|  |
| --- |
|  |
| Proyecto final |
| Industria farmacéutica |
|  |
| **Víctor Manuel Navarro Pérez** |
| **Cristina Fernández García** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EDICIÓN | FECHA | REALIZADO | REVISADO | REVISADO |
| 0.0 | 07/05/2023 | Cristina Fernández |  |  |
| 0.1 | 28/05/2023 | Víctor Navarro |  |  |

**ÍNDICE**

[1. Introducción 3](#_Toc136277234)

[2. Acrónimos y abreviaturas 3](#_Toc136277235)

[3. Empresa 4](#_Toc136277236)

[4. Material empleado 5](#_Toc136277237)

[5. Roles y funciones 6](#_Toc136277238)

[6. Tamaño de almacenamiento del proyecto 7](#_Toc136277239)

[7. Sensores 7](#_Toc136277240)

[8. Modelo Hardware 10](#_Toc136277241)

[9. Modelo Software 12](#_Toc136277242)

[10. Base de datos 14](#_Toc136277243)

[11. Comunicación Arduino – Raspberry Pi 15](#_Toc136277244)

[12. Interfaces para la visualización de datos 17](#_Toc136277245)

[12.1. Thingsboard 17](#_Toc136277246)

[12.2. Google Chart 20](#_Toc136277247)

[12.3. Arduino IoT Cloud 23](#_Toc136277248)

[13. Avisos y alarmas 27](#_Toc136277257)

[14. GitHub 30](#_Toc136277258)

[15. Mejoras futuras 31](#_Toc136277259)

[16. Conclusiones 31](#_Toc136277260)

[1. Bibliografía 32](#_Toc136277261)

[2. Anexos 33](#_Toc136277262)

[2.1. Anexo 1: Código Arduino IoT 33](#_Toc136277263)

[2.2. Anexo 2: Código Arduino LoRa 35](#_Toc136277264)

[2.3. Anexo 3: Código Arduino Puerto Serie 38](#_Toc136277265)

[2.4. Anexo 4: 1\_iot\_read.py 39](#_Toc136277266)

[2.5. Anexo 5: 2\_iot\_cad.py 40](#_Toc136277267)

[2.6. Anexo 6: 3\_iot\_bbdd.py 41](#_Toc136277268)

[2.7. Anexo 7: 6\_iot\_thingsboard.py 42](#_Toc136277269)

[2.8. Anexo 8: googlechart.php 45](#_Toc136277270)

# Introducción

El proyecto IoT consiste en llevar a cabo la interconexión de dispositivos a través de internet. Esta conexión permite que los dispositivos generen y compartan datos en tiempo real.

Una plataforma de IoT es un conjunto de tecnologías y herramientas que permiten la creación, gestión y análisis de soluciones de IoT (Internet of Things).

Al emplearse una plataforma de IoT, las empresas pueden recopilar datos en tiempo real de dispositivos conectados para monitorizar y optimizar operaciones.

En este proyecto se hace uso de una placa Arduino Mega, una Raspberry Pi, una placa Arduino IoT y un dispositivo LoRa (emisor/transmisor y receptor). La obtención de datos en este se llevará a cabo a partir de diversos sensores, que son detallados en el apartado 4, para a continuación almacenarlos y explotarlos con el fin de poder compartir dichos datos por internet.

Imagen que contiene Diagrama

Descripción generada automáticamente

*Figura 1. Etapas del proceso de monitorización y control de datos.*

# Acrónimos y abreviaturas

|  |  |
| --- | --- |
| BBDD | Base de datos |
| RPI | Raspberry Pi |
| SO | Sistema Operativo |
| PHP | Hypertext Preprocessor |
| SSH | Secure Shell |
| IoT | Internet de las Cosas |
| LoRa | Long Range |

# Empresa

“Farma Industries” es un grupo farmacéutico especializado y dedicado a la investigación, desarrollo, fabricación y comercialización nacional e internacional bajo licencia de productos farmacéuticos (medicamentos).

Esta empresa societaria tiene su sede principal en el Polígono Industrial El Cerbellón en Paracuellos del Jarama donde se llevará a cabo la elaboración nivel nacional e internacional, así como la comercialización y venta a nivel únicamente nacional de los productos farmacéuticos.

En la sede secundaria ubicada en el Polígono Industrial Zona Oeste en Tres Cantos se llevará a cabo la comercialización y venta a nivel internacional de los productos farmacéuticos.

Es importante tener en cuenta el alcance de comunicación de la tecnología inalámbrica que se emplea en el proyecto (LoRa) para determinar la ubicación de las sedes.

Mapa

Descripción generada automáticamente

*Figura 2. Localización de las sedes de la empresa.*

# Material empleado

Al estar formada dos sedes, existirán datos que se deberán transmitir por comunicación inalámbrica y otros por puerto serie.

* Sede principal en el Polígono Industrial Zona Oeste en Tres Cantos: En la sede principal se recogen los datos de los sensores de control de la sala donde se ubica el sistema principal (Raspberry Pi), transmitiéndose los datos por puerto serie, así como la zona de empaquetado final.

Tabla 1. Material necesario para simular la sede principal.

|  |  |
| --- | --- |
| Material | Precio |
| Arduino UNO | 29,28 € |
| Sensor de temperatura | 5,24 € |
| Cable USB para Arduino | 3,96 € |
| Arduino IoT | 25,49 € |
| Sensor de distancia | 1,40 € |
| Sensor de pesaje | 2,52 € |

* Sede secundaria en el Polígono Industrial El Cerbellón en Paracuellos del Jarama: En la sede secundaria se recogen los datos de los sensores de control del propio proceso de producción, transmitiéndose los datos por comunicación inalámbrica (LoRa).

Tabla 2. Material necesario para simular la sede secundaria.

|  |  |
| --- | --- |
| Material | Precio |
| Arduino UNO | 29,28 € |
| Sensor de distancia | 1,40 € |
| Cable USB para Arduino | 3,96 € |
| Sensor de humedad | 2,05 € |
| Sensor de pesaje | 2,52 € |
| Sensor de partículas | 27,79 € |
| Sensor de presión | 5,99 € |

Además de estos materiales, en la sede principal estarán ubicados diferentes componentes necesarios para la transmisión de datos.

Tabla 3. Material genérico necesario.

|  |  |
| --- | --- |
| Material | Precio |
| Raspberry Pi | 55,90 € |
| Módulo inalámbrico LoRa | 34.99 € |
| Cable Micro USB | 3,22 € |
| Tarjeta SD | 7,71 € |

Un escritorio con una computadora

Descripción generada automáticamente con confianza media

Figura 3. Montaje de la maqueta del proyecto.

# Roles y funciones

Es fundamental definir los roles y las responsabilidades dentro de la organización para garantizar un correcto sistema de gestión de calidad y cumplir con la norma ISO 9001.

* Víctor Manuel Navarro Pérez: Responsable de HW, programador SW Raspberry Pi, implementador de las comunicaciones inalámbricas y responsable de las pruebas.
* Cristina Fernández García: Responsable de SW, programador SW Arduino, programador de la BBDD y responsable de las pruebas.

Tabla 4. Funciones de los distintos roles.

|  |  |  |  |
| --- | --- | --- | --- |
| Responsable de HW | Responsable de SW | Programador SW Raspberry Pi | Programador SW Arduino |
| Planificar, gestionar y diseñar las modificaciones de la parte del hardware del proyecto | Garantizar el correcto funcionamiento del sistema | Configurar la RPI y garantizar el correcto funcionamiento del servidor | Traducir a líneas de código la lectura de los sensores |

|  |  |  |
| --- | --- | --- |
| Implementador de las comunicaciones inalámbricas | Programador de la BBDD | Responsable de las pruebas |
| Instalar los sistemas de red y asegurar el correcto funcionamiento de las comunicaciones | Creación y mantenimiento de la base de datos | Realización de pruebas de validación del sistema para comprobar el correcto funcionamiento del mismo |

# Tamaño de almacenamiento del proyecto

Tabla 5. Tamaño de almacenamiento del proyecto.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tamaño de cada muestra = 32,6 B/med | | | | | | |
| Tasa de medidas | 2,33 med/s | 139,8 med/min | 8.388 med/h | 201.312 med/día | 6.039.360 med/mes | 72.472.320 med/año |
| Tamaño almacenamiento | 75,96 B/s | 4,6 kB/min | 273,46 kB/h | 6,56 MB/día | 196,89 MB/mes | 2.36 TB/año |

# Sensores

En la industria farmacéutica, los sensores de temperatura desempeñan un papel crucial en la monitorización y control de las condiciones ambientales para garantizar la calidad y la integridad de los productos farmacéuticos. Estos sensores permiten medir y registrar con precisión la temperatura en áreas críticas.



