DICIEMBRE 2022

SMART COWORK

PROYECTO INTEGRADOR 2

FACULTAD DE INGENIERÍA - ORT

Integrantes
Pilar Busquets - 251364
Valentina García -251371
Dana Zejerman - 220089

Grupo
N6A (IZ YI IE)
Profesor
Emiliano Espíndola
Juan Pedro Silva

Índice

Integrantes

Pilar Busquets - 251364 - Ingeniería en Electrónica

Valentina García - 251371 - Ingeniería en Electrónica

Dana Zejerman - 220089 - Ingeniería en Electrónica

Descripción del proyecto

Problema a resolver y objetivos planteados

Como fue planteado en el anteproyecto, el problema a resolver con nuestra solución implica la integración y sincronización de distintos subsistemas del funcionamiento de un edificio, enmarcado en el contexto de un espacio de trabajo colaborativo: un cowork. Esta integración fue basada en la tecnología Internet of Things (IoT), que comprende la interconexión de una serie de dispositivos a través de internet.

En cuanto a la solución a implementar, nos trazamos dos objetivos principales, el primero enfocado en los beneficios para un potencial usuario del cowork y el segundo apuntando a una administración más eficiente del espacio y de energía.

Para el primer objetivo pensamos en la mejora del confort de los usuarios a la hora de hacer uso del espacio, específicamente en cuanto a la temperatura y la seguridad.

Otro objetivo muy importante planteado fue realizar un uso eficiente de la energía en el espacio de cowork. Al realizar esto se reduce el costo de las facturas además de ayudar a reducir los efectos del cambio climático.

A su vez, a nivel del desarrollo de nuestras capacidades técnicas como profesionales en el área de la tecnología, identificamos los siguientes objetivos. Primero, comprender el funcionamiento de una plataforma cloud, en este caso Thingsboard. Esto incluye su instalación y configuración. Entender sus características y la arquitectura del mismo. Por último, también se planteó comprender y utilizar REST APIs y los distintos componentes electrónicos necesarios como para poder llevar a cabo el proyecto.

Conceptualización de la solución desarrollada

Las funcionalidades del proyecto desarrollado se pueden segmentar según los subsistemas descritos en el anteproyecto. A su vez, estos subsistemas están interconectados según una cadena lógica en la nube. Debido a las restricciones de tiempo y a que algunas funciones requieren más tiempo de lo anticipado en el diagrama de Gantt, especialmente la etapa de testeo de cada funcionalidad, sólo fueron implementadas las funciones básicas descritas en el anteproyecto. De cualquier forma, con mayor tiempo, este podría seguir siendo desarrollado para alcanzar su potencial total.

Calefacción/Refrigeración

Tanto la calefacción como la refrigeración están controlados por el sensor de temperatura del ambiente. El sensor de temperatura de ambiente tiene dos umbrales, uno para cuando hace calor, el umbral máximo, y otro para cuando hace frío, umbral mínimo. Cuando el sensor indica que la temperatura está por debajo del umbral mínimo, se activa la calefacción, es decir se prende la caldera para activar la

loza radiante, con la intención de aumentar la temperatura del lugar. En cambio, cuando la temperatura pasa el umbral máximo, se activa la refrigeración, teniendo como actuadores el ventilador y las ventanas.

Ventilación
Los posibles casos en donde se requiera de la ventilación son: cuando hay mucha gente en el espacio y comienza a subir el nivel de CO2, o cuando la temperatura es muy alta. La ventilación está compuesta por un ventilador y la posibilidad de abrir ventanas en caso de que así se desde o así se indique según los parámetros de control.

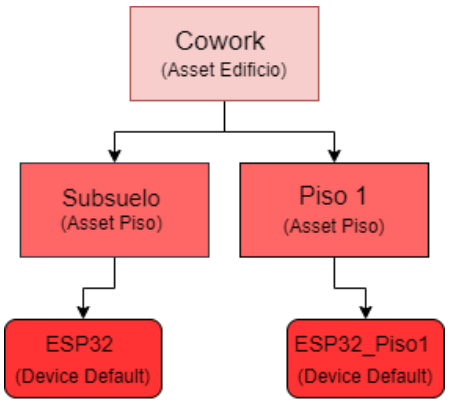
Iluminación
La iluminación en este caso está relacionada con las ventanas, teniendo la posibilidad de permitir o prohibir el ingreso de la luz externa, pudiendo abarcar problemas de iluminación y de calefacción.

Control de acceso
El control de acceso nos permite hacer más eficiente el ingreso y salida del cowork. Con el módulo RFID podemos detectar los tags correspondientes a cada usuario, pudiendo así permitir o negar el ingreso a cada persona, y también llevar un conteo de las personas dentro del lugar, lo cual nos ayuda también con la ventilación.

Implementación de la solución

El proyecto fue desarrollado utilizando dos placas ESP32 conectadas por WIFI al servidor de Thingsboard. Distribuímos estas placas de forma que hubiera una por piso, reportando la información de los sensores correspondientes y desatando las órdenes para poner en funcionamiento los actuadores. De esta forma, se definió la siguiente topología, con las relaciones de dependencia identificadas, que posteriormente se implementó en la sección de Assets en Thingsboard.

Figura 1. Diagrama jerárquico de las entities



Subsuelo

En el piso inferior, se encuentra ubicada la caldera encargada de calentar el agua para el sistema de losa radiante y la puerta con su respectivo sensor RFID para controlar el acceso y contabilizar la cantidad de personas. Por lo tanto la información reportada por la placa ESP32 corresponde:

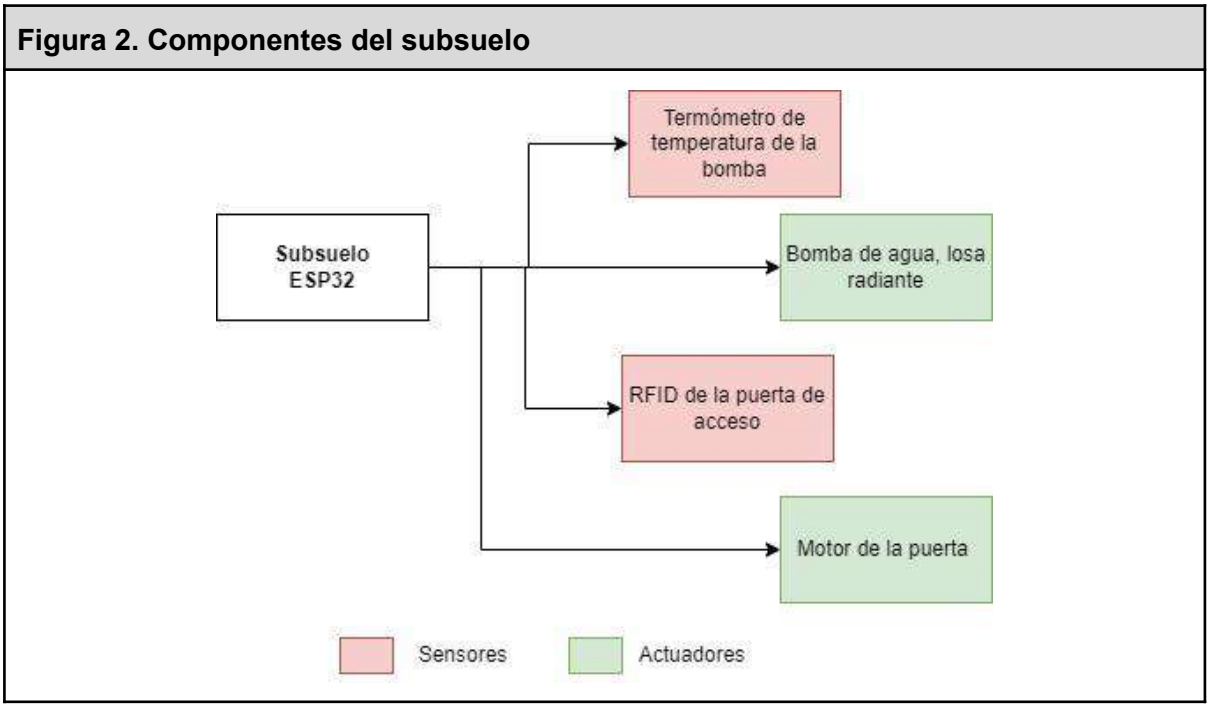
- Telemetría del sensor de temperatura sumergible
- Telemetría de la ID leída del sensor RFID

La información que recibe esta placa correspondiente al accionamiento de los actuadores es la siguiente.

- Prender/apagar calentador
- Prender/apagar bomba de agua
- Abrir/cerrar puerta

A diferencia del anteproyecto, se decidió ubicar la puerta de ingreso al Cowork en el piso inferior, dado que de esta forma quedaba mejor balanceada la carga de componentes, y en consecuencia la cantidad de pines utilizados, en ambas placas ESP32.

