Índice general

III

Índice general

Índice de figuras

v

Índice de figuras

3.3. Desarrollo del firmware

Como el soporte y la expansión del sistema quedará a cargo del cliente, se optó por desarrollar el firmware en C++ mediante la aplicación Arduino IDE. Esta elección se fundamenta en la baja curva de aprendizaje de la herramienta, la amplia disponibilidad de librerías y ejemplos para el uso de componentes, además de contar con una vasta comunidad de entusiastas de IoT en Internet.

En líneas generales, el firmware de todos los módulos respetan una misma estructura de tres bloques:

1. Inicio: corresponde al encendido del chip.
2. Setup(): función que se ejecuta una única vez al comienzo del programa y se utiliza para inicializar las variables, configurar los pines de entrada y salida, y establecer las comunicaciones necesarias, tales como la velocidad del puerto serial o la conexión a la red Wi-Fi.
3. Loop(): función que se ejecuta continuamente en un bucle que, por lo general, solo se interrumpe al apagar el dispositivo. En esta sección se encuentra el código principal del firmware.

Adicionalmente, el código puede contener variables, funciones y bibliotecas.

Para facilitar la conexión a la aplicación ThingsBoard se utilizó el kit de desarrollo Arduino ThingsBoard SDK [44] que, entre otras funciones, provee mecanismos para el manejo de los protocolos MQTT y RPC a la vez que permite la actualización OTA [45] del firmware.

3.3.1. Módulos sensores de humedad del suelo

En este caso, debido a que ThingsBoard no permite ajustar los períodos de retención de las colas de MQTT para soportar configuraciones prolongadas de *deep sleep*, se resolvió utilizar el protocolo HTTP para las comunicaciones entre el módulo y la aplicación central.

Luego del inicio, el programa realiza una llamada HTTP para obtener el valor del tiempo de hibernación. A continuación lleva a cabo la lectura de los sensores según el caso:

- Sensor simple: lee el valor del pin de conversión analógica a digital (ADC).
- Sensor doble: como el chip ESP8266 posee un único pin ADC compartido por ambas sondas, se procede al energizado secuencial de los sensores al activar la salida GPIO correspondiente, como se mostró en la figura 3.3b.

Para que el módulo pueda reportar un medición que represente la humedad del suelo, es necesario llevar a cabo una conversión del valor leído en el pin ADC. A fin de realizar esta calibración, se utilizó una variante respecto del método descrito por Joshua Hrisko [46] que consiste en realizar mediciones en suelo seco para luego ir agregando cantidades precisas de agua sobre las que se evalúa la diferencia de potencial observada en cada caso. Del proceso resulta una expresión que permite estimar el contenido volumétrico de agua presente.

Una vez obtenido el o los valores, se los reporta a la aplicación por medio una llamada HTTP POST y se da inicio al período de *deep sleep* durante el lapso de tiempo establecido.

3.3. Desarrollo del firmware 23

3.3. Desarrollo del firmware

Como el soporte y la expansión del sistema quedará a cargo del cliente, se optó por desarrollar el firmware en C++ mediante la aplicación Arduino IDE. Esta elección se fundamenta en la baja curva de aprendizaje de la herramienta, la amplia disponibilidad de librerías y ejemplos para el uso de componentes, además de contar con una vasta comunidad de entusiastas de IoT en Internet.

En líneas generales, un programa escrito mediante esta herramienta respeta una estructura de dos bloques:

1. Setup: función que se ejecuta una única vez al comienzo del programa y se utiliza para inicializar las variables, configurar los pines de entrada y salida y establecer las comunicaciones necesarias, tales como la velocidad del puerto serial o la conexión a la red Wi-Fi.
2. Loop: función que se ejecuta continuamente en un bucle que, por lo general, solo se interrumpe al apagar el dispositivo. En esta sección se encuentra el código principal del firmware.

Adicionalmente, el código puede contener variables, funciones y bibliotecas.

Para facilitar la conexión a la aplicación ThingsBoard se utilizó el kit de desarrollo Arduino ThingsBoard SDK [44] que, entre otras funciones, provee mecanismos para el manejo de los protocolos MQTT y RPC, a la vez que permite la actualización OTA [45] del firmware.

Desde el mismo Arduino IDE se transfiere el código generado a la memoria no volátil (*flash*) del dispositivo. Al iniciar (o reiniciar) el microcontrolador, el programa se carga desde la memoria *flash* a la memoria RAM y comienza su ejecución.

3.3.1. Módulos sensores de humedad del suelo

En este caso, debido a que ThingsBoard no permite ajustar los períodos de retención de las colas de MQTT para soportar configuraciones prolongadas de *deep sleep*, se resolvió utilizar el protocolo HTTP para las comunicaciones entre el módulo y la aplicación central.

Luego del inicio, el programa realiza una llamada HTTP para obtener el valor del tiempo de hibernación y a continuación lleva a cabo la lectura de los sensores según el caso:

- Sensor simple: lee el valor del pin de conversión analógica a digital (ADC).
- Sensor doble: como el chip ESP8266 posee un único pin ADC compartido por ambas sondas, se procede al energizado secuencial de los sensores al activar la salida GPIO correspondiente, como se mostró en la figura 3.3b.

Para que el módulo pueda reportar un medición que represente la humedad del suelo, es necesario llevar a cabo una conversión del valor leído en el pin ADC. La relación entre cuentas valor obtenido y humedad se obtiene con una variante respecto del método descrito por Joshua Hrisko [46] que consiste en realizar mediciones en suelo seco para luego ir agregando cantidades precisas de agua sobre las que se evalúa la diferencia de potencial observada en cada caso. Del proceso resulta una expresión que permite estimar el contenido volumétrico de agua presente.

El flujo de ejecución del código para un módulo doble se visualiza en la figura 3.11.

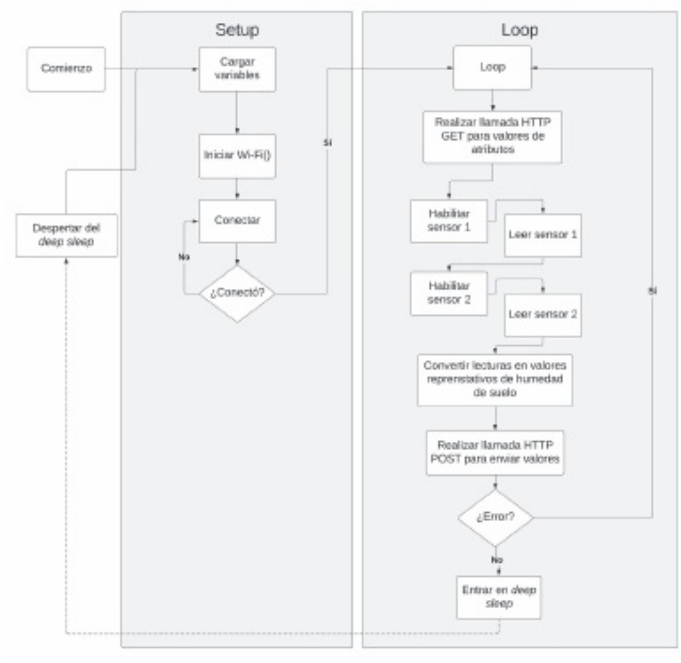


FIGURA 3.11. Diagrama de flujo del firmware de los módulos sensores de humedad del suelo.