*Figura 4. Sensor de temperatura.*

Los sensores de presión al igual que los sensores de temperatura, son muy importantes para la monitorización y el control de los diferentes procesos y equipos. Estos sensores permiten medir y supervisar la presión en áreas críticas.

Cámara de video

Descripción generada automáticamente con confianza baja

*Figura 5. Sensor de presión.*

Por otro lado, los sensores de humedad desempeñan igualmente un papel importante en el control de las condiciones ambientales para garantizar la calidad y estabilidad de los procesos farmacéuticos. Estos sensores permiten medir y monitorizar el nivel de humedad relativa en las áreas críticas.

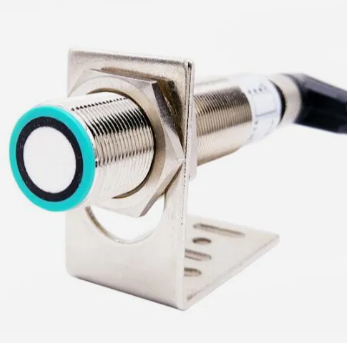


*Figura 6. Sensor de humedad.*

En la industria farmacéutica, los sensores de distancia por ultrasonidos se utilizan para medir distancias y detectar objetos en diferentes aplicaciones. Estos sensores utilizan ondas ultrasónicas de alta frecuencia para determinar la distancia desde el sensor hasta un objeto o una superficie.

Teniendo en cuenta que este tipo de sensores se emplean en diversas aplicaciones, para este proyecto solo se utilizan para las siguientes funciones:

* En la sede principal, se emplean para el control de nivel de líquidos, es decir, se utilizan para medir el nivel de líquido en tanques para controlar los niveles de llenado y evitar los derrames.
* En la sede secundaria, se emplean para el seguimiento de movimiento, es decir, estos sensores monitorizan la posición de los objetos (paquetes) para permitir el movimiento de una cantidad fija de los mismos y llevar a cabo su empaquetamiento.



*Figura 7. Sensor de distancia por ultrasonidos.*

Otro sensor comúnmente empleado en la industria farmacéutica es el sensor de partículas por dispersión láser. Se emplean para contar y monitorizar la concentración de partículas en el aire de entornos críticos. Estos sensores son fundamentales para garantizar que el ambiente de fabricación y producción cumplan con los estándares de calidad y pureza requeridos.



*Figura 8. Sensor de partículas.*

Por último, se ha incorporado en el proyecto los sensores de pesaje que desempeñan un papel crucial en la medición precisa de la masa de sustancias en los procesos de fabricación y control de calidad, es decir, se emplean para garantizar la exactitud en la dosificación del producto farmacéutico.



*Figura 9. Sensor de pesaje.*

# Modelo Hardware

Para llevar a cabo el proyecto descrito anteriormente, es necesario conectar los diferentes dispositivos con los que se cuentan:

* La conexión de la placa Arduino Mega 1 al emisor LoRa se realiza a través del puerto serie (UART), lo que implica utilizar los pines de comunicación serial del Arduino para establecer la comunicación con el emisor.
* La conexión entre el emisor (transmisor) y el receptor LoRa se realiza utilizando la tecnología de comunicación inalámbrica LoRa (“Long Range”).
* La conexión entre el receptor LoRa y la Raspberry Pi se realiza a través del puerto serie (UART), lo que implica utilizar los pines de comunicación seril de la Raspberry Pi para establecer la comunicación con el receptor.
* La conexión de la Raspberry Pi a Internet se realiza a través de Ethernet, lo que implica utilizar el puerto Ethernet incorporado en la Raspberry Pi para establecer una conexión por cable con un router.
* La conexión entre el ordenador y la Raspberry Pi se realiza utilizando el protocolo de control de transmisión TCP/IP, lo que implica una conexión de red entre los dos dispositivos a través de una red local o a través de Internet.
* La conexión del ordenador a Internet se realiza a través de Wi-Fi.
* La conexión de la placa Arduino Mega 2 a la Raspberry Pi se realiza a través del puerto serie (UART), lo que implica utilizar los pines de comunicación serial de ambos dispositivos para establecer la comunicación.
* La conexión de la placa Arduino IoT a Internet se realiza a través de Wi-Fi, por lo que será necesario utilizar un módulo Wi-Fi compatible con Arduino.



Figura 10. Diagrama de bloques del proyecto.

Diagrama

Descripción generada automáticamente

*Figura 11. Modelo Hardware del proyecto.*

# Modelo Software

Diagrama

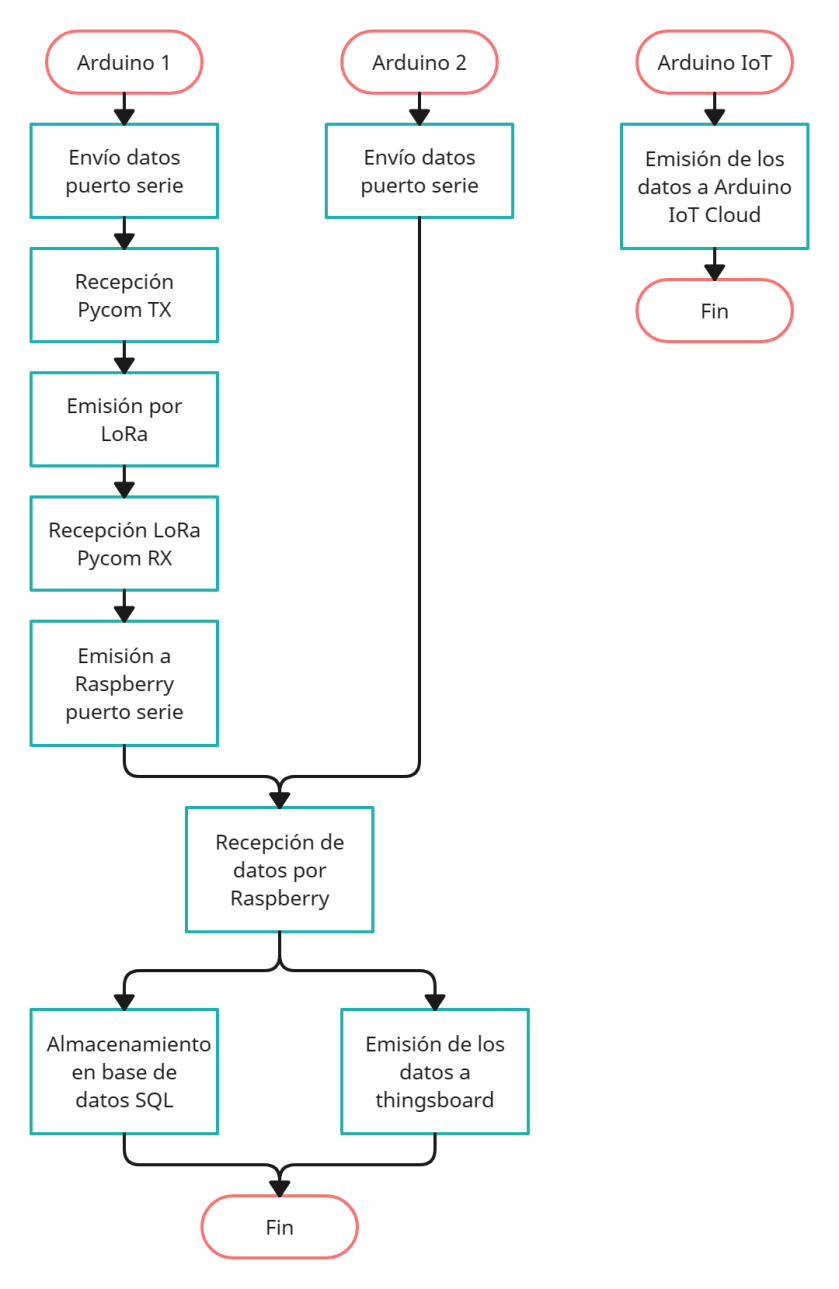
Descripción generada automáticamente

Figura 12. Modelo software del proyecto.

En este apartado se muestra el diagrama de flujo del sistema a grandes rasgos. Como se puede observar, hay tres ramas, dos de ellas siendo confluyentes. La primera empezando por la izquierda corresponde al Arduino instalado en la “Sede de Tres Cantos” y será el encargado de enviar los datos de todo el proceso productivo a la base de datos a través del protocolo LoRa, ya que el servidor se encuentra en Paracuellos del Jarama, a una distancia física de poco menos de 10 km.

La segunda rama corresponde al Arduino instalado en la misma caja del servidor y se encargará exclusivamente de medir las condiciones interiores de ésta para evitar sobrecalentamientos, aunque en un futuro podría desempeñar funciones adicionales. Si conexión será directa al servidor a través de puerto serie.