Figura 2. Componentes del subsuelo



Piso 1

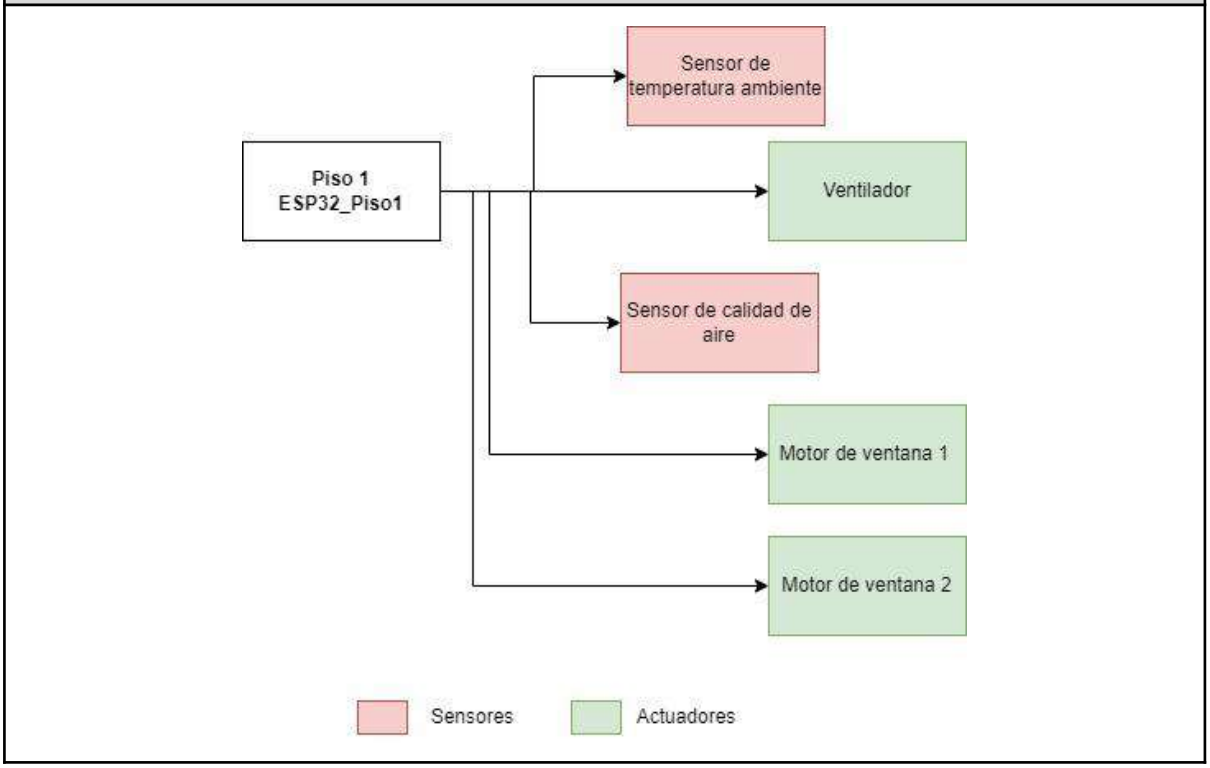
En el piso superior, se encuentra la sala de trabajo. Aquí se encuentran varias interfaces donde nuestro sistema interactúa con el entorno. En primer lugar, se reporta continuamente la medición de distintos parámetros ambientales que están siendo monitoreados por distintos sensores. Este reporte es realizado por la placa llamada “ESP32_Piso1”, obteniendo entonces:

- Telemetría de la temperatura ambiente
- Telemetría de la calidad del aire

A su vez, esta placa también maneja distintos actuadores para ajustar los parámetros leídos a los valores umbral, como temperatura ambiente de confort, definidos por el administrador.

- Ventilador
- Servomotor en las ventanas (2)
- Opacar ventanas (2)

Figura 3. Componentes del Piso 1



Ajuste del sistema a condiciones dadas

Como se describió anteriormente, el sistema está programado para reaccionar a distintos cambios en el entorno percibidos por los sensores. Estas automatizaciones fueron codificadas en la Rule Chain de Thingsboard y son explicadas en mayor detalle en la sección Prototipos.

Apertura de puerta

Para esta lógica implementamos una lista de usuarios, cada uno correspondido con un determinado tag, por lo tanto la puerta sólo se abrirá en caso de leer un tag perteneciente a dicha lista. La puerta se mantendrá abierta durante 30 segundos para que el usuario pueda ingresar.

Prender ventilador o abrir ventana

Ambos actuadores pertenecen al sistema de refrigeración, es decir, son controlados por el sensor de temperatura del ambiente. Dentro de la lógica de esta

implementación se elige cual es el más óptimo considerando las condiciones climáticas actuales.

No hay gente dentro del cowork
La lógica pensada para la apertura de puerta nos permitió generar un contador con la cantidad de personas que ingresan o salen del cowork, teniendo así la información de si hay o no gente dentro del cowork. El principal objetivo de este actuador es el ahorro de energía, con acciones como el hacer menos estrictos los umbrales de temperatura y opcar ventanas en caso de que haga mucho calor.

Horario de apertura y cierre del cowork
Al determinar una hora de apertura y de cierre, podemos prender la calefacción unas horas antes de la apertura así en el momento que el usuario ingresa ya hay una temperatura ideal.

Planificado y experimentación

El desarrollo del proyecto se segmentó en distintos subniveles de abstracción con el fin de dividir el trabajo en tareas más manejables con objetivos alcanzables y medibles. Para ello se creó un plan de prototipos y pruebas de concepto con el objetivo de minimizar los riesgos que un trabajo de esta magnitud presenta. Se utilizó el criterio de encontrar distintas acciones concretas que cada componente debía realizar para definir cada prueba de concepto, teniendo como objetivos genéricos el aprendizaje sobre el funcionamiento del componente y producir un código que lo haga funcionar. Posteriormente, todos estos distintos segmentos fueron incorporados progresivamente a un código único para cada placa ESP32.

Pruebas de concepto

Para comprender conceptos básicos del funcionamiento de los componentes y del software a utilizar, se implementaron las siguientes pruebas de concepto que se describen en las categorías a continuación.

1. THINGSBOARD

1.1. Conexión de la placa ESP32 con Thingsboard

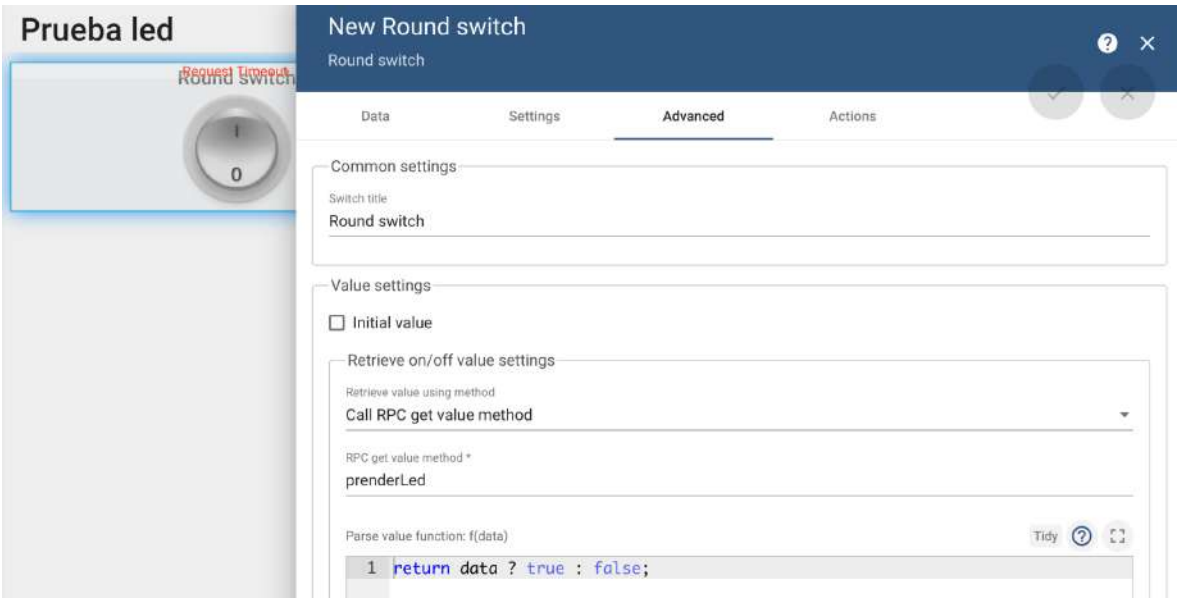
Esta prueba de concepto tenía como principal objetivo efectivamente establecer una conexión bidireccional entre la placa ESP32 con la plataforma Thingsboard. Para ello, primero se requirió conectar el módulo a WIFI, que se realizó utilizando la biblioteca <Wifi.h>. Una vez conectados, para entablar una “conversación” con el servidor, se utilizó el protocolo MQTT a través de la biblioteca <PubSubClient.h>[1].

[1]Código prueba 1.1.
https://github.com/Conectando-Cosas-2022/Smart-Cowork/tree/pruebas_thingsboard/ConectarWifi

Esta permite fácilmente subscribirse y publicar información del servidor desde la placa utilizada. Para ello, se tuvo que definir una entidad “Device” que correspondiera con la placa y copiar su token en el código.

Para verificar que esta información efectivamente estaba siendo intercambiada, creamos un dashboard con un control widget que realizara un llamado RPC a una función “prenderLed” en la placa. Esta estaba conectada a una led que era encendida o apagada, dependiendo de su estado actual, al llamar a la función en cuestión y posteriormente enviaba su estado actual.

Figura 4. Dashboard de prueba



Con esta prueba comprendimos las bases del intercambio de información en Thingsboard, cómo funcionan y cómo se procesan los llamados RPC. Estos conocimientos fueron básicos pero claves para el desarrollo de las partes más complejas del proyecto que implicaban la interconexión con thingsboard.

1.2. Envío y procesamiento de una telemetría

De la mano con la prueba de concepto anterior, era de suma importancia reportar el estado de nuestros sensores al sistema en la nube dado que se esperaba que últimamente sea allí donde se centralice la toma de decisiones lógicas. Para ello desarrollamos una rutina que publica la lectura de una variable en forma de una telemetría.

En conjunto con este punto, se encontró una biblioteca que permite fácilmente realizar mediciones con el sensor de temperatura sumergible. De esta forma, utilizamos este dato para ser reportado en la telemetría[2].