3.3.2. Módulo controlador del riego

Es el módulo de mayor complejidad, tal como se refleja en las figuras 3.12, 3.13 y 3.14.

La ejecución inicia con la carga de variables, entre las que se destacan:

- Tiempo de riego: define la duración de encendido de la bomba en segundos.
- Bomba lista: variable lógica que indica el pedido de encendido de la bomba.
- Estado de válvulas: controlan las GPIOs de las válvulas.
- Estado de bomba: controla la GPIO de la bomba.

Luego de la conexión a la red Wi-Fi, el firmware se registra con la aplicación y se suscribe a los *getter* y *setter topics* para reportar los estados a la aplicación o recibir comandos.

Una vez obtenido el o los valores, se los reporta a la aplicación por medio de una llamada HTTP POST y se da inicio al período de *deep sleep* durante el lapso de tiempo establecido.

El flujo de ejecución del código para un módulo doble se visualiza en la figura 3.11.

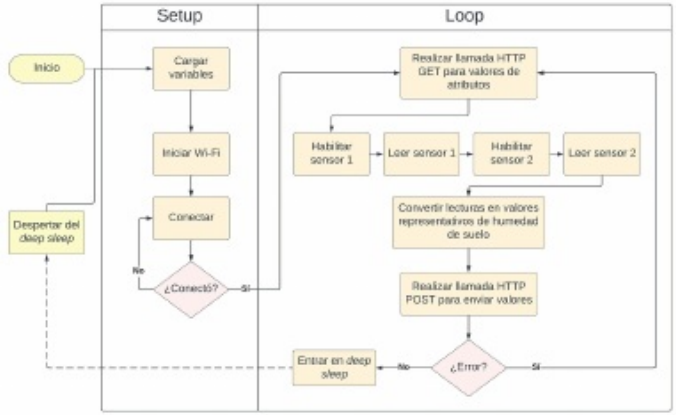


FIGURA 3.11. Diagrama de flujo del firmware de los módulos sensores de humedad del suelo.

3.3.2. Módulo controlador del riego

Es el módulo de mayor complejidad, tal como se refleja en las figuras 3.12, 3.13 y 3.14.

La ejecución inicia con la carga de variables, entre las que se destacan:

- Tiempo de riego: define la duración de encendido de la bomba en segundos.
- Bomba lista: variable lógica que indica el pedido de encendido de la bomba.
- Estado de válvulas: controlan las GPIOs de las válvulas.
- Estado de bomba: controla la GPIO de la bomba.

Luego de la conexión a la red Wi-Fi, el firmware se registra con la aplicación y se suscribe a los *getter* y *setter topics* para reportar los estados a la aplicación o recibir comandos.

En el caso de recibir un mensaje de pedido de información, el código reporta el o los valores de las variables solicitadas en el *topic* de respuesta. Por otro lado, si el mensaje recibido indica realizar un cambio de estado, se pueden presentar las siguientes situaciones:

En el caso de recibir un mensaje de pedido de información, el código reporta el o los valores de las variables solicitadas en el *topic* de respuesta. Por otro lado, si el mensaje recibido indica realizar un cambio de estado, se pueden presentar las siguientes situaciones:

- Pedido de reconfigurar el tiempo de riego: el código actualiza la variable y reinicia el bucle.
- Pedido de apertura o cierre de válvula:
  - Apertura: procede a abrir la válvula seleccionada y comprueba si la bomba se encuentra en estado de pedido de encendido (bomba lista), en cuyo caso ordena el encendido.
  - Cierre: verifica si otras válvulas están abiertas y en caso afirmativo, procede con el cierre. En caso contrario, si es la única, ordena el apagado de la bomba y luego el cierre de la válvula.
- Pedido de encendido o apagado de la bomba:
  - Encendido: en el caso de haber al menos una válvula abierta, procede a encender la bomba durante el tiempo que indique la variable de duración de riego. Una vez finalizado, procede ordenadamente al apagado de la bomba y al cierre de las válvulas. De no haber válvulas abiertas, se configura la variable de bomba lista en positivo, indicando que el riego fue pedido pero aún no están dadas las condiciones para habilitarlo.
  - Apagado: en el caso de requerir una interrupción en el riego, se ordena el apagado de la bomba y el cierre de las válvulas.

3.3. Desarrollo del firmware 25

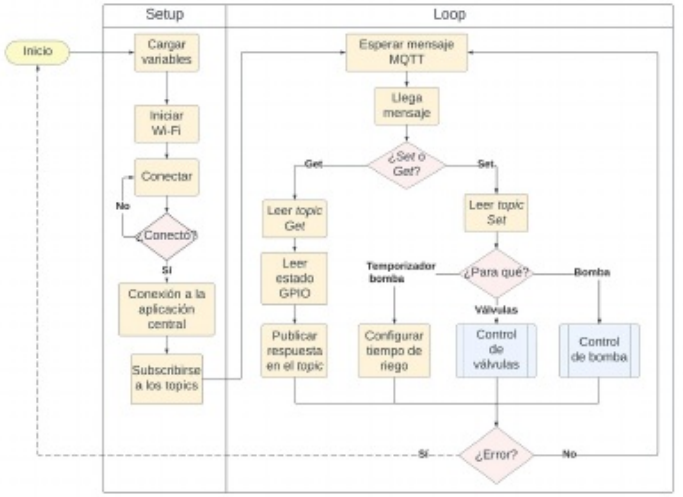


FIGURA 3.12. Diagrama de flujo del firmware del módulo de control de riego.

- Pedido de reconfigurar el tiempo de riego: el código actualiza la variable y reinicia el bucle.
- Pedido de apertura o cierre de válvula:
  - Apertura: procede a abrir la válvula seleccionada y comprueba si la bomba se encuentra en estado de pedido de encendido (bomba lista), en cuyo caso ordena el encendido.
  - Cierre: verifica si otras válvulas están abiertas y en caso afirmativo procede con el cierre. En caso contrario, si es la única, ordena el apagado de la bomba y luego el cierre de la válvula.
- Pedido de encendido o apagado de la bomba:
  - Encendido: en el caso de haber al menos una válvula abierta, procede a encender la bomba durante el tiempo que indique la variable de duración de riego. Una vez finalizado, procede ordenadamente al apagado de la bomba y al cierre de las válvulas. De no haber válvulas abiertas, se configura la variable de bomba lista en positivo, indicando que el riego fue pedido pero aún no están dadas las condiciones para habilitarlo.
  - Apagado: en el caso de requerir una interrupción en el riego, se ordena el apagado de la bomba y el cierre de las válvulas.