Por último, la tercera rama corresponde al Arduino IoT (nano 33), que se encarga de controlar el proceso productivo de la sede principal, localizada en Paracuellos del Jarama. Se ha considerado adecuando, ya que esta sede se dedica principalmente al empaquetamiento y distribución de los productos y no se tienen condiciones tan limitantes. Sus datos se envían mediante WiFi a un dashboard creado en Arduino IoT Cloud y puede ser visualizado hasta a través de un smartphone con facilidad.



*Figura 13. Diagrama de flujo de la parte software.*

# Base de datos

Para gestionar la base de datos del proyecto se hace uso de la herramienta “HeidiSQL” que se emplea como interfaz gráfica que permite la administración y la consulta de bases de datos.

Al usar HeidiSQL es posible conectarse a la base de datos del proyecto existente y crear y modificar tablas.

Para conectarse a una base de datos en HeidiSQL, se debe proporcionar la información de conexión, como el host o servidor de la base de datos, el puerto, el nombre de usuario y la contraseña. Una vez que te hayas conectado correctamente, podrás explorar la estructura de la base de datos, ver las tablas y los datos almacenados, y ejecutar consultas para realizar operaciones en la base de datos.

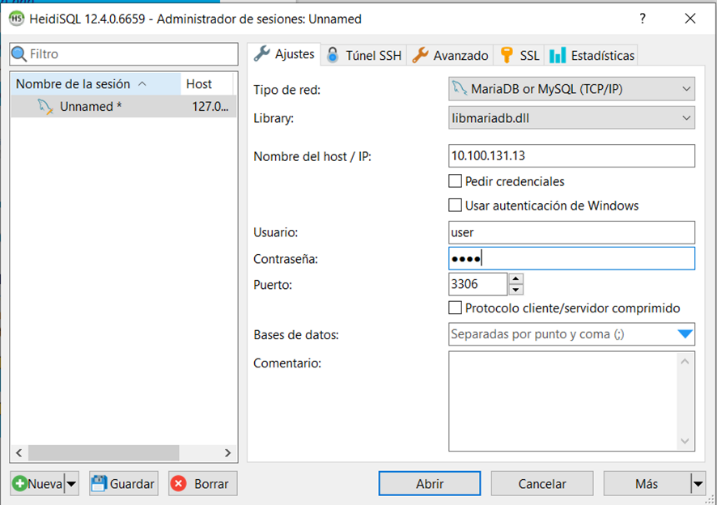


Figura 14. Información de conexión para acceder a la base de datos de proyecto.

**NOTA:** Credenciales “HeidiSQL”

* Usuario: user
* Contraseña: 1234

La base de datos del proyecto tiene asociado el nombre “Proyecto\_Farma”, en la cual es necesario crear dos tablas distintas, en una de ella se registrará el inventario de sensores empleados y en la otra tabla se registrarán los datos recogidos por los diferentes sensores.

La tabla “Inventario” estará compuesta por dos columnas y siete filas. En el caso de las columnas, la primera de ellas representará la ID del sensor mientras que la segunda columna muestra el nombre del valor medido por el sensor. La primera columna se categoriza como “Primary Key”, lo que implica identificar de manera única cada fila sin ser posible los valores nulos.

Tabla 6. Inventario de sensores empleados.

|  |  |
| --- | --- |
| ID | SENSOR |
| 1 | Distancia [cm] |
| 2 | Presión [Pa] |
| 3 | Humedad [%] |
| 4 | Contador de partículas 0,5 µm |
| 5 | Contador de partículas 5 µm |
| 6 | Peso [g] |
| 7 | Temperatura [ºC] |

La tabla “Datos” estará compuesta por cuatro columnas y tantas filas como valores se registren (se irán rellenando automáticamente cuando se recojan los valores de los diferentes sensores).

La primera columna representará la ID del valor recogido, la segunda columna representará la ID del sensor al que hace referencia cada valor, la tercera columna representa el momento en el que se registra el dato (día y hora del registro) y la cuarta y última columna refleja el valor registrado por los sensores en cada momento.

Tabla 7. Datos registrados por los sensores.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | IDSENSOR | TIME | VALOR |
| 1 | … | … | … |
| … | … | … | … |

# Comunicación Arduino – Raspberry Pi

Debido a la necesidad de comprobar la correcta conectividad entre las placas Arduino y la Raspberry Pi para detectar fallos en el sistema de control de las plantas farmacéuticas es necesario recibir los registros de los sensores de los dos puertos configurados (el puerto “ttyACM1” para la sede secundaria de Paracuellos del Jarama y el puerto “ttyACM0” para la sede principal de Tres Cantos).

Para poder comprobar que la Raspberry Pi recibe los datos del Arduino (tanto por puerto serie como por LoRa) se emplea el código de Python “1\_iot\_read.py” especificado en el Anexo 4.

En este código se crean dos instancias (una por cada Arduino conectado a la Raspberry Pi) especificando el puerto serie al que está conectado y la velocidad de transmisión, para a continuación leer una línea de datos del Arduino e imprimirlo por pantalla y cerrar todas las conexiones.

Texto

Descripción generada automáticamente

Figura 15. Correcta comunicación Arduino - Raspberry Pi.

Con el fin comprobar la correcta comunicación entre la Raspberry y las placas Arduino, con un tratamiento de los datos recibidos, se emplea el código de Python “2\_iot\_cad.py” especificado en el Anexo 5.

En este código, se toma una cadena de texto y se realiza un procesamiento específico. En este caso, la función busca patrones dentro de la cadena y extrae la información relevante, para a continuación mostrar por pantalla los valores de los distintos sensores y el id del sensor.

Texto

Descripción generada automáticamente

Figura 16. Tratamiento de los datos recibidos.

Una vez se ha comprobado que la comunicación es correcta, es necesario configurar la Raspberry Pi para que estos registros sean almacenados en la base de datos mencionada en el apartado 8. Para ello, se emplea el código de Python “3\_iot\_bbdd.py” especificado en el Anexo 6.

En este código se crea una función encargada de enviar los datos recibidos al servidor MySQL. Esta función establece una conexión con la base de datos utilizando los parámetros de conexión adecuados. A continuación, se construye una consulta SQL para insertar el id de los sensores y el valor recibido en la tabla “Datos”. Por último, se ejecuta la consulta, se confirma la transacción y se cierran el cursor y la conexión con la base de datos.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 17. Comprobación del almacenamiento de datos en la base de datos.

# Interfaces para la visualización de datos

* 1. Thingsboard

Thingsboard es una plataforma de IoT de código abierto que permite a los usuarios recopilar, visualizar y analizar datos de dispositivos conectados a Internet.

Tras crear el panel del proyecto se pueden visualizar los widgets donde se representan los datos recopilados por los sensores.

En primer lugar, se debe crear un nuevo código desarrollado en python que se pueda comunicar con la plataforma de thingsboard. Dicho código se encuentra en el anexo 7 de este documento. Además, se debe hacer que se ejecute al iniciar la raspberry.

Posteriormente, una vez que ya enviamos los datos, se debe definir un nuevo panel para poder crear el dashboard apropiado para nuestro proyecto. En nuestro caso, el dashboard que usaremos será para monitorizar una sede localizada en Paracuellos del Jarama, que se encarga de fabricar productos farmacéuticos. Debido a las estrictas condiciones y peligrosidad de algunos procesos productivos necesarios, es preciso localizar la base en un polígono industrial a las afueras de núcleos poblacionales. Dicha creación del panel se puede observar en la siguiente imagen:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Figura 18. Creación del panel para Thingsboard.

Una vez creado el panel, se deben añadir los widgets necesarios para nuestra aplicación, asociando cada uno con la variable que deseamos mostrar. En este proyecto desarrollamos el siguiente dashboard:

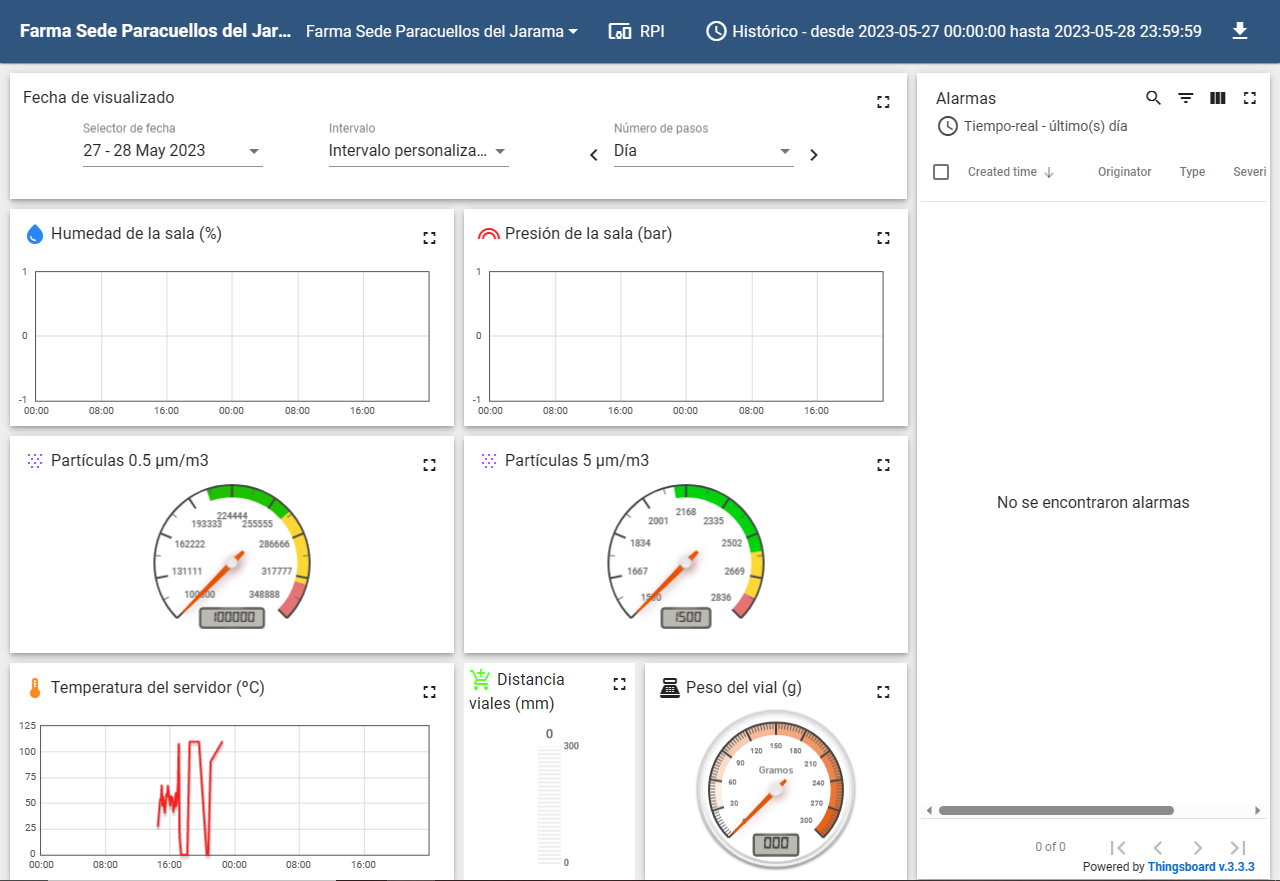


Figura 19. Dashboard de Thingsboard.

Este dashboard contiene:

* Controlador de fecha: Usado para poder definir la fecha e intervalo de tiempo con el que se quieren visualizar los datos almacenados.
* Gráficas de humedad y presión de la sala: Para controlar que el proceso se realice en condiciones adecuadas.
* Contadores de partículas de 0.5 y 5 µm/m3.
* Medidor de temperatura de la caja del servidor: Para evitar situaciones de sobrecalentamiento en el sistema.
* Un medidor de distancia: Usado para medir la distancia que hay desde el sensor al último vial introducido en una línea. Si se sabe que el vial tiene un tamaño determinado, a través de la distancia se puede saber el número de viales concretos que hay en ella. Cuando se llega a un límite determinado de viales, éstos se retiran de la línea para realizar su empaquetamiento.
* Un medidor del peso de cada vial: Para comprobar que su peso sea adecuado.
* Un controlador de alarmas: Para observar las alarmas que salten en el sistema, su estado y poder administrarlas.
  1. Google Chart

Google chart es una biblioteca de visualización de datos de código abierto que permite crear gráficos utilizando HTML, CSS y JavaScript.

Para explotar los datos de la base de datos con Google Chart se hará uso de la aplicación “WinSCP”, que permitirá copiar o mover a la ruta “/var/www/html” el código de explotación de datos dividido en cuatro archivos:

* config.php: contiene los datos de conexión a la base de datos.
* connectBD.php: crea y destruye la conexión a la bbdd.
* funciones.php: mediante una Query en SQL recopila los datos y les da el formato del tipo de gráfica elegida de Google Chart. El formato es una matriz de una pareja de datos ([valorX,valorY]).
* googlechart.php: es el archivo que se busca desde el navegador. Es el que invoca la API de Google Chart y al que se le pasan los datos a dibujar.

Es necesario modificar los códigos “config.php”, “funciones.php” y “googlechart.php”, mientras que el código “connectBD.php” se queda igual.

En el archivo “config.php” se deben cambiar las credenciales de conexión a la bbdd (usuario, password y nombre de la bbdd).

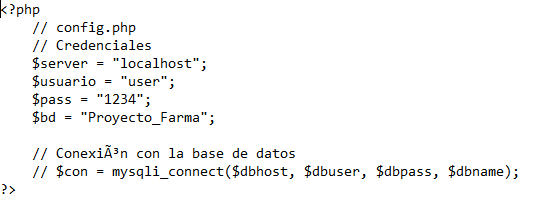


Figura 20. Cambios a “config.php”.

En “funciones.php” se debe cambiar el nombre de nuestra variable de valor, de tiempo y del ID de los sensores.

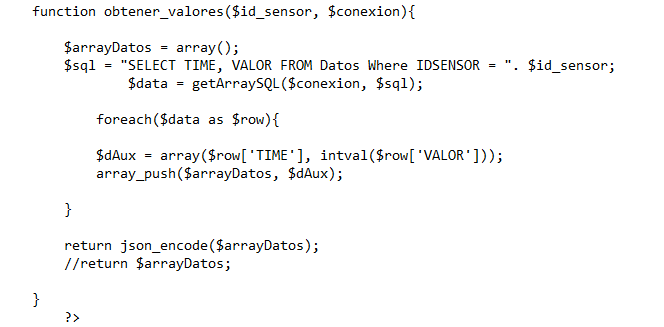


Figura 21. Cambios a “funciones.php”.

Por último, el “googlechart.php” se recoge en el anexo 8.

Una vez modificados los archivos, es necesario subirlos al directorio remoto “/var/www/html”.

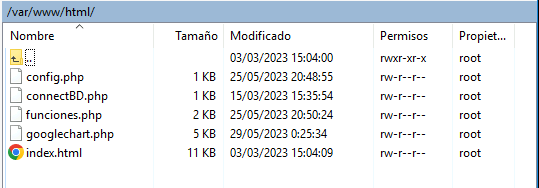


Figura 22.Copia en ruta de los archivos.

Una vez subidos dichos archivos al directorio seleccionado, desde el navegador se accede al directorio remoto para poder ver los gráficos.

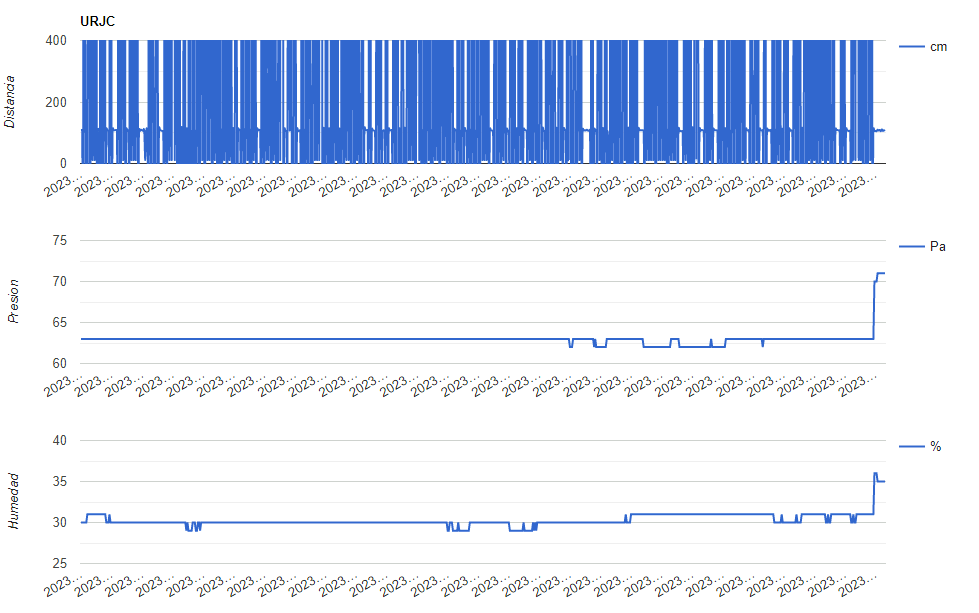


Figura 23.Visualización por GoogleChart, primera parte.