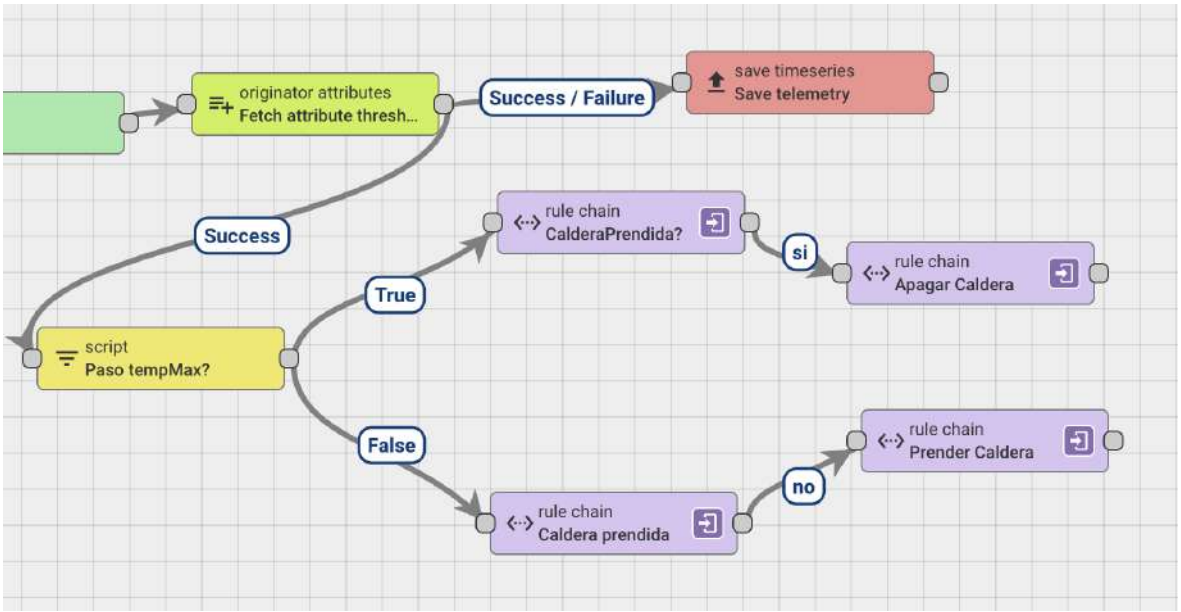
[2] Código prueba 1.2.

1.3. Uso de la rule chain para llamados RPC condicionales

Dado que por diseño la toma de decisiones lógicas se iba a realizar en la nube, fue de vital importancia entender como poder filtrar los llamados RPC a la placa con distintas condiciones. Hasta ahora, solo se había podido invocar llamados RPC directamente desde el dashboard utilizando el control widget.

Para ello se creó una rule chain de prueba que se ejecutara cada vez que se reciba un mensaje de telemetría. En esta prueba, uno de nuestros objetivos era entender cómo definir variables globales desde Thingsboard y cómo importarlas a la rule chain para poder trabajar con ellas. Esto se resolvió definiendo atributos de servidor, que pueden ser editados únicamente desde la nube, que posteriormente fueron importados a la cadena de reglas usando los bloques de enrichment. De esta forma, se agregan la “key” y el “value” de la variable a utilizar al JSON correspondiente a la metadata del mensaje, donde es sencillamente accesible desde un script.

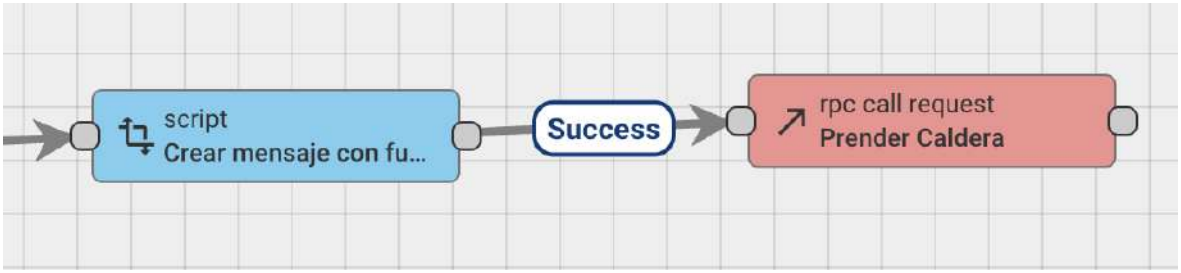
Figura 5. Rule chain de prueba para el prendido y apagado de la caldera



Para poder generar el mensaje que contenga dentro una key ‘method’ con el valor del nombre del método, y una key “params” con los parámetros del método, se utilizó el bloque “Create New Message” que permite definir manualmente el mensaje a ser envíado. Este fue concatenado con el bloque “rpc call request” que ejecuta la acción de enviarle a la placa el mensaje recién creado

https://github.com/Conectando-Cosas-2022/Smart-Cowork/tree/pruebas_thingsboard/pruebaSensorThbrd

Figura 6. Elementos del llamado RPC (Rulechain prender caldera)



2. SENSORES

2.1. Sensor de temperatura de agua (DS18B20)

Para realizar las mediciones de la temperatura del agua dentro de la caldera utilizamos el sensor DS18B20 sellado con el fin de que pudiera ser sumergido. El objetivo del uso de este sensor fue monitorear constantemente la temperatura del agua siendo calentada, para que esta no exceda una temperatura umbral que cause que los componentes a los que está conectado, como el caño y la bomba, se rompan.

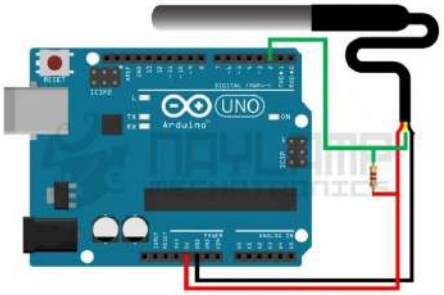
Seleccionamos este en particular dado que utiliza únicamente un puerto de control en la placa utilizando el protocolo 1-Wire y que puede ser alimentado por la placa, siendo sus voltajes operacionales entre 3.0V y 5.5V. A su vez, como se va a estar midiendo temperaturas de agua, esta nunca va a superar los 100ºC, siendo menor a su cota superior de 125ºC [3].

Tras una búsqueda en internet, se encontró un código desarrollado por Saúl García que integraba la biblioteca <DallasTemperature.h> y <OneWire.h> para hacer mediciones utilizando este componente [4]. A su vez, en la misma página se describe la necesidad de utilizar una resistencia pull-up de 4,7 Ω entre el pin de datos y 5V.

[3] DALLAS SEMICONDUCTOR. (n.d.). *DS18B20 Datasheet*. ALLDATASHEET. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html

[4] García, S. (2020, June 17). *Cómo conectar un sensor de temperatura DS18B20 a Arduino*. 330ohms. Retrieved December 11, 2022, from https://blog.330ohms.com/2020/06/04/como-conectar-un-sensor-de-temperatura-ds18b20-a-arduino/

Figura 7. Diagrama de conexión de pines del sensor DS18B20



*Extraída de: https://naylampmechatronics.com/blog/46_tutorial-sensor-digital-de-temperatura-ds18b20.html

Para probar su funcionamiento y observar empíricamente su tiempo de reacción a distintas circunstancias, colocamos dos recipientes con agua a distinta temperatura. Consecuentemente, se realizaron mediciones de un recipiente a otro[5]. Se determinó que el tiempo de establecimiento de una medida dependía de la diferencia de temperatura entre los recipientes. En conclusión, se consideró que realizando mediciones cada 10 segundos se iba a poder observar correctamente la dinámica de la temperatura.

2.2. Sensor de temperatura ambiente y humedad (DHT22)

Este sensor tiene como objetivo reportar continuamente parámetros de temperatura y humedad del ambiente a la nube. El objetivo de esta primera prueba fue poder obtener estas mediciones conectando el sensor con la placa ESP32. Para ello descargamos la biblioteca <DHTesp.h> y seleccionamos uno de los ejemplos proveídos por la propia librería[6]. Este, a su vez, ejecutaba las mediciones de temperatura utilizando llamadas asincrónicas cada 20 segundos utilizando la biblioteca <Ticker.h>, cosa que fue muy útil considerando que la placa iba a correr otro código y realizar mediciones de temperatura en simultáneo. El código de ejemplo estaba implementado para el sensor DHT11 en vez del DHT22, por lo que se le hicieron algunas modificaciones para que este fuera compatible.

En varios blogs consultados se indicaba el uso de una resistencia pull up en el pin de datos, pero bajo mayor inspección se notó que nuestro módulo ya tiene soldada una resistencia y un capacitor en paralelo entre el pin de datos y 5V. Finalmente, se

[5] Código prueba 2.1. https://github.com/Conectando-Cosas-2022/Smart-Cowork/blob/pruebas_thingsboard/pruebaSensorThbrd/pruebaSensorThbrd.ino

[6] Beegee-Tokyo. (2021, February 19). *DHTESP/examples/dht_esp32 at master·Beegee-Tokyo /DHTESP*. GitHub. Retrieved December 11, 2022, from https://github.com/beegee-tokyo/DHTesp/tree/master/examples/DHT_ESP32

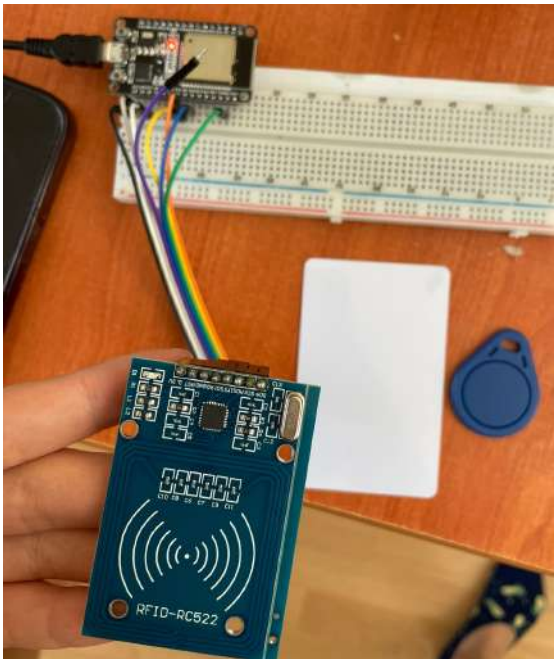
ha utilizado el conocimiento aprendido en la sección Thingsboard para enviar la medición en forma de telemetría al servidor cloud.