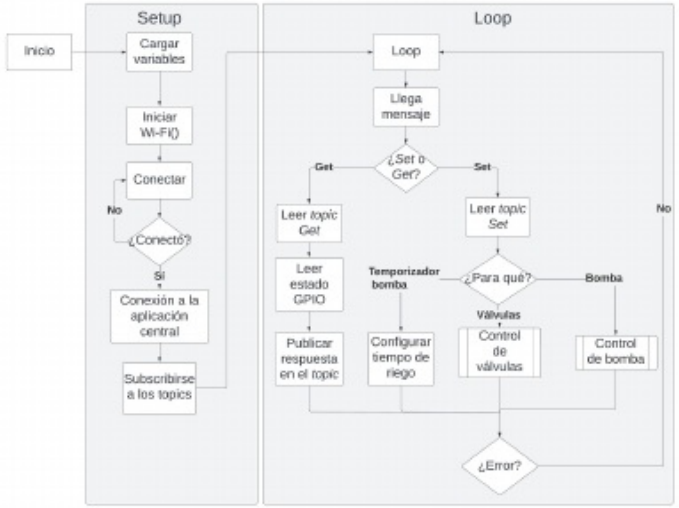


FIGURA 3.12. Diagrama de flujo del firmware del módulo de control de riego.

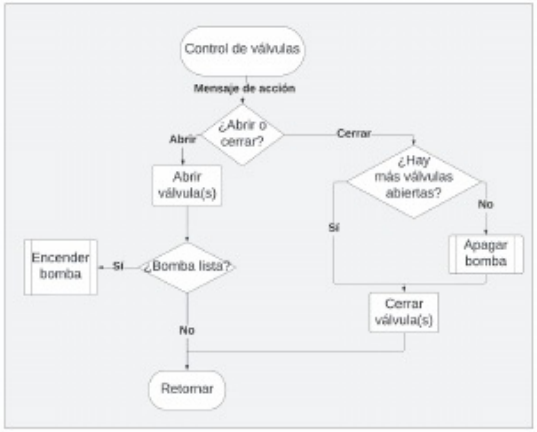


FIGURA 3.13. Diagrama de flujo del firmware del módulo de control de riego - control de válvulas.

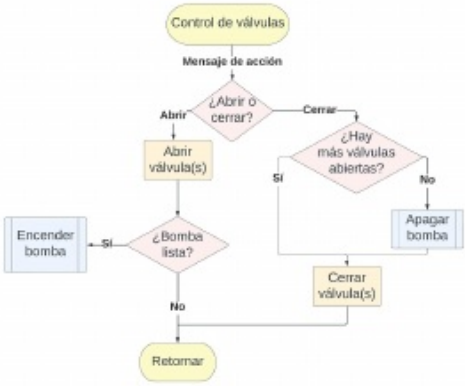


FIGURA 3.13. Diagrama de flujo del firmware del módulo de control de riego - control de válvulas.

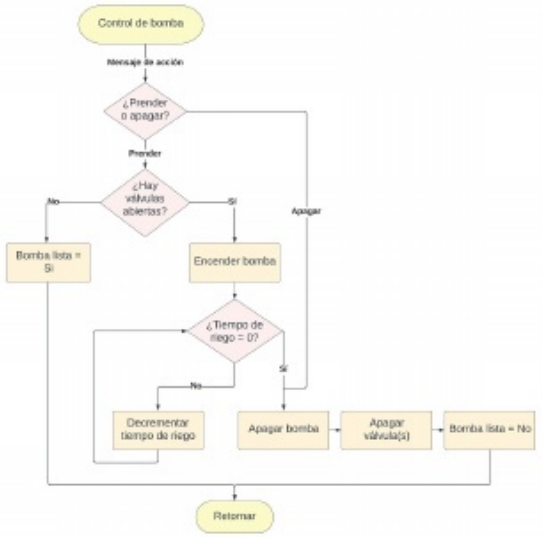


FIGURA 3.14. Diagrama de flujo del firmware del módulo de control de riego - control de bomba.

3.3.3. Módulo sensor de temperatura y humedad

La figura 3.15 describe el funcionamiento de este módulo. Luego del ciclo de inicio y del setup, el código se encarga de obtener las medidas de temperatura y

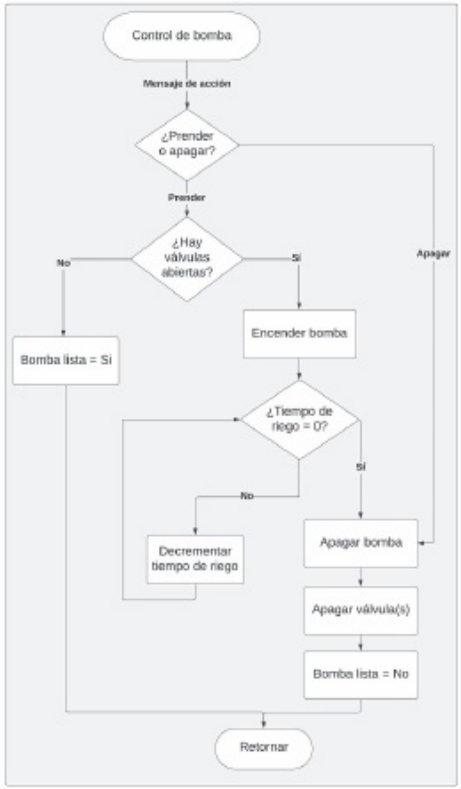


FIGURA 3.14. Diagrama de flujo del firmware del módulo de control de riego - control de bomba.

3.3.3. Módulo sensor de temperatura y humedad

La figura 3.15 describe el funcionamiento de este módulo. Luego del ciclo de inicio y del setup(), el código se encarga de obtener las medidas de temperatura y humedad, valores que despliega en la pantalla del módulo. Periódicamente estos valores se envían a la aplicación de acuerdo con un contador de ciclos.

humedad, valores que despliega en la pantalla del módulo. Periódicamente estos valores se envían a la aplicación de acuerdo con un contador de ciclos.

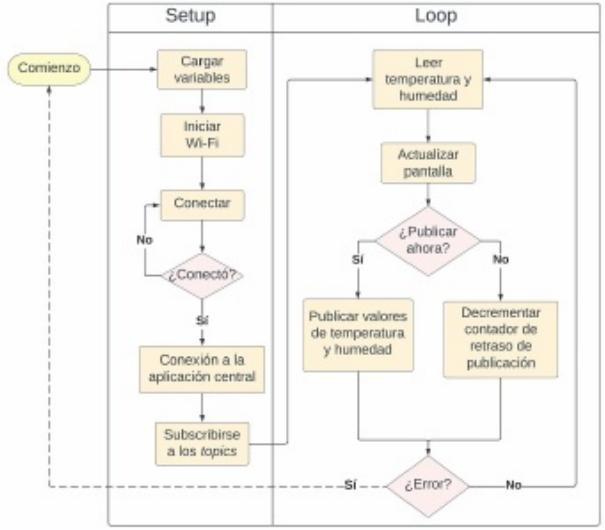


FIGURA 3.15. Diagrama de flujo del firmware del módulo sensor de temperatura y humedad.