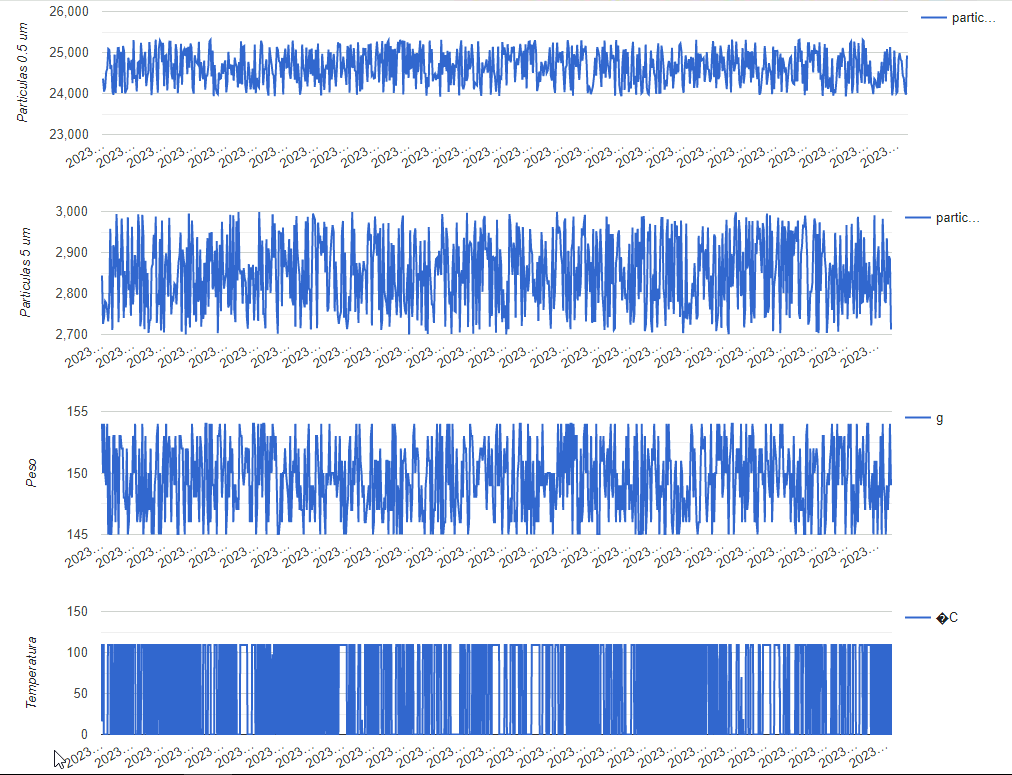


Figura 24.Visualización por GoogleChart, segunda parte.

* 1. Arduino IoT Cloud

Arduino IoT Cloud es una plataforma en la nube proporcionada por Arduino que facilita la conexión y la administración de dispositivos Arduino en el contexto del Internet de las cosas (IoT).

Arduino IoT Cloud permite registrar y administrar dispositivos Arduino compatible, así como también proporciona una interfaz basada en web para crear paneles de control personalizados con el fin de visualizar y controlar los datos recogidos por los dispositivos. Esta plataforma permite definir reglas y acciones automatizadas en función de los datos recibidos.

En primer lugar deberemos asociar un dispositivo Arduino compatible (en nuestro caso usaremos un Arduino nano 33) conectándolo mediante USB al ordenador.

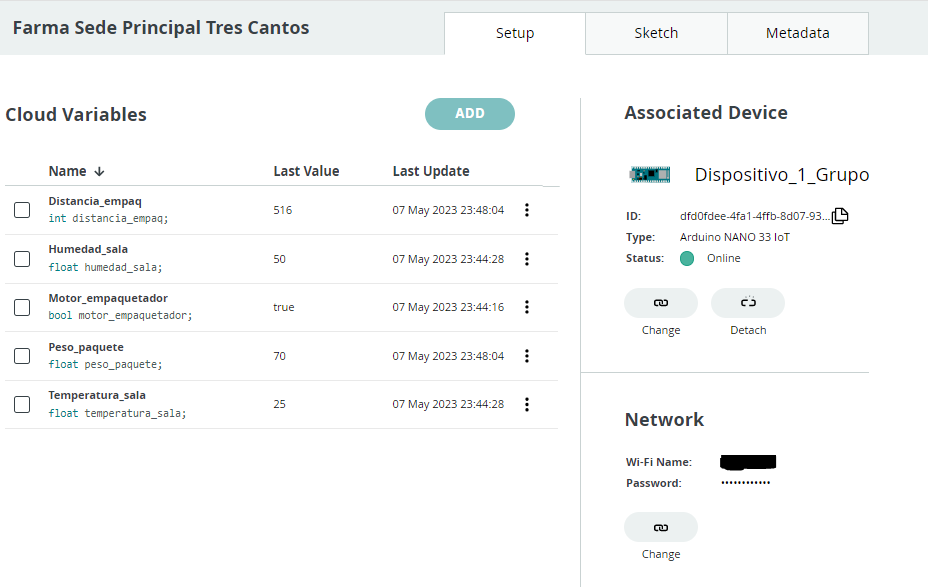


Figura 25. Asociación de un Arduino a la plataforma IoT.

Posteriormente, en el apartado “sketch”, abrimos el “full editor”.

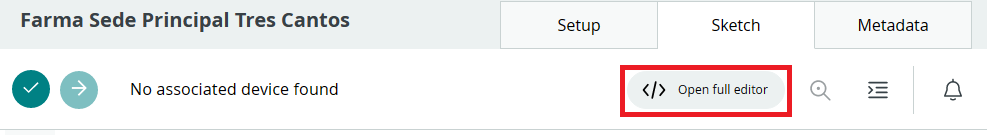


Figura 26. Full editor.

Luego, en el apartado “secrets”, definiríamos las credenciales de la red WiFi a la que conectaríamos el sistema.

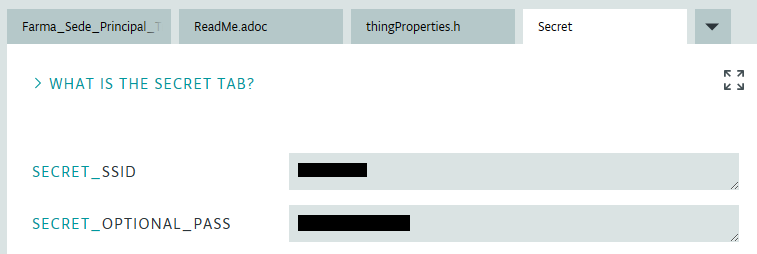


Figura 27. Credenciales del WiFi en secret.

Con esto deberíamos tener el Arduino conectado a la red y listo para enviar datos. Ahora se deberá señalar qué datos son los que se quieren enviar, llamados en Arduino Cloud como “cloud variables” o variables en la nube, que tendrán que corresponder en tipo de datos a lo que luego queramos representar.

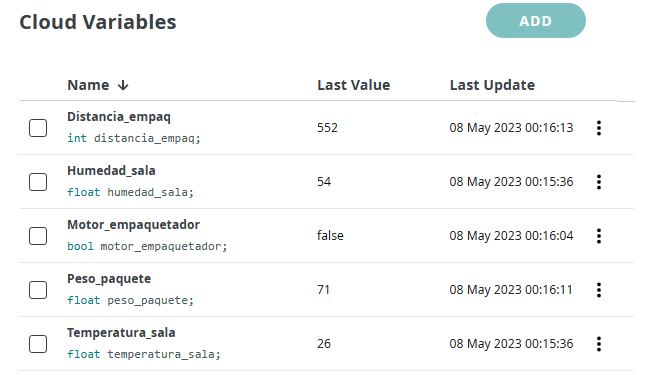


Figura 28. Variables en la nube creadas.

A continuación, se deberá subir el código al Arduino, contemplando en todo momento la inclusión de las variables en la nube antes creadas. Dicho código está incluido en el anexo 1.