2.3. Lecturas Módulo RFID (RC522)

Este módulo[7] permite tanto la lectura como la escritura de tags RFID, pero se va a utilizar únicamente para la lectura. Estas se realizaron a partir de un código encontrado en la web[8] que integraba este módulo con una tarjeta ESP, el cual posteriormente fue probado y optimizado para nuestra aplicación[9]. A su vez, en esta publicación se encontró un diagrama de conexión de pines coherente con nuestra placa, por lo que fue utilizada como base para desarrollar nuestro circuito.

Utilizando lo aprendido en la prueba 1.2, las lecturas del sensor se envían en forma de telemetría al device ESP32. Dado que el ID obtenido es un número de 32 bits, se optó por traducirlo a notación hexadecimal para ser más amigable con el usuario y enviarlo como un string.

Figura 8. Prueba del módulo RFID



[7] NXP Semiconductors. (n.d.). *MFRC522 Datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/227839/NXP/MFRC522.html

[8] Xukyo. (2021, May 3). Using an RFID module with an ESP32. AranaCorp. Retrieved December 11, 2022, from https://www.aranacorp.com/en/using-an-rfid-module-with-an-esp32/

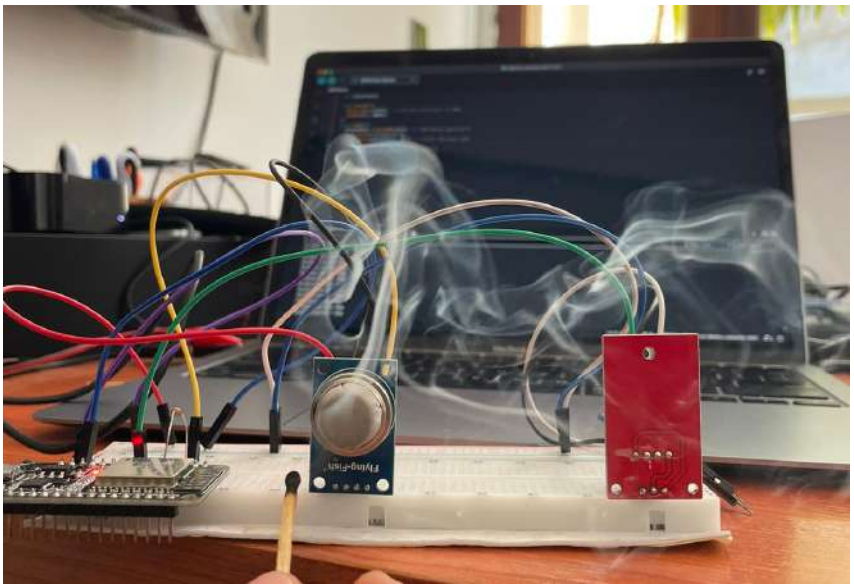
[9] Código prueba 2.3. https://github.com/Conectando-Cosas-2022/Smart-Cowork/tree/pruebas_thingsboard/rfid

2.4. Sensor de CO_2 (MQ135)

Al igual que para los demás sensores utilizados, esta prueba de concepto tuvo como objetivo extraer mediciones de los niveles de CO_2 en el aire a partir de conectar el sensor MQ135 a la placa. Después de leer su datasheet, se concluyó que este sensor detecta la presencia de una gran variedad de gases nocivos para el ser humano, incluyendo el dióxido de carbono, indistintamente otorgando como medición la cantidad de partículas por millón (ppm) detectadas.

Este componente tiene como precondición estar por 48 horas encendidos, previo a realizar cualquier medición[10]. Esto se debe a qué el sensor debe calentar para generar una reacción química dentro que produce el compuesto químico que es sensible a las condiciones de calidad del aire. Una vez cumplidas las 48 horas se realizó una prueba ejecutando lecturas[11] para el aire estándar y al acercar un fósforo encendido. En el reporte de las mediciones visualizado en el monitor serie, se podía notar cómo aumentaba significativamente el valor de ppm censadas. El principal desafío de esta prueba fue entender cómo reportaba las lecturas el sensor e ingeniar un método de precalentar el módulo por 48 horas sin tener que estar conectado a una computadora, que finalmente logramos realizar modificando una fuente de 5V.

Figura 9. Prueba del sensor MQ135



[10] Zhengzhou Winsen Electronics Technology Co. (2015). *MQ135 datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/1307647/WINSEN/MQ135.html

[11] Código de prueba 2.4. https://github.com/Conectando-Cosas-2022/Smart-Cowork/tree/pruebas_thingsboard/prueba_MQ135

3. ACTUADORES

3.1. Servo para trancar la puerta

Todos los servos utilizados en el proyecto fueron programados a través de la biblioteca <Servo.h>. Como primera prueba se ejecutó el código de ejemplo brindado por la propia librería[12], conectando el pin de datos del servomotor a un DGIO pin de la ESP32.

Antes de instalar la tranca y el motor, decidimos hacer pruebas en donde comprobemos que el sistema se tranque de forma correcta y el motor servo efectivamente pueda llevar a cabo el movimiento de la tranca. El principal problema que afrontamos fue con el pasador, dado que presentaba una fricción considerable que hacía que fuese difícil abrirlo hasta moviéndolo con la mano. Para solucionar este inconveniente le pusimos vaselina al eje del pasador, haciendo que presente menos rozamiento y facilitando su apertura.

Unimos el aspa del servo con la perilla del pasador utilizando un cable atado a ambos extremos. Otro inconveniente que afrontamos fue que en primera instancia habíamos diseñado que el movimiento del servo fuese desde la posición vertical arriba y girase en el sentido de las agujas del reloj. Este movimiento ocasionaba que el pasador se insertara en la tranca inferior y que no se abriera correctamente. Para solucionarlo, invertimos el giro del servo, comenzando en la posición vertical abajo y girando en sentido antihorario.

Figura 10. Prueba del mecanismo previo a la instalación

[12]Código prueba 3.1.
https://github.com/arduino-libraries/Servo/tree/master/examples/Sweep

Figura 11. Prueba del mecanismo posterior a la instalación

3.2. Sistema de apertura de ventanas utilizando servos

Se experimentó con distintos mecanismos para la apertura de las ventanas, tanto lateralmente utilizando un motor stepper, su respectivo driver y un engranaje conversor de movimiento circular a traslacional, y radialmente utilizando un servo. Dado el grado de facilidad en el uso y programación del servo y que este producía un movimiento perfectamente compatible con el opacado del Smart Film, en vez de un motor DC como se había planteado en el anteproyecto que presentaba el uso de un servo, optamos por utilizar este en la maqueta final.

Figura 12. Mecanismo de movimiento de ventana

3.3. Ventilador, bomba y resistencia calentadora de agua

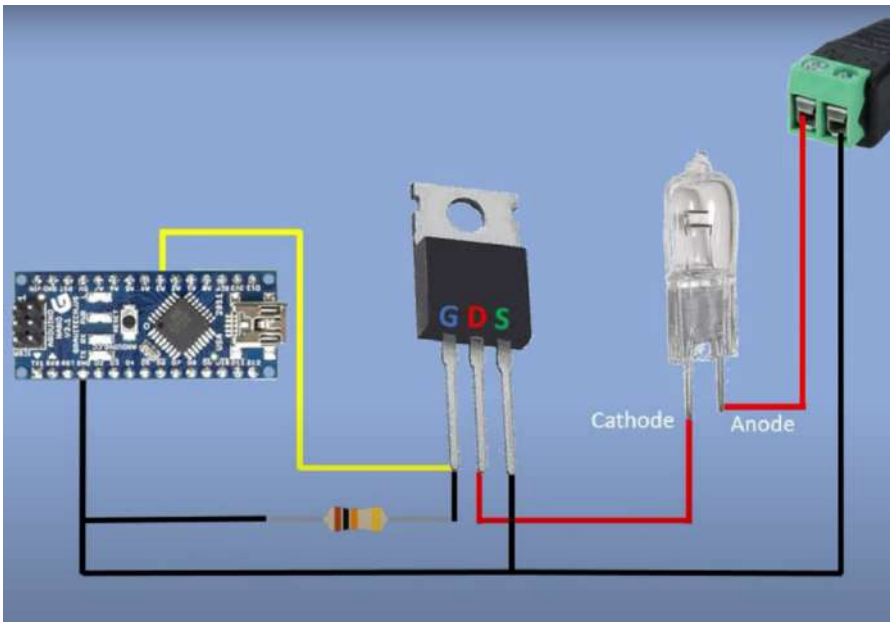
Todos estos tres actuadores se agruparon en una sección dado que todos requieren una alimentación de 12V. La placa ESP32 tiene salidas de 3.3 V por lo que se requiere el uso de una fuente externa para hacerlos funcionar.

El principal problema que se afrontó fue en cuanto al control ON/OFF de los componentes. El funcionamiento de cada uno de ellos es controlado únicamente a través de la alimentación de 12V, sin embargo, las salidas digitales de nuestra placa

solo entregaban 3.3V. Por lo que, surgió la pregunta: ¿cómo controlar un voltaje de 12V utilizando un pin de control de menor voltaje?

Tras realizar una investigación para responder la pregunta planteada, se encontró dos posibles maneras de resolver el problema. Una fue usar un relay y la otra un transistor mosfet. Debido a que debíamos usar tres y que ambos métodos presentaban una diferencia de precio sustancial, decidimos utilizar la última opción específicamente el mosfet IRFZ44, que tolera hasta 55V en el gate y el drain[13]. A su vez, utilizamos como referencia un circuito encontrado en la web[14], que se puede ver en la Figura 13, el cual probamos alimentando a una led de 12V y funcionó perfectamente. Esto permitió que estos tres actuadores fueran controlados simplemente haciendo un digitalWrite con el valor HIGH o LOW para sus respectivos pines de control.