3.3.4. Módulo de control de clima

Es el encargado del encendido y apagado de los ventiladores para controlar el clima. Luego del inicio, el programa se conecta a la red, se subscribe en los *topics* correspondientes e ingresa en la función de loop a la espera de mensajes. En caso de recibir un pedido de reporte, responde con el estado del pin GPIO asociado con el ventilador. Si recibe una orden de encendido o apagado, realiza la acción correspondiente y retorna al inicio del bucle.

En la figura 3.16 se describe el flujo del código para este módulo.

3.4. Selección y configuración del software

La propiedad del cliente se encuentra en una zona rural con acceso a internet limitado, por tal motivo se implementó una solución local basada en ThingsBoard Community Edition. Esta plataforma tiene bajos requerimientos de hardware, buen diseño de seguridad y ofrece amplias posibilidades de expansión.

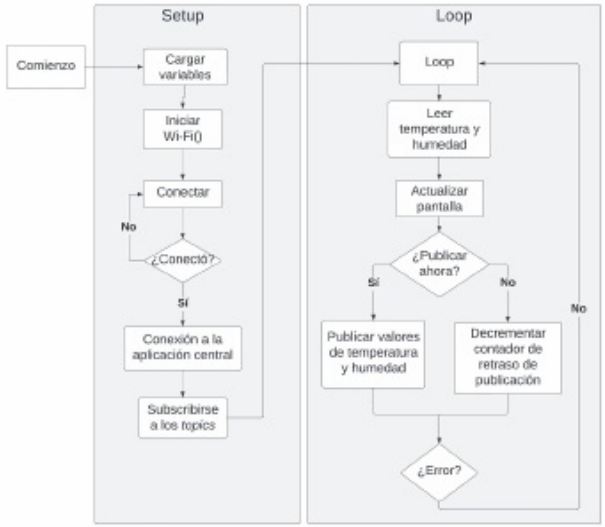


FIGURA 3.15. Diagrama de flujo del firmware del módulo sensor de temperatura y humedad.

3.3.4. Módulo de control de clima

Es el encargado del encendido y apagado de los ventiladores para controlar el clima. Luego del inicio, el programa se conecta a la red, se subscribe en los *topics* correspondientes e ingresa en la función de loop() a la espera de mensajes. En caso de recibir un pedido de reporte, responde con el estado del pin GPIO asociado con el ventilador. Si recibe una orden de encendido o apagado, realiza la acción correspondiente y retorna al inicio del bucle.

En la figura 3.16 se describe el flujo del código para este módulo.

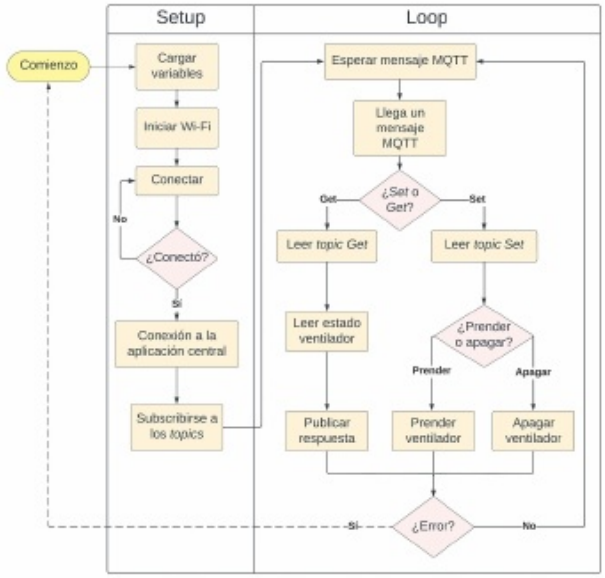


FIGURA 3.16. Diagrama de flujo del firmware del módulo de control de clima.

3.4.1. Servidor de la aplicación

La instalación se realizó sobre una Raspberry Pi 4B de 8 GB de memoria RAM, con sistema operativo Linux Raspbian versión 11, todo instalado sobre una tarjeta de memoria SD de 64 GB de capacidad.

3.4.2. Configuración de la aplicación

La instalación de la aplicación se realizó conforme a las instrucciones provistas por el desarrollador en su página oficial [47]. ThingsBoard requiere de una base de datos para almacenar los valores de telemetría y atributos. Para una carga de hasta 5000 mensajes por segundo se recomienda el uso de PostgreSQL [48]. Dado que la cantidad de mensajes enviados es de aproximadamente 4 por minuto en el prototipo y no se espera un crecimiento desmedido en el pasaje a producción, la capacidad de memoria y procesamiento empleados resulta suficiente para la instalación conjunta de la base de datos y la aplicación.

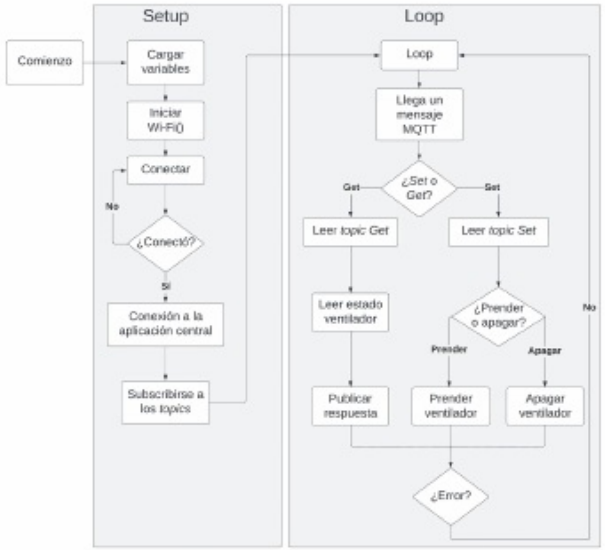


FIGURA 3.16. Diagrama de flujo del firmware del módulo de control de clima.

3.4. Selección y configuración del software

La propiedad del cliente se encuentra en una zona rural con acceso a Internet limitado, por tal motivo se implementó de una solución local basada en ThingsBoard Community Edition. Esta plataforma tiene bajos requerimientos de hardware, buen diseño de seguridad y ofrece amplias posibilidades de expansión.

3.4.1. Servidor de la aplicación

La instalación se realizó sobre una Raspberry Pi 4B de 8 GB de memoria RAM, con sistema operativo Linux Raspbian versión 11, todo instalado sobre una tarjeta de memoria SD de 64 GB de capacidad.

Por simplicidad, durante la construcción del prototipo se utilizó una red única, pero es intención que el modelo final emplee la placa Ethernet de la Raspberry PI para el acceso a Internet y la inalámbrica para establecer una red de tipo *ad hoc* con los módulos.

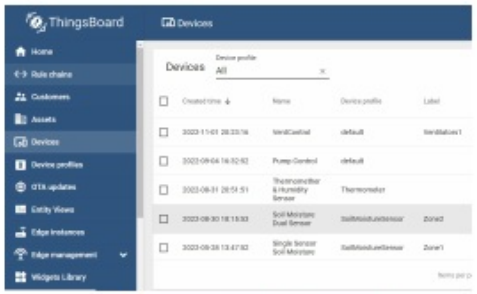
3.4.2. Configuración de la aplicación

La instalación de la aplicación se realizó conforme a las instrucciones provistas por el desarrollador en su página oficial [47]. ThingsBoard requiere de una base de datos para almacenar los valores de las telemetrías y atributos. Para una carga de hasta 5000 mensajes por segundo se recomienda el uso de PostgreSQL [48].

