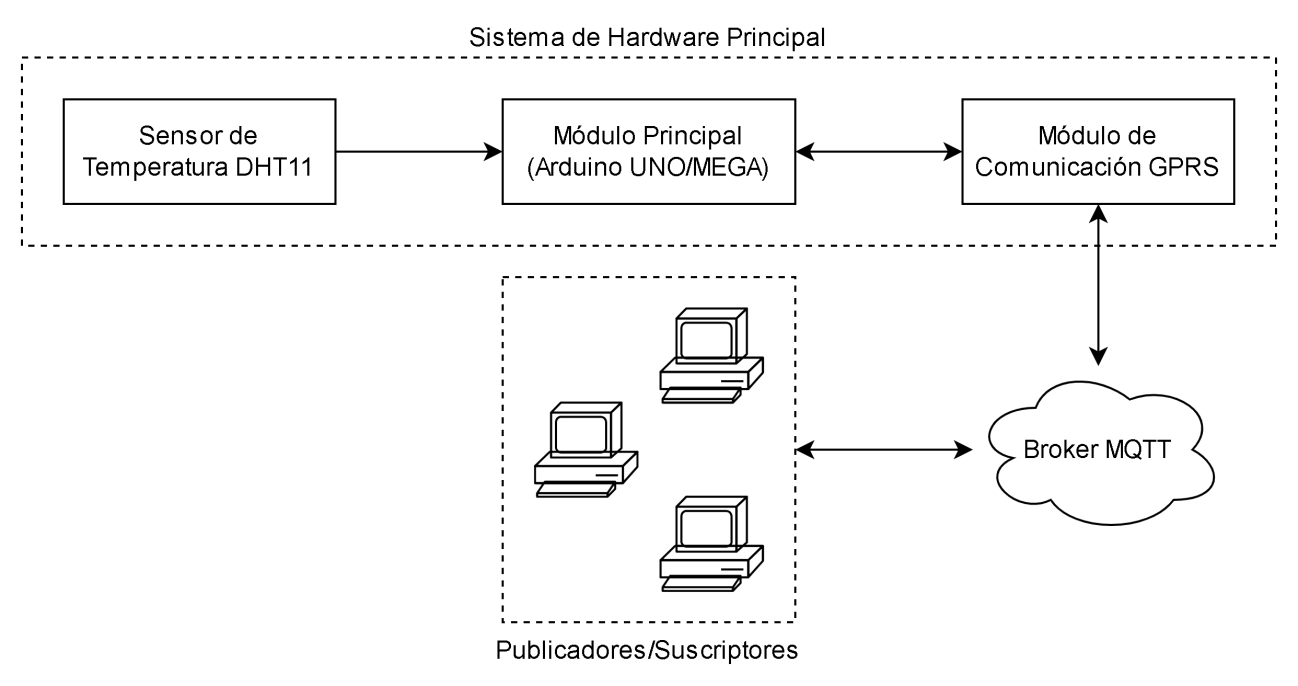
# 2. Objetivos

## 2.1. Objetivos Principales

El objetivo principal del proyecto es realizar 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.

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 conectarían usuarios suscriptores que accederían a los datos del sensor y los almacenarían 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. Como objetivo principal se propone alimentar el módulo GPRS utilizando una fuente separada de los demás componentes debido a que éste presenta un consumo de corriente pico de 2A durante la comunicación, lo que podría no ser soportado por una única fuente de alimentación. Los demás componentes (placa Arduino y sensor DHT11) serían alimentados por una misma fuente de alimentación, la cuál podría ser un puerto USB de una PC.

## 2.2. Objetivos Secundarios

A continuación se incluye un listado de posibles mejoras o ampliaciones realizables al sistema una vez cumplidos los objetivos principales.

### 2.2.1. 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:

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

De no ser posible la adición de dichos periféricos, el sistema incluiría al menos un indicador LED que se encendería sólo cuando la temperatura se encuentra fuera del rango seleccionado.

### 2.2.2. 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.

### 2.2.3. 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.

# 3. Identificación de Partes

## 3.1. Componentes de Hardware y Presupuesto

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 |
|  |  |
| Costo total | $53383 |

**Tabla 1.** Presupuesto de los materiales obtenidos

(Arduino UNO, DHT11, GPRS SIM 908C: mercadolibre, 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.

## 3.2. Componentes de Software

Los siguientes componentes forman el sistema de software:

### 3.2.1. Firmware del Sistema

Se utilizará para el control del sistema, monitoreo del sensor y comunicación con el bróker MQTT. Será programado en lenguaje C y será ejecutado directamente sobre la placa de desarrollo Arduino. El firmware abarca los siguientes procesos:

* Control del sistema: El firmware controla las acciones y operaciones de la placa Arduino, incluyendo la adquisición de datos de sensores y la toma de decisiones.
* Monitoreo del sensor: Implementa rutinas para leer datos de los sensores conectados y garantizar que los datos sean precisos y consistentes.
* Comunicación MQTT: Establece una comunicación bidireccional con un bróker MQTT para enviar y recibir datos de manera eficiente.