Para terminar, se deberá crear el dashboard a nuestro gusto y según las funcionalidades que queramos explotar. En nuestro caso hemos creado el siguiente:

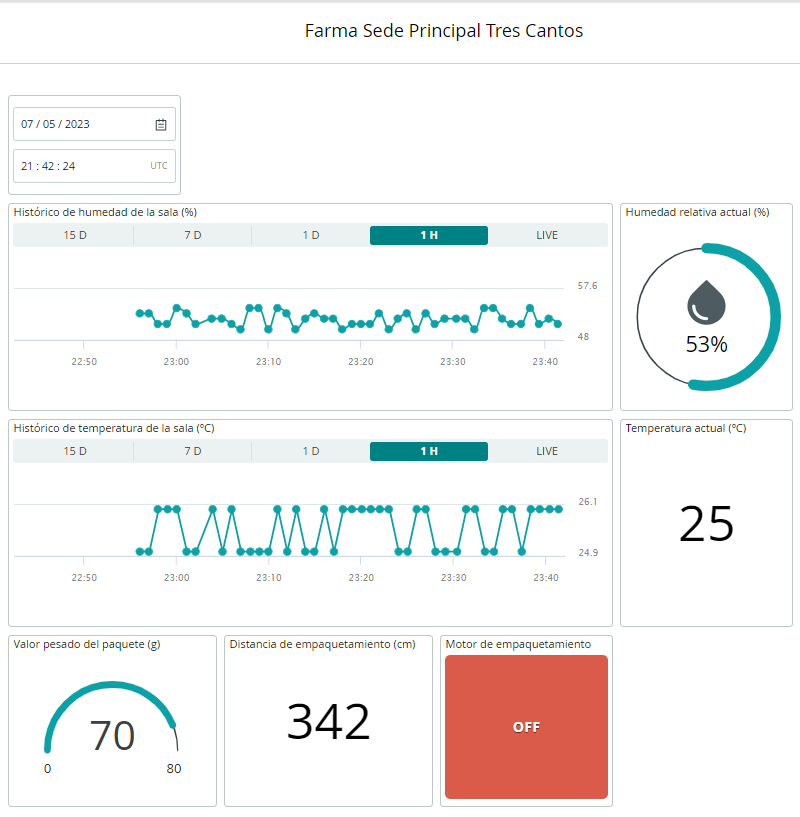
**

Figura 29. Dashboard para el Arduino IoT.

Este simple, pero funcional dashboard servirá para monitorizar la sede de Tres Cantos, dedicada principalmente para el empaquetamiento de los productos. Incluye:

* Controlador de fecha: Usado para poder definir la fecha e intervalo de tiempo con el que se quieren visualizar los datos almacenados.
* Gráficas de humedad y presión de la sala: Para controlar que el proceso se realice en condiciones adecuadas.
* Un medidor de distancia: Usado para medir la distancia que hay desde el sensor al último paquete introducido en una línea. Si se sabe que el paquete tiene un tamaño determinado, a través de la distancia se puede saber el número de paquetes concretos que hay en ella. Cuando se llega a un límite determinado de paquetes, éstos se retiran de la línea para realizar su paletización y posterior carga en un camión.
* Un medidor del peso de cada paquete: Para comprobar que su peso sea adecuado.
* Piloto de estado del motor de empaquetamiento: Para controlar que el motor se esté activando cuando se llega a la cifra objetivo.

En concreto, cuando el sensor de distancia llega a 6cm, el motor debería activarse para retirar los paquetes de la línea, como se puede ver en la siguiente imagen, donde, debido a la activación del motor, los paquetes se han liberado de la línea, causando que el sensor de distancia vuelva a su valor máximo:

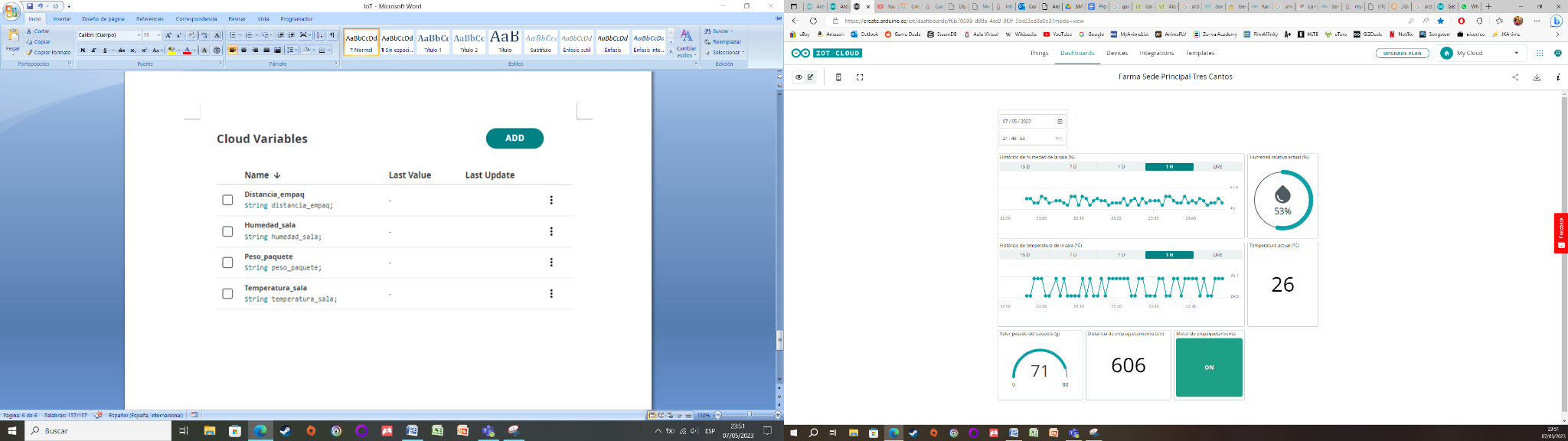
**

Figura 30. Muestra del funcionamiento del motor de empaquetamiento.

Además. Se ha redistribuido y diseñado una interfaz móvil para que se pueda monitorizar todo el proceso desde un smartphone.

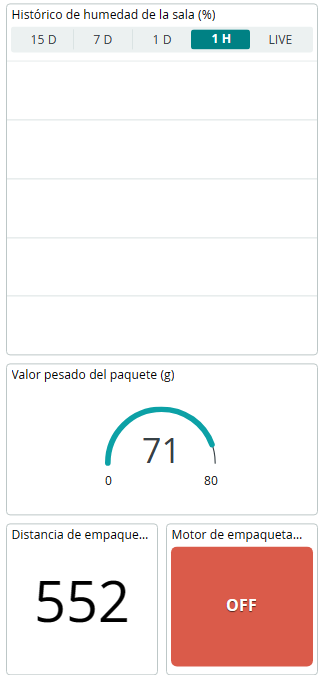
* *

Figura 31. Interfaz de usuario de smartphone. Son dos imágenes sucesivas, de derecha a izquierda.



# Avisos y alarmas

Los límites de aviso y alarma pueden variar según las regulaciones y los estándares de calidad específicos de cada país y empresa.

* Límite de aviso: El límite de aviso se establece para indicar que se ha alcanzado un nivel que requiere atención y seguimiento, aunque aún no se considera crítico. Por lo general, el límite de aviso se establece en un nivel que representa un incremento significativo en comparación con los niveles normales.
* Límite de alarma: El límite de alarma se establece en un nivel que indica una desviación importante de los niveles normales y puede indicar un problema en la calidad del aire o en el proceso de producción. Cuando se alcanza el límite de alarma, se requiere una acción inmediata para investigar y corregir la causa de dicha desviación.

Tabla 8. Límite máximo para el sensor de temperatura.

|  |
| --- |
| Temperatura [ºC] |
| Máximo |
| 100 |

Tabla 9. Límite máximo para el sensor de humedad.

|  |  |  |
| --- | --- | --- |
| Humedad [%] | | |
| Grado | **Aviso** | **Alarma** |
| A | 30 | 35 |
| B | 40 | 45 |
| C | 55 | 60 |
| D | 55 | 60 |

Tabla 10. Límite mínimo para el sensor de presión.

|  |  |  |
| --- | --- | --- |
| Presión [Pa] | | |
| Grado | **Aviso** | **Alarma** |
| A | 58 | 55 |
| B | 42 | 40 |
| C | 23 | 21 |
| D | 11 | 10 |

Tabla 11. Límite máximo para el sensor de partículas.

|  |  |  |
| --- | --- | --- |
| Contador de partículas | | |
| Grado | **0,5 µm/m3** | **5 µm/m3** |
| A | 352.000 – Ahora 35.200 | 2.900 |
| B | 352.000 | 2.900 |
| C | 3.520.000 | 29.000 |
| D | 3.520.000 | 29.000 |

Nota: Se ha cambiado el valor marcado en rojo a un valor inferior porque el Thingsboard no representaba correctamente los datos.

Tabla 12. Límite máximo para el sensor de mínimo y máximo de pesaje.

|  |  |
| --- | --- |
| Pesaje [g] | |
| Mínimo | **Máximo** |
| 147.5 | 152.5 |

Por lo tanto, para cada parámetro hay establecidos unos límites, y cuando su valor está fuera de límites durante un tiempo prefijado, el sistema genera una alarma que queda registrada en el sistema.

Una alarma, implica una condición anormal del sistema que impide su funcionamiento dentro de los parámetros operativos, por lo que el sistema se detiene a la espera de la corrección de la situación anormal; mientras que el aviso implica una condición anormal del sistema que no impide su funcionamiento dentro de los parámetros operativos, por lo que el sistema continúa funcionando dentro de los parámetros operacionales.

En el caso de que se genere una alarma en las salas se activan las balizas visuales y sonoras, y, por otro lado, se enviará un correo electrónico a los usuarios definidos.