3.4.3. Manejo de dispositivos

La baja cardinalidad de módulos creados para el proyecto permitió emplear la interfaz web para gestionarlos en lugar de la opción programática mediante la API REST. Al no contar con una CA para la generación de certificados, la autenticación de los dispositivos se resolvió mediante el empleo de tokens.

La figura 3.17a muestra la pantalla con los dispositivos creados, mientras que en la figura 3.17b se observan las características de un sensor de humedad del suelo doble. Desde allí se puede navegar a los atributos del dispositivo, sus alarmas y eventos e incluso desvincularlo.



(A) Listado de dispositivos del proyecto.

(B) Configuración de las características de un sensor.

FIGURA 3.17. Pantallas de manejo de dispositivos creados en el proyecto.

3.4.4. Reglas de automatización

ThingsBoard posee un motor de reglas [49] para la creación de flujos de trabajo basados en eventos. Una de sus principales ventajas es la de permitir el desarrollo de automatizaciones bajo la metodología *low-code/no-code* [50].

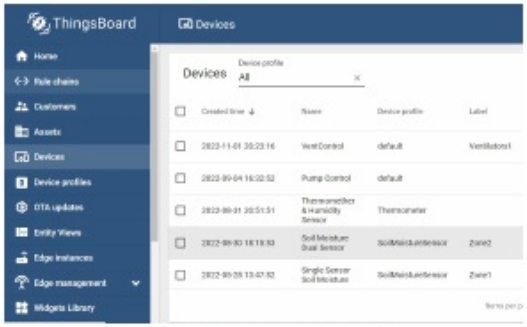
Las reglas se componen principalmente de:

Dado que la cantidad de mensajes enviados es de aproximadamente 4 por minuto en el prototipo y no se espera un crecimiento desmedido en el pasaje a producción, la capacidad de memoria y procesamiento empleados resulta suficiente para la instalación conjunta de la base de datos y la aplicación.

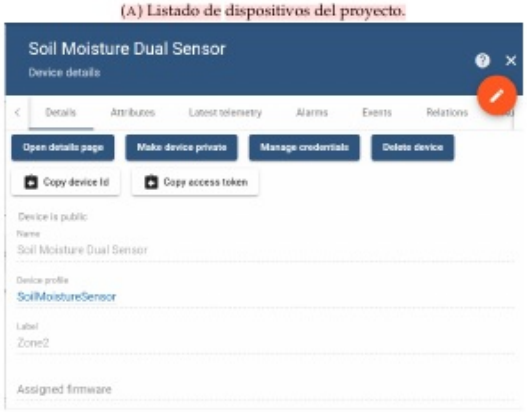
3.4.3. Manejo de dispositivos

La baja cardinalidad de módulos creados para el proyecto permitió emplear la interfaz web para gestionarlos en lugar de la opción programática mediante la API REST. Al no contar con una CA para la generación de certificados, la autenticación de los dispositivos se resolvió mediante el empleo de tokens.

La figura 3.17a muestra la pantalla con los dispositivos creados, mientras que en la figura 3.17b se observan las características de un sensor de humedad del suelo doble. Desde allí se puede navegar a los atributos del dispositivo, sus alarmas y eventos, e incluso desvincularlo.



(A) Listado de dispositivos del proyecto.



(B) Configuración de las características de un sensor.

FIGURA 3.17. Pantallas de manejo de dispositivos creados en el proyecto.

- Mensaje: cualquier evento que ingrese, por ejemplo datos provenientes de un dispositivo, un pedido a través de la API REST (HTTP) o un requerimiento RPC.
- Nodo de regla: función para el filtrado, transformación o acción que se ejecuta sobre un mensaje entrante y que puede producir mensajes salientes.
- Conexión de nodo de regla: los nodos pueden vincularse entre sí mediante relaciones, de manera que la salida de uno es enviada a los próximos nodos conectados. Cada relación tiene una etiqueta que identifica su significado lógico, generalmente del tipo "éxito" o "fracaso".
- Cadena de reglas: grupo lógico de nodos de reglas y sus conexiones.

El procesamiento de mensajes ofrece tres resultados posibles: éxito, error y *timeout* (tiempo excedido). Se obtiene un resultado exitoso cuando el último nodo de regla en la cadena completa correctamente el procesamiento. Se obtiene error cuando en el nodo se produce un fallo al operar sobre un mensaje y no existe un mecanismo para manejarlo. Por último, se establece un *timeout* cuando el tiempo total de procesamiento supera el umbral configurado.

Para el proyecto se desarrollaron una o más reglas por subsistema, de manera de conectar a las unidades de control con los respectivos sensores que informan sobre el estado del invernadero y para el manejo de alertas a los usuarios. La figura 3.18 muestra, a manera de ejemplo, un fragmento de la regla creada para el control de riego a partir de los mensajes recibidos desde los sensores de humedad del suelo. Allí se visualizan los nodos en diferentes colores de acuerdo a su función y las cadenas como las flechas que los unen.

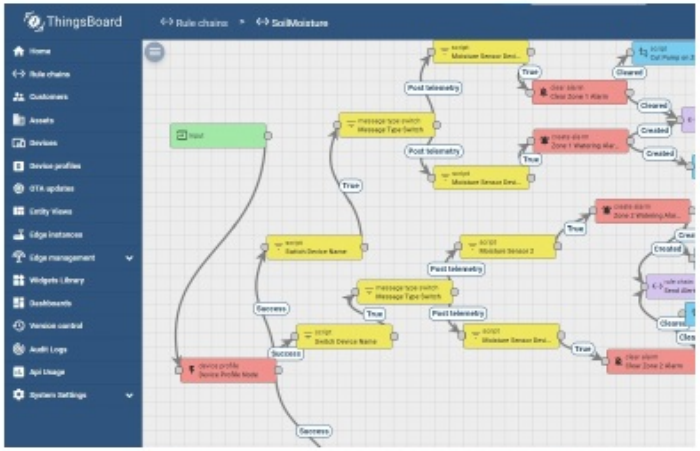


FIGURA 3.18. Fragmento de regla de automatización para el control de riego.

3.4.4. Reglas de automatización

ThingsBoard posee un motor de reglas [49] para la creación de flujos de trabajo basados en eventos. Una de sus principales ventajas es la de permitir el desarrollo de automatizaciones bajo la metodología *low-code/no-code* [50].

Las reglas se componen principalmente de:

- Mensaje: cualquier evento que ingrese, por ejemplo datos provenientes de un dispositivo, un pedido a través de la API REST (HTTP) o un requerimiento RPC.
- Nodo de regla: función para el filtrado, transformación o acción que se ejecuta sobre un mensaje entrante, y que puede producir mensajes salientes.
- Conexión de nodo de regla: los nodos pueden vincularse entre sí mediante relaciones, de manera que la salida de uno es enviada a los próximos nodos conectados. Cada relación tiene una etiqueta que identifica su significado lógico, generalmente del tipo "éxito" o "fracaso".
- Cadena de reglas: grupo lógico de nodos de reglas y sus conexiones.