Figura 13. Circuito de control utilizando un mosfet



*Extraída de:https://www.youtube.com/watch?v=XiOcxyyTxy4&t=460s

A continuación, probamos todos los componentes individualmente para comprobar que estos funcionaran como era esperado. Tanto el ventilador como la bomba se desempeñaron como era esperado. No obstante, no fue así la primera vez que probamos la resistencia calentadora de agua. En un principio, habíamos adquirido una fuente de 12V y 2A, y cuando conectamos a la resistencia esta se calentaba apenas perceptiblemente. Dado que la resistencia era originalmente pensada para

[13] NXP Semiconductors. (1999). IRFZ44 datasheet. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/17808/PHILIPS/IRFZ44.html

[14] Mario's Ideas. (n.d.). *Tutorial on how to control 12V devices with Arduino*. Arduino Project Hub. Retrieved December 11, 2022, from https://create.arduino.cc/projecthub/mdraber/tutorial-on-how-to-control-12v-devices-with-arduino-e5b416?ref=user&ref_id=1474727&offset=6

autos cuya batería suministra notablemente más potencia, sospechamos que con una fuente de mayor amperaje íbamos a lograr que funcionase. Utilizando una fuente del laboratorio se probaron distintas intensidades de corriente, y se determinó que a partir de 6A la resistencia calentaba. Se decidió utilizar una fuente de 12V y 7A dado que era la que se encontraba en el mercado local y satisfacía las condiciones halladas experimentalmente.

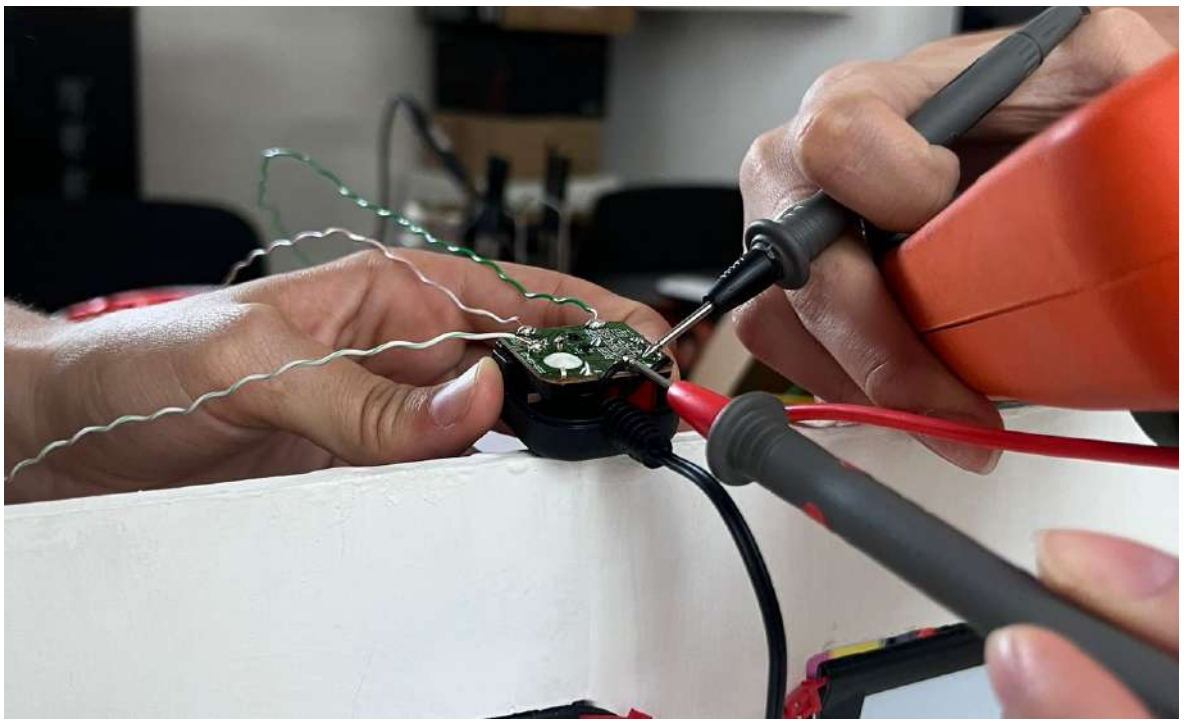
3.4. Smart Film

El control digital del smart film presentó muchos problemas que no pudieron ser anticipados. Ya se sabía que su control iba a ser un desafío dado que no hay mucha información en la web al respecto del componente. Sin embargo, se había investigado el mecanismo que lo hacía funcionar y parecía ser sencillo: cuando el voltaje entre las terminales fuese cero, el vidrio sería opaco y cuando el voltaje fuese un 1 lógico este se encendería.

Al obtener el componente, este venía en conjunto a un controlador con un botón on/off y pilas. Como primera prueba, se separó el controlador y el botón de las pilas y se alimentó la placa con el pin Vin de la ESP32. Posteriormente, lo conectamos al vidrio y este funcionó correctamente.

En segundo lugar, utilizando un multímetro, se analizó el voltaje de salida cuando el botón está presionado dando como resultado 1.2 V. Como prueba, intentamos conectar una fuente de 1.2V a los pines del componente, pero no hizo que funcionara.

Figura 14. Mediciones del voltaje del controlador usando el multímetro



Un poco apresuradamente, intentamos intervenir el controlador original conectando uno de los extremos del botón a un pin digital de la ESP32. Desafortunadamente, esta conexión no estaba bien y causó que la placa ESP32 se quemara, causando que se tuviera que conseguir un repuesto. Como no se contaba con más presupuesto en

Prototipos

Se consideran prototipos a aquellos que tienen como fin construir parte de la solución.

1) Control de temperatura del agua y el encendido del calentador

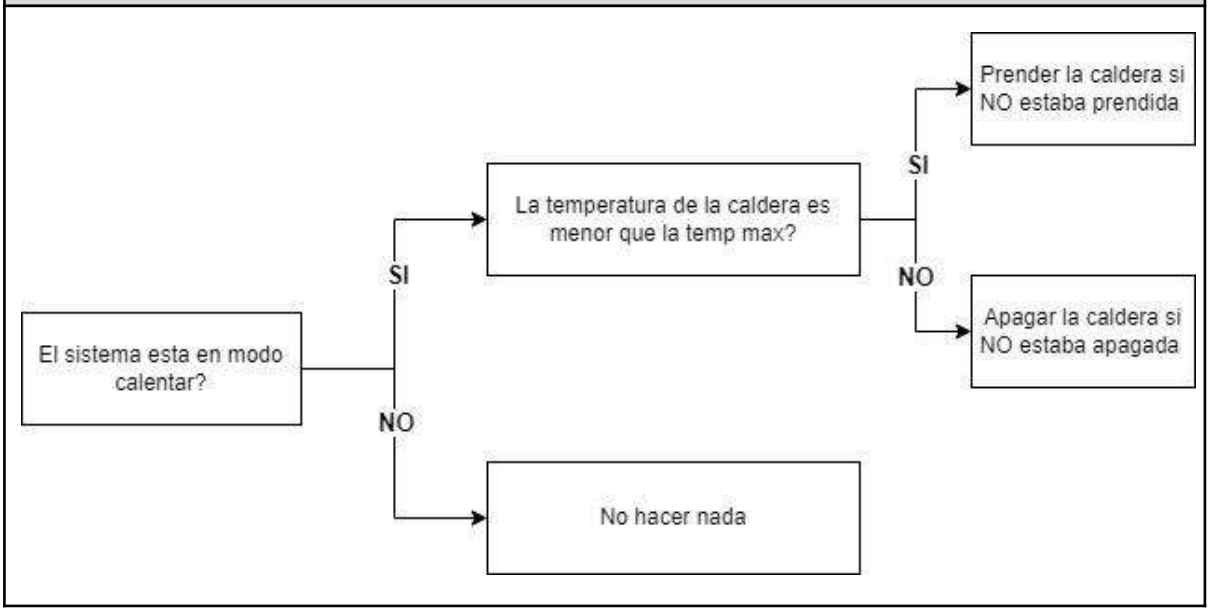
El primer prototipo que fue realizado tuvo como objetivo integrar las mediciones del sensor de temperatura del agua (DS18B20) para efectuar el control del prendido y apagado de la resistencia calentadora. De esta forma, se aisló el sistema “caldera” para verificar su funcionamiento cuando la temperatura del agua supera la temperatura umbral.

Utilizando los conocimientos adquiridos en las pruebas anteriores, integramos los códigos de las secciones 1. y 2.2. para obtener un código que realice mediciones con el sensor de agua, las publique por telemetría y además tuviera una función de callback con los métodos prender y apagar caldera quienes a su vez reportan el estado de la caldera como un atributo de cliente (“estadoCaldera”). A su vez, en una protoboard armamos el circuito de la prueba 3.3. para poder controlar con 12V la resistencia y la conectamos a la placa.

En Thingsboard, se definió un atributo de servidor perteneciente al Device ESP32 (placa del subsuelo) que definiera la temperatura máxima que puede alcanzar el agua en la caldera (“maxTemp”), que puede ser modificado por el administrador del edificio. También se le agregó al device ESP32_piso1 un atributo compartido denominado “modo”. Este puede tomar los valores “enfriar”, “calentar”, “apagar” cuyo objetivo sería indicar cuál es el modo de funcionamiento del sistema de climatización en un momento dado, según distintos parámetros. Este atributo va a ser explicado en mayor detalle en el siguiente punto.