### 3.2.2. Servidor MQTT (Interfaz EMQX)

El servidor MQTT actúa como un componente central en la infraestructura del proyecto, y se compone de los siguientes procesos de software:

* Bróker MQTT: La interfaz EMQX actúa como un bróker MQTT que facilita la publicación y suscripción de datos entre los dispositivos IoT y otros sistemas.
* Gestión de temas: Gestiona la organización de datos en temas específicos para permitir una transmisión eficiente y una fácil suscripción por parte de los clientes.

### 3.2.3. Comunicación entre Arduino y Servidor MQTT

* Establecimiento de Conexión: El firmware en la placa Arduino inicia y mantiene una conexión segura con el servidor MQTT a través de la red.
* Publicación de Datos: Los datos recopilados por los sensores se publican en temas MQTT para su posterior procesamiento.
* Suscripción de Datos: Si es necesario, la placa Arduino puede suscribirse a temas MQTT específicos para recibir comandos o actualizaciones desde el servidor.

### 3.2.4. Aplicaciones y Procesamiento de Datos

* Visualización y Análisis: Las aplicaciones de visualización y análisis pueden consumir los datos desde el servidor MQTT para proporcionar información en tiempo real y análisis históricos.
* Alertas y Notificaciones: Las aplicaciones pueden generar alertas y notificaciones basadas en los datos recibidos, mejorando la capacidad de respuesta.
* Almacenamiento de Datos: Si es necesario, los datos también pueden almacenarse en bases de datos para su posterior referencia y análisis.

# 4. Grado de Avance

## 4.1. Hardware y Firmware (TODO LO QUE ES ARDUINO)

Al principio del desarrollo, se asumió que se trabajaría con el módulo SIM800L ya que era uno de los componentes más usados en proyectos relacionados al uso de GPRS con Arduino por lo que se consiguió un SIM800L para la realización de pruebas de conexión y de uso de mensajes por GPRS utilizando Arduino además de realizar la primera versión del firmware. Al recibir por parte de la catedra el módulo SIM908C, se pasó a trabajar sobre este módulo.



**Figura 2**. Esquema de conexiones físicas del sistema.

La figura 2 representa el conexionado preliminar de todos los componentes electrónicos que conforman al sistema. En el centro se observa el Arduino UNO, conectado a su izquierda el módulo GPRS a través de los pines 2 y 3 a TX y RX respectivamente. Por otro lado, se observa la conexión al sensor DHT11 por los pines de alimentación de +5V y GND del Arduino; y al pin 5 como pin de datos.

Para el firmware, se utilizaron las siguientes librerías:

* DHT: Para el uso del sensor de temperatura.
* SoftwareSerial: 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 envio de datos en formato JSON.
* TinyGsmClient y PubSubClient: Para facilitar la comunicación por GPRS.

A continuación, se describe a modo de pseudocodigo el firmware del sistema en el formato que trabaja Arduino IDE:

|  |
| --- |
| void setup () {    Inicializar la comunicación serie entre Arduino y el módulo GPRS    Inicializar el sensor de temperatura DHT11    Inicializar el módulo GPRS    Conectar a la red GPRS    Configurar los datos del servidor MQTT  }  void loop () {    Loop para conectarse al servidor en caso de que no haya conexión entre el servidor y Arduino    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    En caso contrario, se transmite un mensaje informando del error  Esperar 5 segundos para tomar la próxima medida  } |

Para conectarse a la red GPRS, el firmware debe tener los datos necesarios del APN (Access Point Name) del proveedor del servicio, en este proyecto se utiliza Tuenti como proveedor y para la configuración de la conexión con el servidor MQTT se utiliza la dirección y puerto del servidor junto con las credenciales para acceder.

Lucas/Victor: Lo que habían hecho lo pegué aca, asi redactan algo y queda la imagen como prueba del avance.

## 4.2. Servidor e Interfaz Web (CONFIG SERVER + GIT + DOCKER…)

### 4.2.1. Servidor

Se configuró un equipo de escritorio como servidor para cumplir con las tareas de bróker MQTT y servidor de base de datos. Se instalaron los servicios EMQX y Mosquitto para comunicaciones MQTT, y MySQL para gestionar una base de datos relacional. Como medida de seguridad, se configuró un firewall (UFW), en el cual deben explicitarse los puertos de conexión de los servicios para permitir la conectividad.

Se realizaron pruebas de conexión satisfactorias mediante comandos de diagnóstico (ping) hacia el servidor y se habilitaron los puertos de conexión para todos los servicios y se establecieron conexiones y realizaron pruebas de funcionamiento mediante programas en lenguaje C y Python.

Los programas de prueba en C para el servicio Mosquitto (ubicados en el directorio /mosquitto/test del repositorio del proyecto) configuran la conexión al puerto 15883 del servidor, para luego realizar pruebas. Uno de los programas (test-mosquitto-sub) se suscribe a un tópico de prueba e imprime en la consola los mensajes recibidos hasta que se pierde la conexión. El otro programa (test-mosquitto) se conecta al mismo puerto del servidor y publica datos de prueba (en formato JSON) a un tópico de prueba cada un cierto tiempo, los cuales deben ser recibidos por el programa suscriptor. Ambos programas hacen uso de la librería mosquitto.h.