El procesamiento de mensajes ofrece tres resultados posibles: éxito, error y *timeout* (tiempo excedido). Se obtiene un resultado exitoso cuando el último nodo de regla en la cadena completa correctamente el procesamiento. Se obtiene error cuando en el nodo se produce un fallo al operar sobre un mensaje y no existe un mecanismo para manejarlo. Por último, se establece un *timeout* cuando el tiempo total de procesamiento supera el umbral configurado.

Para el proyecto se desarrollaron una o más reglas por subsistema, de manera de conectar a las unidades de control con los respectivos sensores que informan sobre el estado del invernadero, y para el manejo de alertas a los usuarios. La figura 3.18 muestra, a manera de ejemplo, un fragmento de la regla creada para el control de riego a partir de los mensajes recibidos desde los sensores de humedad del suelo. Allí se visualizan los nodos en diferentes colores de acuerdo a su función y las cadenas como las flechas que los unen.

3.4.5. Interfaz de manejo remoto

Para permitir que los usuarios conozcan el estado del invernadero vía Internet, se desarrolló una aplicación en Python que se integra al *bot* de Telegram mediante la librería SDK python-telegram-bot. Este programa fue alojado en la misma Raspberry Pi donde corre ThingsBoard.

Se trata de un bucle que contacta al *bot* de Telegram periódicamente en busca de mensajes nuevos. Al detectar la llegada de alguno, el programa identifica si se corresponde con alguno de los comandos definidos y de ser así, envía una solicitud HTTP a la aplicación central. En base a la respuesta recibida, la interfaz confecciona un mensaje que es enviado al usuario por medio del *bot*. La figura 3.19a muestra un ejemplo del mensaje de estado del invernadero.

Además, el servicio de *bot* se utilizó para el envío de alertas a usuarios. Para ello se crearon nodos en las reglas de automatización de la aplicación central que, ante ciertos eventos, disparan una solicitud HTTP POST al *bot* con el mensaje de alerta como se ilustra en la figura 3.19b.

3.4.5. Interfaz de manejo remoto

Para permitir que los usuarios conozcan el estado del invernadero vía Internet, se desarrolló una aplicación en Python que se integra al *bot* de Telegram mediante la librería SDK python-telegram-bot. Este programa fue alojado en la misma Raspberry Pi donde corre ThingsBoard.

Se trata de un bucle que contacta al *bot* de Telegram periódicamente en busca de mensajes nuevos. Al detectar la llegada de alguno, el programa identifica si se corresponde con alguno de los comandos definidos y de ser así, envía una solicitud HTTP a la aplicación central. En base a la respuesta recibida, la interfaz confecciona un mensaje que es enviado al usuario por medio del *bot*. La figura 3.19a muestra un ejemplo de mensaje de estado del invernadero.

Además, el servicio de *bot* se utilizó para el envío de alertas a usuarios. Para ello se crearon nodos en las reglas de automatización de la aplicación central que, ante ciertos eventos, disparan una solicitud HTTP POST al *bot* con el mensaje de alerta como se ilustra en la figura 3.19b.

(A) Reporte de estado.

(B) Mensaje de alerta en Telegram.

FIGURA 3.19. Uso del *bot* de Telegram .

3.5. Ciberseguridad del sistema

Por tratarse de un proyecto hogareño, el cliente no cuenta con la infraestructura de autenticación ni con una CA para proveer certificados a los dispositivos o a la aplicación central. Por este motivo se decidió utilizar un certificado autofirmado para encriptar las comunicaciones entre los módulos y ThingsBoard. Sin embargo, el uso de este tipo de certificados tiene la desventaja de carecer del respaldo de una entidad que garantice su autenticidad, por lo que es necesario realizar configuraciones explícitas para forzar la confianza.

Durante el proceso de desarrollo se siguieron las buenas prácticas para el manejo de contraseñas y tokens de autenticación, evitando divulgar estos valores en los repositorios públicos.

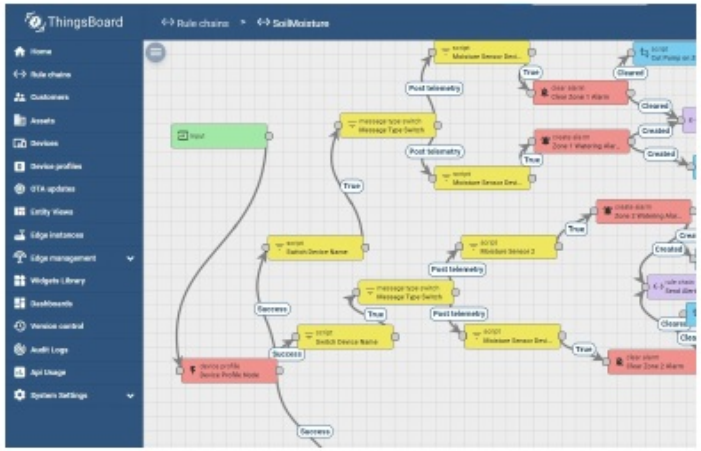
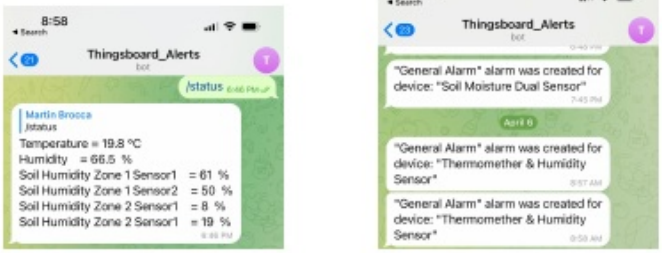


FIGURA 3.18. Fragmento de regla de automatización para el control de riego.



(A) Reporte de estado. (B) Mensaje de alerta en Telegram.

FIGURA 3.19. Uso del *bot* de Telegram .

3.5. Ciberseguridad del sistema

Por tratarse de un proyecto hogareño, el cliente no cuenta con la infraestructura de autenticación ni con una CA para proveer certificados a los dispositivos o a la aplicación central. Por este motivo se decidió utilizar un certificado autofirmado para encriptar las comunicaciones entre los módulos y ThingsBoard. Sin embargo, el uso de este tipo de certificados tiene la desventaja de carecer del respaldo de una entidad que garantice su autenticidad, por lo que es necesario realizar configuraciones explícitas para forzar la confianza.

Durante el proceso de desarrollo se siguieron las buenas prácticas para el manejo de contraseñas y tokens de autenticación, evitando divulgar estos valores en los repositorios públicos.

Capítulo 4

Ensayos y resultados

En este capítulo se explica la metodología de pruebas aplicada tanto a los componentes individuales como al sistema implementado, finalizando con una comparativa con el estado del arte.