Para definir las alarmas mencionadas anteriormente en thingsboard se deben crear cadenas de reglas para cada una de las condiciones, discerniendo en si son condiciones que presentan una situación crítica o un simple aviso. Las cadenas de reglas definidas se pueden ver en la siguiente imagen:

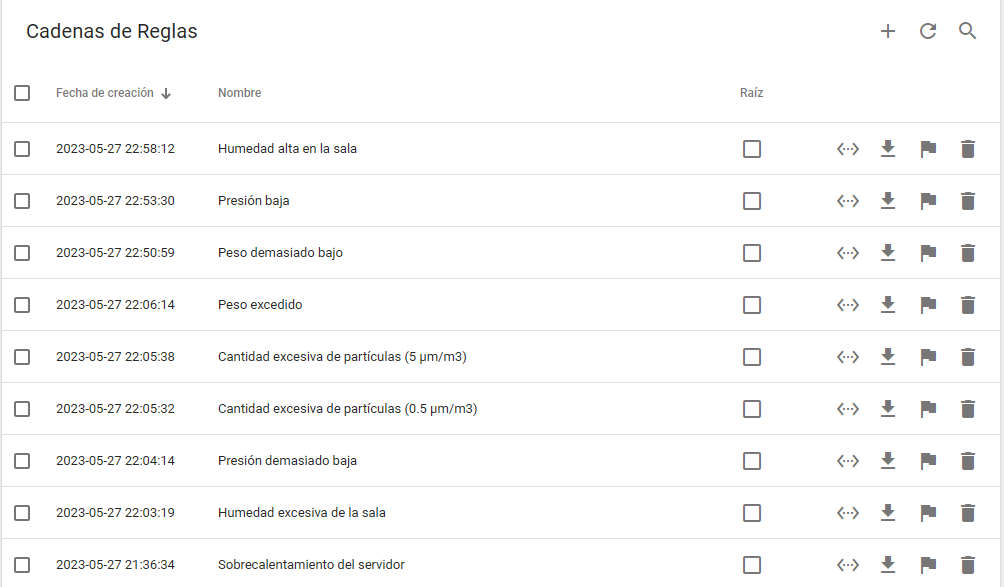


Figura 32. Cadenas de reglas para alarmas en thingsboard.

Cada una de las cadenas de reglas seguirá la siguiente estructura:

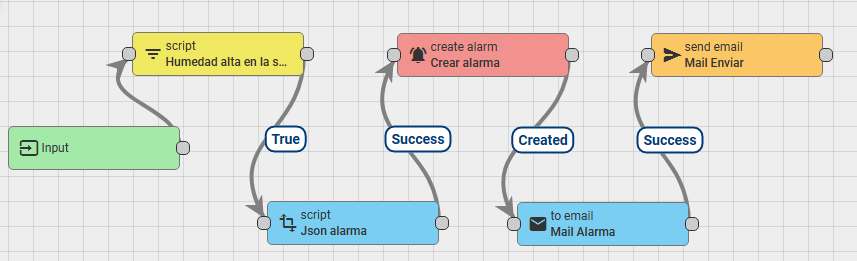


Figura 33. Estructura de las cadenas de reglas en thingsboard.

La estructura corresponde a, en sucesión:

* Una entrada de datos, donde se recibirá el dato correspondiente solicitado.
* Una condición de activación.
* Una codificación a Json.
* La creación de la alarma y su configuración.
* El direccionamiento y template para el email de la alarma.
* El actuador de envío del email.

# GitHub

El proyecto ha sido documentado en la plataforma GitHub, cuyo enlace se encuentra en la bibliografía del proyecto.

GitHub es una plataforma de desarrollo colaborativo que permite a los desarrolladores alojar, revisar y compartir código, así como trabajar en colaboración con otros desarrolladores en proyectos de software.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Figura 34. GitHub del proyecto.

# Mejoras futuras

Como mejoras futuras se han contemplado las siguientes opciones:

* Implementar un sistema de control y depuración de fallos.
* Implementar un código de Arduino con interrupciones y Timers.
* Implementar un modo de funcionamiento de bajo consumo.
* Desarrollar una página web para el proyecto.
* Combinar las dos sedes en un mismo dashboard.
* Crear una nueva interfaz de visualización alternativa como Grafana.
* Mejorar la interfaz GoogleChart.
* Generar un plano de localización de sensores en la base de datos para poder ubicarlos rápidamente en el caso de un mantenimiento correctivo.
* Incorporar un mayor número de actuadores.

# Conclusiones

Se puede discernir con el trabajo realizado que la implementación de una monitorización a distancia junto con una monitorización cableada es posible para un ámbito industrial y abre posibilidades de control y eficiencia frente a alternativas o aplicaciones más tradicionales.

Estas posibilidades son económicamente viables y permiten posibles aplicaciones futuras de ámbitos propios de la industria 5.0.

# Bibliografía

<https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan#:~:text=Largo%20alcance%2010%20a%2020,y%20433%20Mhz%20en%20Asia>

https://github.com/CristinaFG/Farma-Industries

# Anexos

* 1. Anexo 1: Código Arduino IoT

#include "thingProperties.h"

#include <OneWire.h>

float Temperatura\_sala\_temp;

float Humedad\_sala\_temp;

float Peso\_paquete\_temp;

int Distancia\_empaq\_temp = 606; // para 100 cajas (contando con ancho 6 cm) + tolerancia

int peso = 0;

void setup() {

// Initialize serial and wait for port to open:

Serial.begin(9600);

// This delay gives the chance to wait for a Serial Monitor without blocking if none is found

delay(1500);

// Defined in thingProperties.h

initProperties();

// Connect to Arduino IoT Cloud

ArduinoCloud.begin(ArduinoIoTPreferredConnection);

setDebugMessageLevel(2);

ArduinoCloud.printDebugInfo();

}

void loop() {

ArduinoCloud.update();

// Your code here

Temperatura\_sala\_temp = random(2500,2700)/100;

temperatura\_sala = Temperatura\_sala\_temp;

Humedad\_sala\_temp = random(5000,5500)/100;

humedad\_sala = Humedad\_sala\_temp;

if (peso = 0) {

Peso\_paquete\_temp = 0;

peso\_paquete = Peso\_paquete\_temp;

peso = 1;

}

else {

Peso\_paquete\_temp = random(6900,7200)/100;

peso\_paquete = Peso\_paquete\_temp;

peso = 0;

if (Distancia\_empaq\_temp > 6) {

Distancia\_empaq\_temp = Distancia\_empaq\_temp-6; //cm

distancia\_empaq = Distancia\_empaq\_temp;

motor\_empaquetador = 0;

}

else {

motor\_empaquetador = 1;

delay(500);

Distancia\_empaq\_temp = 606;

distancia\_empaq = Distancia\_empaq\_temp;

}

}

delay(1000);

}

* 1. Anexo 2: Código Arduino LoRa

float distancia\_cm = 0;

int echoPin = 7;

int triggerPin = 6;

float presion\_Pa;

float peso\_g;

float humedad\_minima = 20;

float humedad\_maxima = 40;

float presion\_minima = 53;

float presion\_maxima = 80;

float humedad\_anterior;

float humedad;

float presion\_anterior;

float presion;

float suma\_humedad;

float suma\_presion;

int particulas\_05;

int particulas\_5;

int peso;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT);

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

return pulseIn(echoPin, HIGH);

}

void setup()

{

Serial.begin(9600);

humedad = 30;

humedad\_anterior = 25;

presion = 60;

presion\_anterior = 75;

delay(101);

}

void loop()

{

distancia\_cm = 0.01723 \* readUltrasonicDistance(triggerPin,echoPin);

suma\_presion = random(0,9)/100.0;

presion\_anterior = presion;

if (presion\_minima < presion && presion < presion\_maxima)

{

int operacion\_presion = random(0, 2);

if (operacion\_presion == 0)

{

presion = presion - suma\_presion;

}

else

{

presion = presion + suma\_presion;

}

}

else

{

if (presion\_minima > presion)

{

presion = presion + suma\_presion;

}

if (presion > presion\_maxima)

{

presion = presion - suma\_presion; // Sumar el valor generado

}

}

suma\_humedad = random(0,9)/100.0;

humedad\_anterior = humedad;

if (humedad\_minima < humedad && humedad < humedad\_maxima)

{

int operacion\_humedad = random(0, 2);

if (operacion\_humedad == 0)

{

humedad = humedad - suma\_humedad;

}

else

{

humedad = humedad + suma\_humedad;

}

}

else

{

if (humedad\_minima > humedad)

{

humedad = humedad + suma\_humedad;

}

if (humedad > humedad\_maxima)

{

humedad = humedad - suma\_humedad;

}

}

peso = random(145,155);

particulas\_05 = random(351600,353000);

particulas\_5 = random(2700,3000);