Se ejecutaron inicialmente ambos programas en dos terminales separadas (estaciones de trabajo Linux conectadas a Internet) y se comprobó el funcionamiento de la comunicación. La figura X muestra que los mensajes se envían desde una terminal y se reciben en la otra.

(FIGURA X)

Para la prueba del servicio EMQX se escribió un programa en Python (ubicado en el directorio /emqx del repositorio) que utiliza la librería paho.mqtt. Establece una conexión al puerto 1883 del servidor y envía mensajes a un tópico de prueba utilizando un formato de mensaje similar al usado en las pruebas de Mosquitto. La recepción se monitorea inicialmente desde el dashboard provisto EMQX y desde un suscriptor Websocket conectado al servidor. La figura Y muestra que la prueba es exitosa.

(FIGURA Y)

Sin embargo, al momento de rescribir el programa en Python en lenguaje C, la conexión ya no logra establecerse. Por esta razón se optó por emplear Mosquitto hasta que se solucione el problema con EMQX.

Para probar la funcionalidad del servidor de base de datos se creó una base de datos de prueba: una tabla con los campos fecha de medición y temperatura. Se escribió un programa en C que establecería una conexión y listaría en la consola las tablas que pertenecen a la base de datos. Al ejecutar el programa se establece la conexión al servidor, pero el acceso al sistema de base de datos es denegado por MySQL (error 111). Por el momento no se ha encontrado solución a este problema y se supone inicialmente que es un problema de privilegios de acceso al sistema por parte del usuario de prueba de la base de datos, sin embargo, tampoco se ha logrado la conexión con el usuario administrador.

### 4.2.2. Docker

Se desarrolló un docker-compose con un bróker Mosquitto y un programa de backend en Python para realizar pruebas mientras se terminaba de configurar el servidor. Dicho programa se suscribe a un tópico de prueba. Se le agregaron al contenedor los scripts necesarios para inicio y apagado.

Hernan: aca redactate algo de como armaste el docker

Todo lo que sobró del punteo anterior que quede como ayuda para armar el informe redactado, se borra antes de la entrega

## 4.2. Comunicaciones con la PC o dispositivo móvil

* Se realizó una prueba de conexión con el módulo SIM800L, incluyendo la conexión a la red celular, el envío de SMS y una petición GET a un endpoint dummy.
* Se desarrolló código para la conexión del módulo GPRS usando Arduino.

## 4.4. Software para PC o dispositivo móvil

* Se desarrolló código en Arduino para la integración de sensor DHT11, módulo GPRS, conexión a bróker MQTT y utilización de formato de mensaje JSON.

## 4.5. Problemas más importantes y soluciones

* Se tuvo que actualizar el proyecto de Arduino para adecuarse al módulo GPRS recibido (SIM908).

# 5. Documentación Relacionada

* El repositorio de GitHub del proyecto es accesible desde el siguiente enlace web. Es de acceso público y contiene tanto el código fuente de los módulos de hardware/sofwtare del proyecto como la bitácora e instrucciones de configuración para los servicios web en el servidor:

<https://github.com/tpII/2023-G2-MQTT-GRPS.git>

* La bitácora está almacenada dentro del repositorio de GitHub del proyecto, en un directorio del mismo nombre y en formato markdown para su correcta visualización en un navegador
* Los videos de las pruebas de funcionamiento están en formato (.mp4) se suben al repositorio del trabajo (directorio /docs/video)

Sitios web consultados para armado de presupuesto:

* Mercadolibre. Arduino Uno Ch340 C/cable Usb Compatible (2023). Disponible en internet: <https://articulo.mercadolibre.com.ar/MLA-925594160-arduino-uno-ch340-ccable-usb-compatible-_JM#position=1&search_layout=grid&type=item&tracking_id=ca13e077-19a1-4eae-bea0-c149b530f095> Consultado el 5/10/2023.
* Mercadolibre. Shield Arduino Gsm Gprs Gps Bluetooth Con Módulo Sim808 (2023). Disponible en internet: <https://articulo.mercadolibre.com.ar/MLA-627307217-shield-arduino-gsm-gprs-gps-bluetooth-con-modulo-sim808-_JM#position=5&search_layout=grid&type=item&tracking_id=436fc6ec-2ab3-49a4-a422-764096a882f7> Consultado el 5/10/2023.
* Mercadolibre. Sensor De Temperatura Y Humedad Relativa Dht11 Arduino (2023). Disponible en internet: <https://articulo.mercadolibre.com.ar/MLA-773164135-sensor-de-temperatura-y-humedad-relativa-dht11-arduino-_JM#position=1&search_layout=grid&type=item&tracking_id=fbf3a1ac-242c-4ef7-8b03-582d76927350> Consultado el 5/10/2023.