4.1. Banco de pruebas

4.2. Pruebas unitarias

4.3. Pruebas de sistema

4.4. Comparativa con el estado de arte

Capítulo 4

Ensayos y resultados

En este capítulo se explica la metodología de pruebas aplicada tanto a los componentes individuales como al sistema implementado para finalizar con una comparación con el estado del arte.

4.1. Banco de pruebas

La verificación del correcto funcionamiento de los módulos que componen el sistema se realizó mediante una maqueta que se muestra en la figura 4.1 y representa en escala reducida al invernadero del cliente.

El modelo cuenta con una bomba de agua conectada a tres válvulas para alimentar a los circuitos de riego. Se utilizaron mangueras y conexiones neumáticas de aluminio de acople rápido y el conjunto armado puede verse en la figura 4.2a.

El ensamblado se muestra en las figuras 4.2b, 4.2c y 4.2d donde se observa la implementación de dos circuitos de riego independientes. También se configuró un tercer circuito cerrado para pruebas de accionamiento de bomba y válvula a fin de evitar desperdicios durante las fases de calibración.

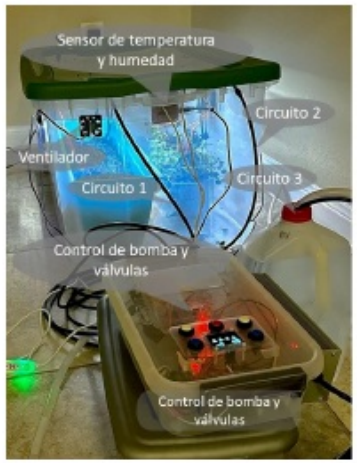


FIGURA 4.1. Modelo completo del invernadero.

Además se incorporaron a la maqueta dos módulos sensores de humedad del suelo, uno de temperatura y humedad y un actuador para el encendido de los ventiladores.

Para las pruebas manuales de accionamiento de los diferentes sistemas se empleó una computadora portátil y un celular para las pruebas de acceso concurrente a la aplicación central.

(A) Conjunto bomba y válvulas.

(B) Armado de mangueras de riego.

(C) Pruebas de riego.

(D) Ensamble general.

FIGURA 4.2. Modelo de pruebas del invernadero.

Capítulo 5

Conclusiones

En este capítulo se muestran las conclusiones sobre el trabajo realizado. A su vez se presentan algunas modificaciones o mejoras como posible trabajo futuro

5.1. Resultados obtenidos

5.2. Trabajo futuro

Acá se indica cómo se podría continuar el trabajo más adelante.

4.2. Pruebas unitarias

Aquí se describen las pruebas individuales efectuadas sobre los módulos sensores y actuadores y la verificación del cumplimiento de requerimientos planteados

4.3. Pruebas de sistema

Aquí se describen las pruebas de sistema en donde interactúan dos o más componentes

4.4. Comparativa con el estado de arte

Aquí se describe la comparación entre el modelo armado y lo informado en el capítulo 1

Bibliografía

[1] Scott Eldridge y Lyman Chapin Karen Rose. «The Internet of Things: An Overview». En: (2015).

[2] Krishna Nemali. «History of Controlled Environment Horticulture: Greenhouses». En: *HortScience* 57.2 (2022), págs. 239 -246. DOI: 10.21273/HORTSCI16160-21. URL: https://journals.ashs.org/hortsci/view/journals/hortsci/57/2/article-p239.xml.

[3] Chrysanthos Maraveas y Thomas Bartzanas. «Aplicación de internet de las cosas (IoT) para entornos de invernadero optimizados». En: *Magna Scientia UCEVA* 2 (dic. de 2022), págs. 253-268. DOI: 10.54502/msuceva.v2n2a11.

[4] John W. Bartok. *Greenhouses for Homeowners and Gardeners*. Natural Resource, Agriculture, y Engineering Service (NRAES), 2000-06.

[5] Intel. *¿Qué son los invernaderos inteligentes?* https://agrofacto.com/invernaderos-inteligentes/m. 2022. (Visitado 20-03-2023).

[6] Rakiba Rayhana, Gaozhi Xiao y Zheng Liu. «Internet of Things Empowered Smart Greenhouse Farming». En: *IEEE Journal of Radio Frequency Identification* 4.3 (2020), págs. 195-211. DOI: 10.1109/JRFID.2020.2984391.

[7] Argus Controls. https://arguscontrols.com/. (Visitado 20-03-2023).

[8] Grodan. https://www.grodan.com/. (Visitado 20-03-2023).

[9] Growlink. https://www.growlink.com/. (Visitado 20-03-2023).

[10] Chet Udell y Alan Dennis Lloyd Nackley. https://diggermagazine.com/the-smart-greenhouse/. 2020. (Visitado 20-03-2023).

[11] Arduino. https://www.arduino.cc/. (Visitado 20-03-2023).

[12] OASIS. *MQTT Version 5.0*. Oasis Standard. URL: \url{https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html} (visitado 20-03-2023).

[13] Inc. Sun Microsystems. *RPC: Remote Procedure Call Protocol Specification Version 2*. RFC 1057. RFC Editor, 1988. URL: https://www.rfc-editor.org/rfc/rfc1057.

[14] E. Rescorla T. Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC. 2008. URL: https://www.rfc-editor.org/rfc/rfc5280.

[15] C. Shao et al. *IEEE 802.11 Medium Access Control (MAC) Profile for Control and Provisioning of Wireless Access Points (CAPWAP)*. RFC. URL: \url{https://www.rfc-editor.org/rfc/rfc7494.html} (visitado 20-03-2023).

[16] T. Socolofsky et al. *A TCP/IP Tutorial*. RFC 1180. RFC Editor, 1991. URL: https://www.rfc-editor.org/rfc/rfc1180.

[17] www.survivingwithandroid.com. https://www.survivingwithandroid.com/mqtt-protocol-tutorial/. (Visitado 20-03-2023).

37

Capítulo 5

Conclusiones

En este capítulo se muestran las conclusiones sobre el trabajo realizado. A su vez se presentan algunas modificaciones o mejoras como posible trabajo futuro

5.1. Resultados obtenidos

5.2. Trabajo futuro

Acá se indica cómo se podría continuar el trabajo más adelante.

[18] Patrick Th. Eugster et al. «The Many Faces of Publish/Subscribe». En: *ACM Comput. Surv.* 35.2 (2003), 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: https://doi.org/10.1145/857076.857078.

[19] The Internet Society. *Hypertext Transfer Protocol – HTTP/1.1*. RFC. URL: \url{https://www.ietf.org/rfc/rfc2616.txt} (visitado 20-03-2023).

[20] https://www.oreilly.com/library/view/javaserver-pages-3rd/0596005636/ch02s01.html. (Visitado 20-03-2023).

[21] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Inf. téc. 2000. Cap. Chapter 5: Representational State Transfer (REST).

[22] Jasenka Dizdarević et al. «A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration». En: *ACM Comput. Surv.* 51.6 (ene. de 2019). ISSN: 0360-0300. DOI: 10.1145/3292674. URL: https://doi.org/10.1145/3292674.