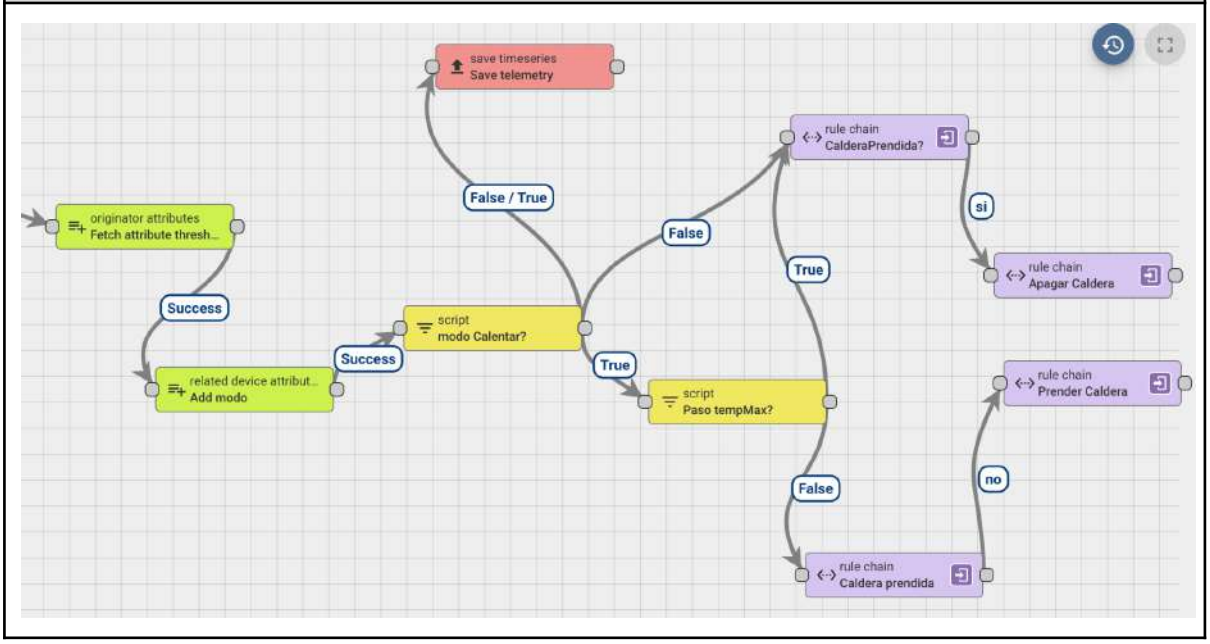
A su vez, se creó una serie de rule chains concatenadas para modularizar los siguientes pasos:

Figura 15. Diagrama de bloques de la prueba



La Rule Chain creada se iba a ejecutar cada vez que llegara un reporte de telemetría proveniente del device ESP32 (correspondiente a la placa del subsuelo).

Figura 16. Rule Chain "validar tempCaldera"

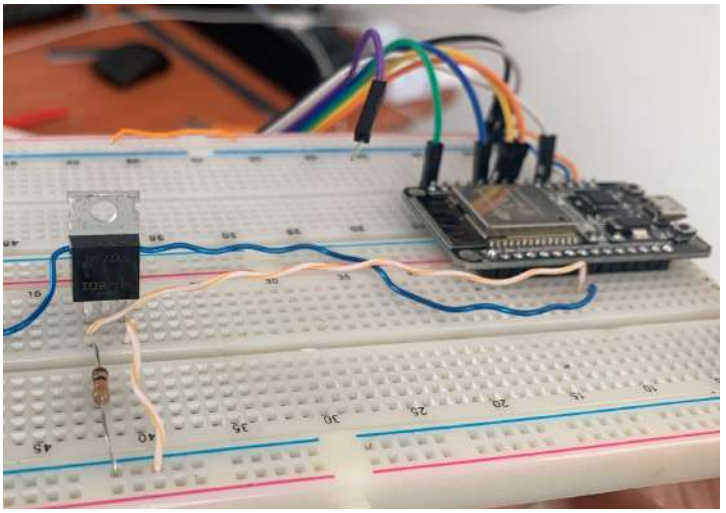


Los principales problemas que afrontamos fueron en cuanto al uso de la propia rulechain. La parte más desafiante sin duda fue intentar integrar al mensaje recibido, un atributo que pertenecía a otro device, que fue el caso del atributo "modo". Este estaba definido en la ESP32_piso1 dado que iba a ser editado a partir de parámetros de ese device, pero en este caso se debía acceder desde el device ESP32_piso0, asemejándose a una variable global. Tras una investigación más

profunda de la documentación de Thingsboard[15] se encontró un bloque de enriquecimiento que permite acceder a los atributos de un device relacionado. Para poder utilizar este nodo, definimos una relación de dependencia lógica entre los dos devices, en la cual el device ESP32_piso1 contiene el device ESP32_piso0. Esto permitió que se pudiera acceder exitosamente a este atributo.

Después de construir esta cadena lógica, se realizaron pruebas extensivas para los casos donde la temperatura leída fuese inferior o superior a la maxTemp definida y se monitoreó la respuesta de la resistencia poniendo en su lugar una led de 12V para poder observar si esta se encontraba prendida o apagada. También se probó cambiar el modo de funcionamiento a “enfriar” y “apagar”, para verificar que la lógica no se ejecute en estos casos.

Figura 17. Imagen del circuito de control de 12V armado en la protoboard



2) Control de umbral de temperatura ambiente

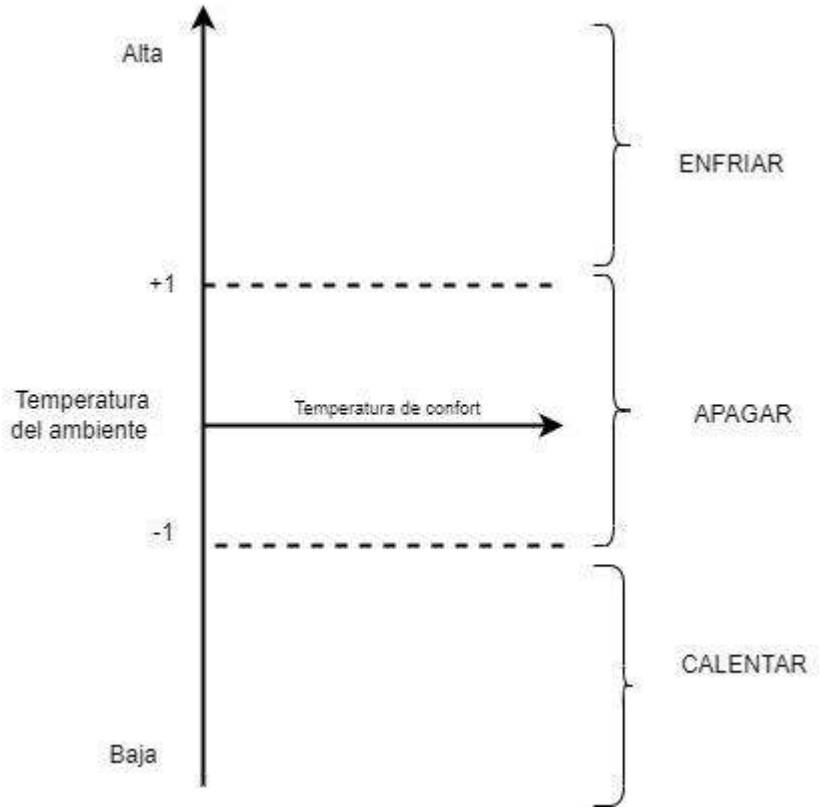
En este prototipo, se integró la recepción de mensajes de telemetría indicando updates de la temperatura ambiente (sensor DHT22) pertenecientes al device ESP32_piso1 con la definición del modo de climatización requerido. Como ya se mencionó en la sección anterior, el atributo “modo” contiene la información de cómo tiene que actuar el sistema de climatización bajo distintas circunstancias. El objetivo de esta integración fue resolver la pregunta: *bajo qué condiciones hay que enfriar, calentar o apagar?*

Está claro que para realizar una comparación hay que definir un valor umbral de temperatura de confort. Este lo definimos como un atributo de servidor en el device correspondiente al piso 1. De esta forma, se desarrolló una rule chain que agregara

[15] Thingsboard. (n.d.). *Enrichment nodes*. ThingsBoard. Retrieved December 12, 2022, from https://thingsboard.io/docs/user-guide/rule-engine-2-0/enrichment-nodes/#device-attributes

este atributo a la metadata del mensaje y que se compare con la medición de temperatura del ambiente obtenida mediante telemetría. El nodo switch tiene el siguiente funcionamiento:

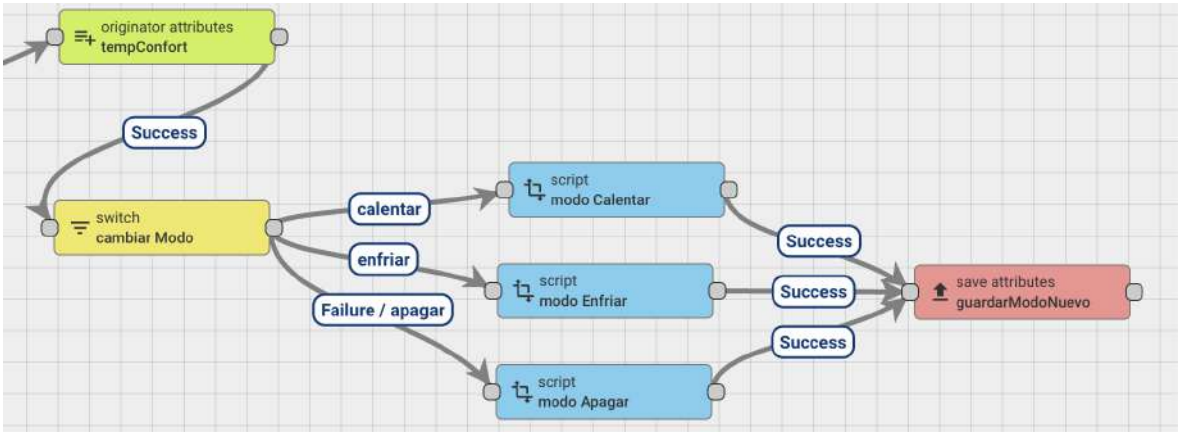
Figura 18. Lógica pensada para el control de umbral de temperatura ambiente



De esta forma hay una zona de temperatura de confort, en donde el sistema de climatización se encuentra apagado, haciendo que se haga un uso eficiente de la energía.