Serial.println("#1:"+String(distancia\_cm)+"#2:"+String(presion)+"#3:"+String(humedad)+"#4:"+String(particulas\_05)+"#5:"+String(particulas\_5)+"#6:"+String(peso)+"@");

delay(3000);

}

* 1. Anexo 3: Código Arduino Puerto Serie

float temperatura\_C = 0;

int temperatura = A0;

void setup()

{

Serial.begin(9600);

pinMode(temperatura, INPUT);

}

void loop()

{

temperatura\_C = analogRead(temperatura);

temperatura\_C = (1.1 \* temperatura\_C \* 100.0)/1024.0;

Serial.println("#7:"+String(temperatura\_C)+"@");

delay(3000);

}

* 1. Anexo 4: 1\_iot\_read.py

import serial

arduino1 = serial.Serial('/dev/ttyACM1', 9600)

arduino2 = serial.Serial('/dev/ttyACM0', 9600)

while True:

line1 = arduino1.readline()

print(line1)

line2 = arduino2.readline()

print(line2)

arduino1.close()

arduino2.close()

* 1. Anexo 5: 2\_iot\_cad.py

import serial

arduino1 = serial.Serial('/dev/ttyACM1', 9600)

arduino2 = serial.Serial('/dev/ttyACM0', 9600)

def cad\_proc(cad):

print ("%n%nInicio------------------------------------------------>" + cad)

i = cad.find("@")

while (i > 0):

j = cad.find("#")

cad = cad [j+1:]

aux = cad

j = cad.find("#")

if (j < 0):

j = cad.find("@")

aux = aux[:j]

cad = cad[j:]

x = aux.find(":")

sensor = aux[:x]

value = aux[x+1:]

print("sensor:" + sensor)

print("value:" + value)

i = cad.find("@")

while True:

line1 = arduino1.readline()

cad\_proc(str(line1))

line2 = arduino2.readline()

cad\_proc(str(line2))

arduino1.close()

arduino2.close()

* 1. Anexo 6: 3\_iot\_bbdd.py

import serial

import mysql.connector

arduino1 = serial.Serial('/dev/ttyACM1', 9600)

arduino2 = serial.Serial('/dev/ttyACM0', 9600)

def cad\_proc(cad):

print ("%n%nInicio------------------------------------------------>" + cad)

i = cad.find("@")

while (i > 0):

j = cad.find("#")

cad = cad [j+1:]

aux = cad

j = cad.find("#")

if (j < 0):

j = cad.find("@")

aux = aux[:j]

cad = cad[j:]

x = aux.find(":")

sensor = aux[:x]

value = aux[x+1:]

print("sensor:" + sensor)

print("value:" + value)

send\_mysql(sensor,value)

i = cad.find("@")

def send\_mysql(sensor, value\_):

cnx = mysql.connector.connect(user='user', password='1234', host='127.0.0.1',

database='Proyecto\_Farma')

cursor = cnx.cursor()

query = "Insert into Datos (IDSENSOR,VALOR) VALUES ("+sensor\_ +","+value\_+");"

print(query)

cursor.execute(query)

cnx.commit()

cursor.close()

cnx.close()

while True:

line1 = arduino1.readline()

cad\_proc(str(line1))

line2 = arduino2.readline()

cad\_proc(str(line2))

arduino1.close()

arduino2.close()

* 1. Anexo 7: 6\_iot\_thingsboard.py

import serial

import mysql.connector

import paho.mqtt.client as mqtt

import json

arduino = serial.Serial('/dev/ttyACM0', 9600)

arduino2 = serial.Serial('/dev/ttyACM1', 9600)

THINGSBOARD\_HOST = 'arduo.es'

ACCESS\_TOKEN = 'tjvh2jq9RCFrfNnba1FA'

client = mqtt.Client()

client.username\_pw\_set(ACCESS\_TOKEN)

client.connect(THINGSBOARD\_HOST, 1883, 60)

client.loop\_start()

def cad\_proc(cad):

print ("%n%nInicio------------------------------------------------>" + cad)

i = cad.find("@")

while (i > 0):

j = cad.find("#")

cad = cad [j+1:]

aux = cad

j = cad.find("#")

if (j < 0):

j = cad.find("@")

aux = aux[:j]

cad = cad[j:]

x = aux.find(":")

sensor = aux[:x]

value = aux[x+1:]

print("sensor:" + sensor)

print("value:" + value)

send\_mysql(sensor,value)

i = cad.find("@")

if (sensor == '1'):

print("Sensor1")

sensor\_data = {'Distancia': 0}

sensor\_data['Distancia'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '2'):

print("Sensor2")

sensor\_data = {'Presion': 0}

sensor\_data['Presion'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '3'):

print("Sensor3")

sensor\_data = {'Humedad': 0}

sensor\_data['Humedad'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '4'):

print("Sensor4")

sensor\_data = {'ContadorParticulas05': 0}

sensor\_data['ContadorParticulas05'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '5'):

print("Sensor5")

sensor\_data = {'ContadorParticulas5': 0}

sensor\_data['ContadorParticulas5'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '6'):

print("Sensor6")

sensor\_data = {'Peso': 0}

sensor\_data['Peso'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

elif (sensor == '7'):

print("Sensor7")

sensor\_data = {'Temperatura': 0}

sensor\_data['Temperatura'] = value

client.publish('v1/devices/me/telemetry', json.dumps(sensor\_data), 1)

else:

print("OTHER")

def send\_mysql(sensor\_,value\_):

cnx = mysql.connector.connect(user='user', password='1234',

host='127.0.0.1', database='Proyecto\_Farma')

cursor = cnx.cursor()

query = "Insert into Datos (IDSENSOR,VALOR) VALUES ("+sensor\_ +","+value\_+");"

print(query)

cursor.execute(query)

cnx.commit()

cursor.close()

cnx.close()

while True:

line1 = arduino.readline()

line2 = arduino2.readline()

line = line1 + line2

cad\_proc(str(line))

arduino1.close()

arduino2.close()

* 1. Anexo 8: googlechart.php

<scripttype="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<div id="chart\_div"></div>

<div id="chart\_div2"></div>

<div id="chart\_div3"></div>

<div id="chart\_div4"></div>

<div id="chart\_div5"></div>

<div id="chart\_div6"></div>

<div id="chart\_div7"></div>

<?php

include ('connectBD.php');

include ('funciones.php');

// ConexiÃ³n con la base de datos

$conexion = conectarBD();

$valor\_temp1 = obtener\_valores(1, $conexion);

//var\_dump ($valor\_temp1);die();

$valor\_temp2 = obtener\_valores(2, $conexion);

$valor\_temp3 = obtener\_valores(3, $conexion);

$valor\_temp4 = obtener\_valores(4, $conexion);

$valor\_temp5 = obtener\_valores(5, $conexion);

$valor\_temp6 = obtener\_valores(6, $conexion);

$valor\_temp7 = obtener\_valores(7, $conexion);

desconectarBD($conexion);

?>

<script>

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic1);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic2);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic3);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic4);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic5);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic6);

google.charts.load('current', {packages: ['corechart', 'line']});

google.charts.setOnLoadCallback(drawBasic7);

function drawBasic1() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'datetime');

data.addColumn('number', 'cm');

data.addRows(<?php echo $valor\_temp1;?>);

var options = {

title: 'URJC',

curveType: 'function',

vAxis: {

title: 'Distancia',

viewWindow: {

max:400,

min:0

}

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div'));

chart.draw(data, options);

}

function drawBasic2() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', 'Pa');

data.addRows(<?php echo $valor\_temp2;?>);

var options = {

vAxis: {

title: 'Presion'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div2'));

chart.draw(data, options);

}

function drawBasic3() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', '%');

data.addRows(<?php echo $valor\_temp3;?>);

var options = {

vAxis: {

title: 'Humedad'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div3'));

chart.draw(data, options);

}

function drawBasic4() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', 'particulas');

data.addRows(<?php echo $valor\_temp4;?>);

var options = {

vAxis: {

title: 'Particulas 0.5 um'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div4'));

chart.draw(data, options);

}

function drawBasic5() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', 'particulas');

data.addRows(<?php echo $valor\_temp5;?>);

var options = {

vAxis: {

title: 'Particulas 5 um'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div5'));

chart.draw(data, options);

}

function drawBasic6() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', 'g');

data.addRows(<?php echo $valor\_temp6;?>);

var options = {

vAxis: {

title: 'Peso'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div6'));

chart.draw(data, options);

}

function drawBasic7() {

var data = new google.visualization.DataTable();

data.addColumn('string', 'Time of Day');

data.addColumn('number', 'ºC');

data.addRows(<?php echo $valor\_temp7;?>);

var options = {

vAxis: {

title: 'Temperatura'

}

};

var chart = new google.visualization.LineChart(document.getElementById('chart\_div7'));

chart.draw(data, options);

}

</script>