[23] Dimitrios Glaroudis, Athanasios Iossifides y Periklis Chatzimisios. «Survey, comparison and research challenges of IoT application protocols for smart farming». En: *Computer Networks* 168 (2020), pág. 107037. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.107037. URL: https://www.sciencedirect.com/science/article/pii/S1389128619306942.

[24] J. Postel. *Transmission Control Protocol*. RFC 793. Internet Engineering Task Force, 1981, pág. 85. URL: http://www.rfc-editor.org/rfc/rfc793.txt.

[25] Internet Society. «TLS Basics». En: (). URL: https://www.internetsociety.org/deploy360/tls/basics/.

[26] D. Cooper et. al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC. 2008. URL: https://www.rfc-editor.org/rfc/rfc5280.

[27] Shanna L. «Comparative Analysis of Infrastructure and Ad-Hoc Wireless Networks». En: *ITM Web of Conferences 25, 01009* (2019).

[28] https://es.wikipedia.org/wiki/Raspberry_Pi. (Visitado 20-03-2023).

[29] https://opensource.com/resources/raspberry-pi. (Visitado 20-03-2023).

[30] *Raspberry Pi Computer Model B*. Datasheet. Raspberry Pi. Ene. de 2021. URL: \url{https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf}.

[31] *ESP32 Series*. Datasheet. V4.2. Espressif. Ene. de 2023.

[32] *ESP8266EX*. Datasheet. V6.9. Espressif. Feb. de 2023.

[33] *Digital output relative humidity and temperature sensor/module*. DHT22. Aosong Electronics Co.,Ltd. URL: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[34] https://smartbitbn.com/product/soil-humidity-sensor/. (Visitado 20-03-2023).

[35] https://www.amazon.com/dp/B07BW21Z5M/. (Visitado 20-03-2023).

[36] https://es.wikipedia.org/wiki/RelÅľ. (Visitado 20-03-2023).

[37] https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/. (Visitado 20-03-2023).

[38] Apache Software Foundation. *Apache License 2.0*. Contract. Apache Software Foundation, 2004. URL: https://www.apache.org/licenses/LICENSE-2.0.

[39] *Soporte de Arduino IDE para hardware de terceros*. URL: https://github.com/per1234/ino-hardware-package-list/blob/master/ino-hardware-package-list.tsv.

[40] *Arduino Integrated Development Environment (IDE) v1*. URL: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics.

[41] *What is arduino software (IDE), and how use it ?* URL: https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/.

[42] *Telegram - Preguntas Frecuentes*. URL: https://telegram.org/faq?setln=es#p-que-es-telegram-que-puedo-hacer-aqui.

[43] https://www.iec.ch/ip-ratings. (Visitado 20-03-2023).

[44] Thingsboard. *Thingsboard Arduido SDK*. Visitado el 2023-03-20. 2023. URL: https://github.com/thingsboard/thingsboard-arduino-sdk.

[45] Jan Bauwens et al. «Over-the-Air Software Updates in the Internet of Things: An Overview of Key Principles». En: *IEEE Communications Magazine* 58.2 (2020), págs. 35-41. DOI: 10.1109/MCOM.001.1900125.

[46] Joshua Hrisko. https://makersportal.com/blog/2020/5/26/capacitive-soil-moisture-calibration-with-arduino. 2020. (Visitado 20-03-2023).

[47] Thingsboard. *Thingsboard Arduido SDK*. Visitado el 2023-04-09. URL: https://thingsboard.io/docs/user-guide/install/rpi/.

[48] PostgreSQL. *PostgreSQL - About*. Visitado el 2023-03-20. 2023. URL: https://www.postgresql.org/about/.

[49] ThingsBoard. *Rule Engine Overview*. Visitado el 2023-03-21. 2023. URL: https://thingsboard.io/docs/user-guide/rule-engine-2-0/overview/.

[50] Chris Johannessen y Tom Davenport. *When Low-Code/No-Code Development Works — and When It Doesn't*. Visitado el 2023-04-09. URL: https://hbr.org/2021/06/when-low-code-no-code-development-works-and-when-it-doesnt.

Bibliografía

[1] Scott Eldridge y Lyman Chapin Karen Rose. «The Internet of Things: An Overview». En: (2015).

[2] Krishna Nemali. «History of Controlled Environment Horticulture: Greenhouses». En: *HortScience* 57.2 (2022), págs. 239 -246. DOI: 10.21273/HORTSCI16160-21. URL: https://journals.ashs.org/hortsci/view/journals/hortsci/57/2/article-p239.xml.

[3] Chrysanthos Maraveas y Thomas Bartzanas. «Aplicación de internet de las cosas (IoT) para entornos de invernadero optimizados». En: *Magna Scientia UCEVA* 2 (dic. de 2022), págs. 253-268. DOI: 10.54502/msuceva.v2n2a11.

[4] John W. Bartok. *Greenhouses for Homeowners and Gardeners*. Natural Resource, Agriculture, y Engineering Service (NRAES), 2000-06.

[5] Intel. *¿Qué son los invernaderos inteligentes?* https://agrofacto.com/invernaderos-inteligentes/m. 2022. (Visitado 20-03-2023).

[6] Rakiba Rayhana, Gaozhi Xiao y Zheng Liu. «Internet of Things Empowered Smart Greenhouse Farming». En: *IEEE Journal of Radio Frequency Identification* 4.3 (2020), págs. 195-211. DOI: 10.1109/JRFID.2020.2984391.

[7] Argus Controls. https://arguscontrols.com/. (Visitado 20-03-2023).

[8] Grodan. https://www.grodan.com/. (Visitado 20-03-2023).

[9] Growlink. https://www.growlink.com/. (Visitado 20-03-2023).

[10] Chet Udell y Alan Dennis Lloyd Nackley. https://diggermagazine.com/the-smart-greenhouse/. 2020. (Visitado 20-03-2023).

[11] Arduino. https://www.arduino.cc/. (Visitado 20-03-2023).

[12] OASIS. *MQTT Version 5.0*. Oasis Standard. URL: \url{https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html} (visitado 20-03-2023).

[13] Inc. Sun Microsystems. *RPC: Remote Procedure Call Protocol Specification Version 2*. RFC 1057. RFC Editor, 1988. URL: https://www.rfc-editor.org/rfc/rfc1057.

[14] E. Rescorla T. Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC. 2008. URL: https://www.rfc-editor.org/rfc/rfc5280.

[15] C. Shao et al. *IEEE 802.11 Medium Access Control (MAC) Profile for Control and Provisioning of Wireless Access Points (CAPWAP)*. RFC. URL: \url{https://www.rfc-editor.org/rfc/rfc7494.html} (visitado 20-03-2023).

[16] T. Socolofsky et al. *A TCP/IP Tutorial*. RFC 1180. RFC Editor, 1991. URL: https://www.rfc-editor.org/rfc/rfc1180.

[17] www.survivingwithandroid.com. https://www.survivingwithandroid.com/mqtt-protocol-tutorial/. (Visitado 20-03-2023).