Esta subcadena se ejecuta cada vez que llega un mensaje de telemetría reportando la temperatura ambiente actual proveniente de la placa del piso 1.

Figura 19. Rule chain validar tempConfort



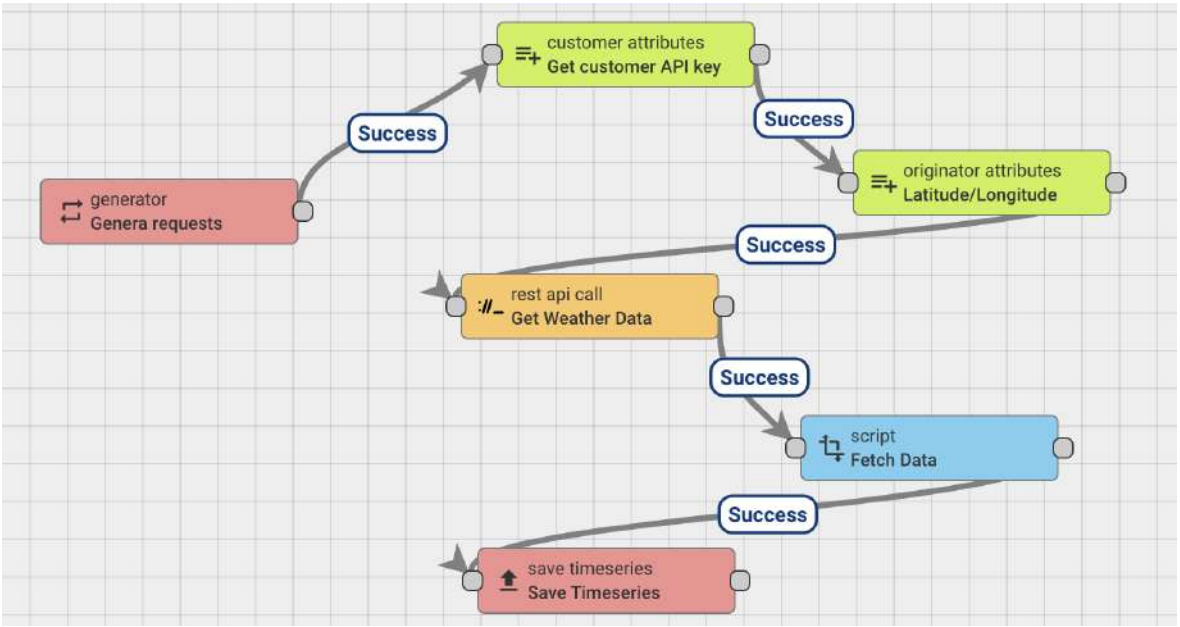
3) Integración con la API del clima

El objetivo de este prototipo fue poder obtener información del clima para una locación dada en tiempo real, con el fin de poder integrar este conocimiento en la toma de decisiones respecto al mecanismo de climatización a utilizar.

Para ello, se encontró un ejemplo proveído por el propio Thingsboard que explicaba detalladamente cómo obtener esta información[16]. Estos datos son extraídos de la REST API de Open Weather, en la cual se creó una cuenta y se obtuvo un token que permite su utilización de forma gratuita. Siguiendo este ejemplo, se construyó la rule chain mostrada a continuación, la cual cada un período de 15 segundos obtiene un reporte del clima para la ubicación geográfica de la Universidad ORT. Este reporte es almacenado en forma de telemetría en el asset Cowork, ubicación que se considera lógica dado que es una característica de la localización y es accesible por todos los componentes que pertenecen al asset Cowork. Debido a la gran calidad de la fuente encontrada, este paso no tuvo mayores inconvenientes. Asimismo, se usó un esquema muy similar para obtener la fecha y hora a partir de una REST API denominada Time Api.

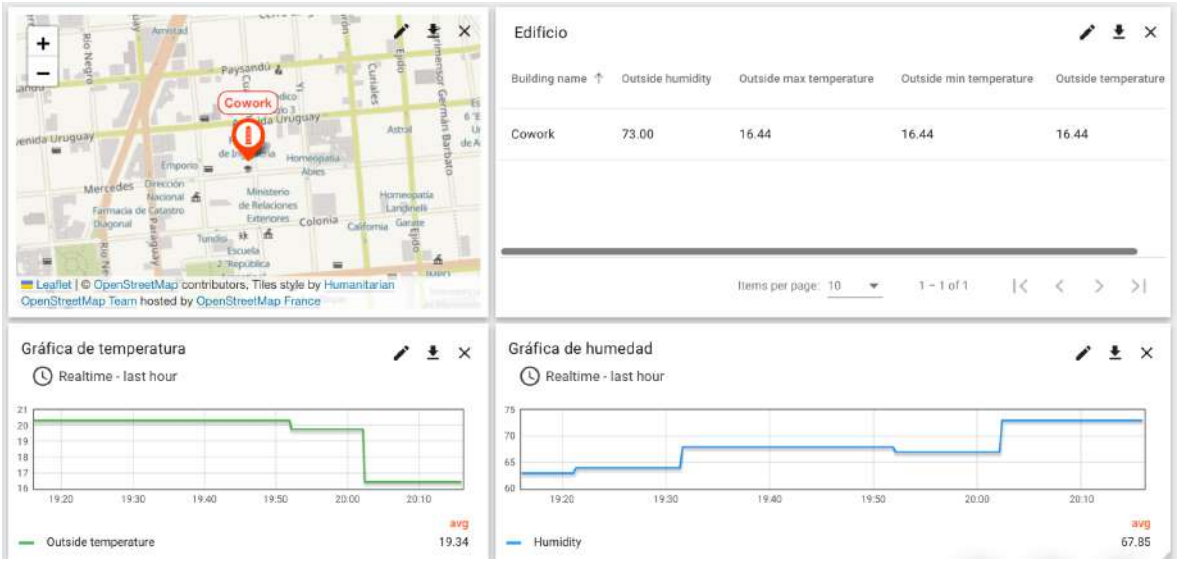
[16] Thingsboard. (2022). *Weather reading using REST API calls*. ThingsBoard. Retrieved December 12, 2022, from https://thingsboard.io/docs/user-guide/rule-engine-2-0/tutorials/get-weather-using-rest-api-call/

Figura 20. Rule chain Get APIs (primera versión)



A su vez, se creó un dashboard que exporte toda la información obtenida de forma gráfica.

Figura 21. Dashboard de los datos obtenidos en la API del clima



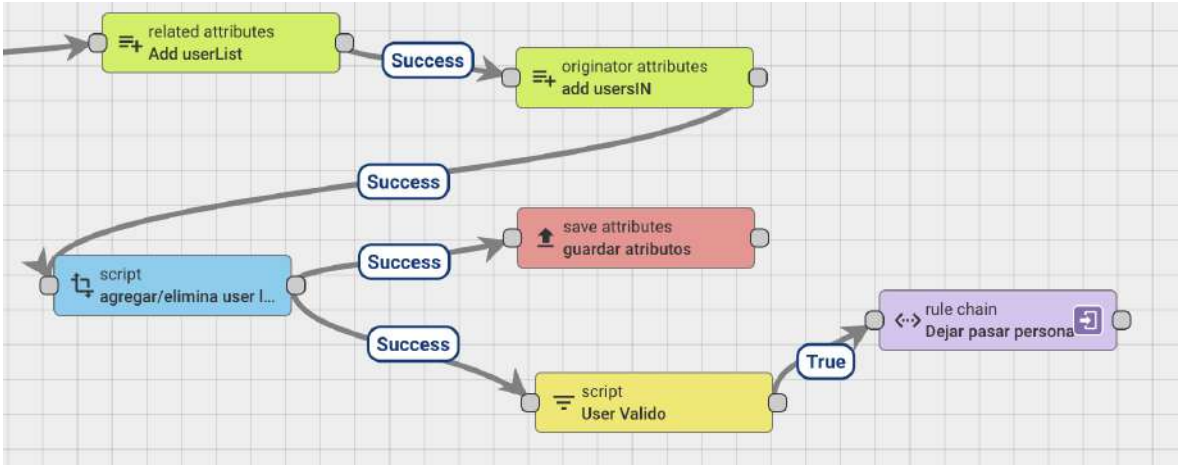
4) Control de acceso y de cantidad de personas

Para este prototipo, aislamos el sistema de acceso con la intención de verificar que el código tanto de la placa como el de thingsboard funcionara bien a la hora de leer un tag y en casos bordes como cuando se vacía el cowork o cuando se ingresa un tag desconocido. Para ello utilizamos el sensor RFID y el servomotor de la puerta en el plano del hardware, y la rule chain validarID en el plano del software en la nube. En este punto ya se estaba integrando todo el software correspondiente a la placa del subsuelo en un solo código[17].

Previo a diseñar la rule chain, se definió un atributo de servidor denominado “userList”. Este es del tipo JSON y contiene una lista de usuarios precargada por el administrador de todos aquellos habilitados para entrar. Cada usuario contiene un atributo “nombre” y un atributo “ID”. Dado que disponíamos de tres tarjetas RFID distintas, definimos tres usuarios en esta lista para posteriormente poder probar el funcionamiento. A su vez, en el device ESP32 (placa subsuelo) se definió un atributo compartido que contenga en una lista todos aquellos usuarios que se encontraran dentro del cowork, denominado “usersIN”.

El funcionamiento planeado fue que cuando llegara al servidor cloud una telemetría que contuviera información de idLeida, este chequee si es un usuario válido. Si lo fuera, se agregaría o eliminaría de la lista usersIN dependiendo si este ya estaba o no en la misma, dado que si estaba en la lista previamente significa que el usuario está saliendo. De ser un usuario válido, se desbloqueará la puerta por 30 segundos para permitir que el usuario ingrese o se retire del edificio.

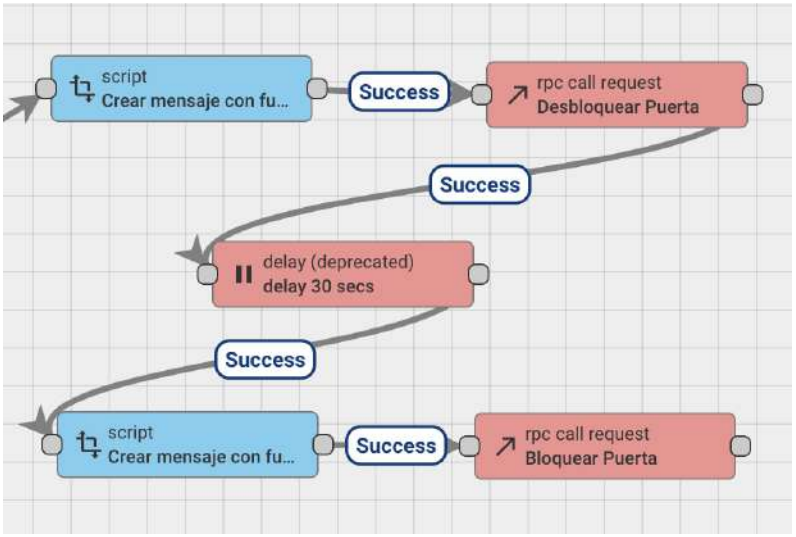
Figura 22. Rule chain Validar RFID



[17] Código utilizado para el prototipo 4: https://github.com/Conectando-Cosas-2022/Smart-Cowork/tree/main_paralelo/placa_piso0

Se entiende que el funcionamiento de la puerta fue simplificado para poder ser modelado, cosa que se discute en mayor profundidad en la sección de Posibles Mejoras.

Figura 23. Rule chain Dejar pasar persona



El mayor problema que tuvimos con este prototipo fue producto de la utilización del tipo de dato JSON, específicamente en el bloque script, dado que se había trabajado muy poco con él anteriormente. Sin embargo, investigando las dudas a medida que surgían, se pudo construir un código funcional.

Utilizando los reportes de telemetría reales leídos por el sensor RFID, se probó que todos los usuarios pudieran ingresar y salir, observando el contenido de la lista usersIN. A su vez, se observó la respuesta del sistema al ingresar usuarios válidos y no válidos, chequeando que no se abriera la puerta ni se agregaran a la lista cuando no lo eran.

Después de verificar que todos los casos funcionaran, se definió un modoNoHayGente para modificar el funcionamiento de la climatización bajando las exigencias de la temperatura de confort de acuerdo a la temperatura exterior. De esta forma, se podría conservar a grandes rasgos la climatización con menor gasto energético y cuando un usuario ingresa, se retoman los parámetros estándar. Esta rule chain se invoca cada vez que la lista usersIN se vacía. A su vez, cuando la lista tiene un único elemento, el atributo “tempConfort” se setea a 21ºC como fue designado inicialmente.

Figura 24. Bloques agregados en rulechain Validar RFID para modoNoHayGente

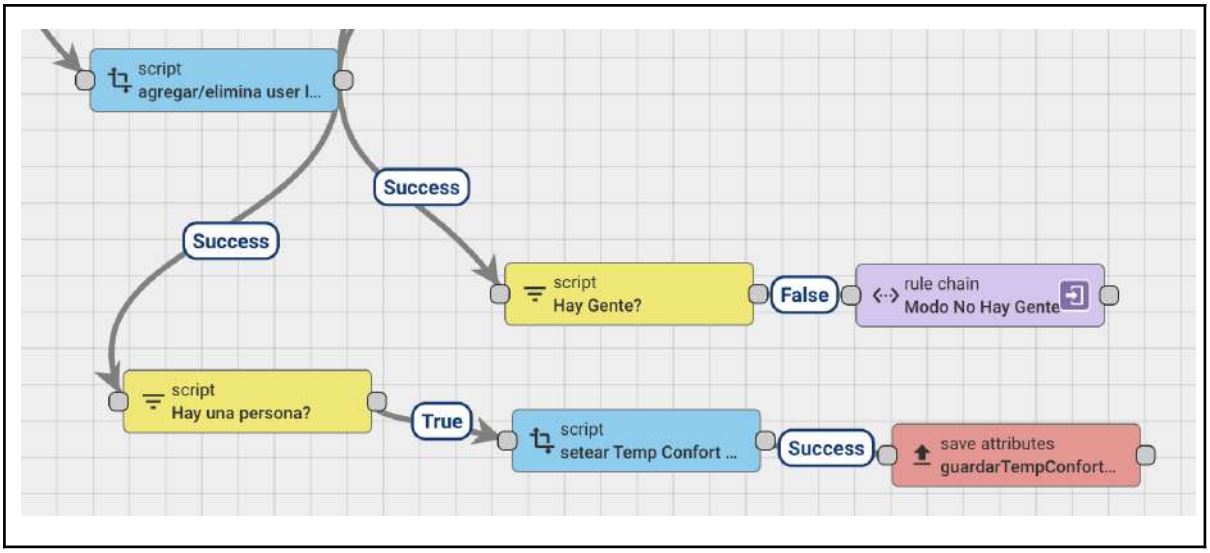
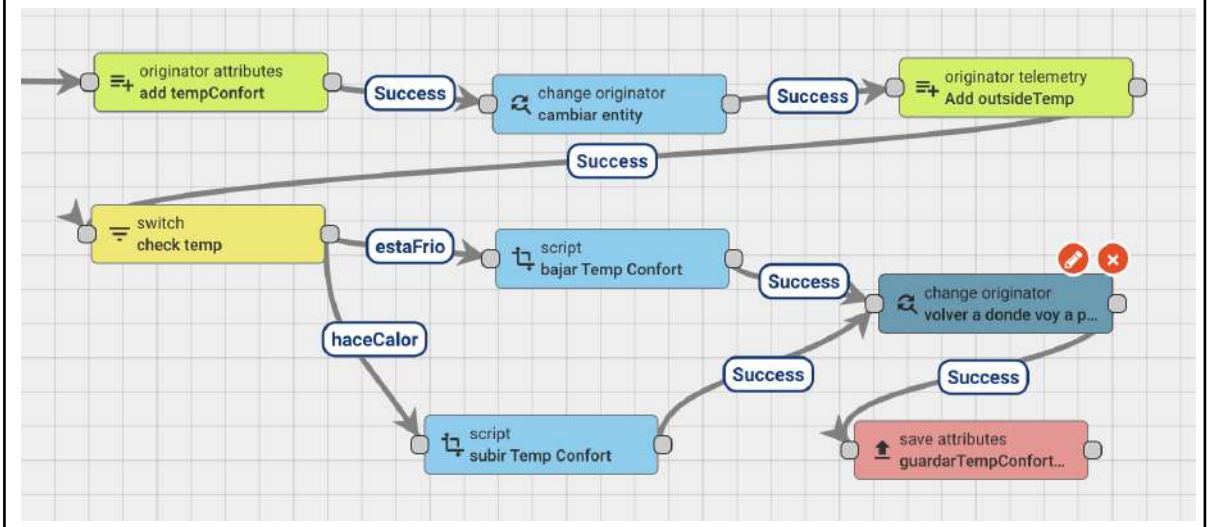


Figura 25. Rule chain modoNoHayGente



5) Llamados a actuadores de climatización

Como fue descrito anteriormente, el atributo “modo” perteneciente a la placa ESP32_piso1 es el que dicta el funcionamiento del mecanismo de climatización. En este prototipo, el objetivo es construir una cadena de reglas que continuamente chequee el estado del atributo “modo” y haga los llamados RPC necesarios para hacer que los actuadores de climatización (la bomba, el ventilador y la apertura de ventanas) se prendan o se apaguen.

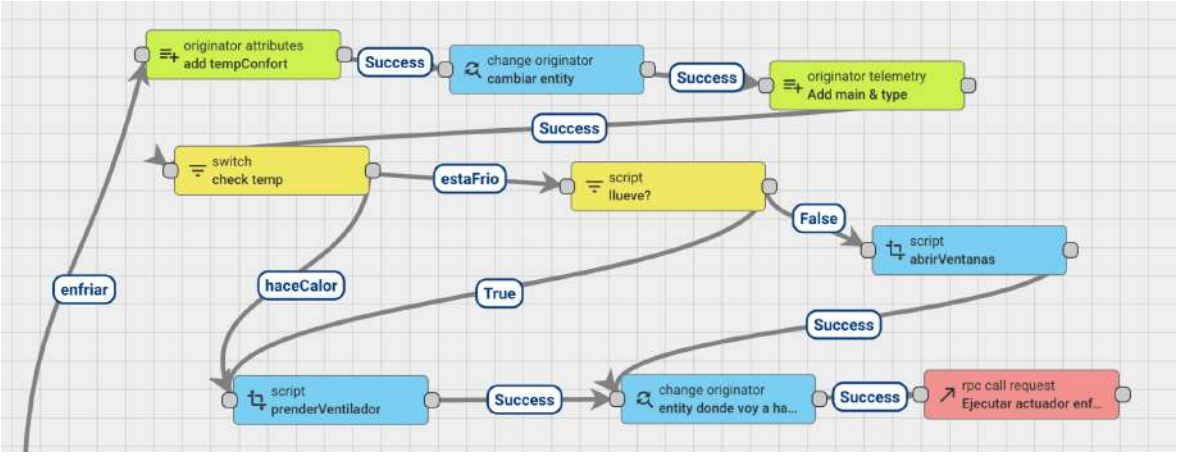
Tabla 1. Actuadores a ser prendidos según modo

	Enfriar	Apagar	Calentar
Bomba	OFF	OFF	ON
Ventilador	ON ?	OFF	OFF
Ventanas	ON ?	OFF ?	OFF

*El “?” corresponde a una condición de acuerdo a la temperatura exterior, si llueve o si ya estaba abierta por otro motivo.

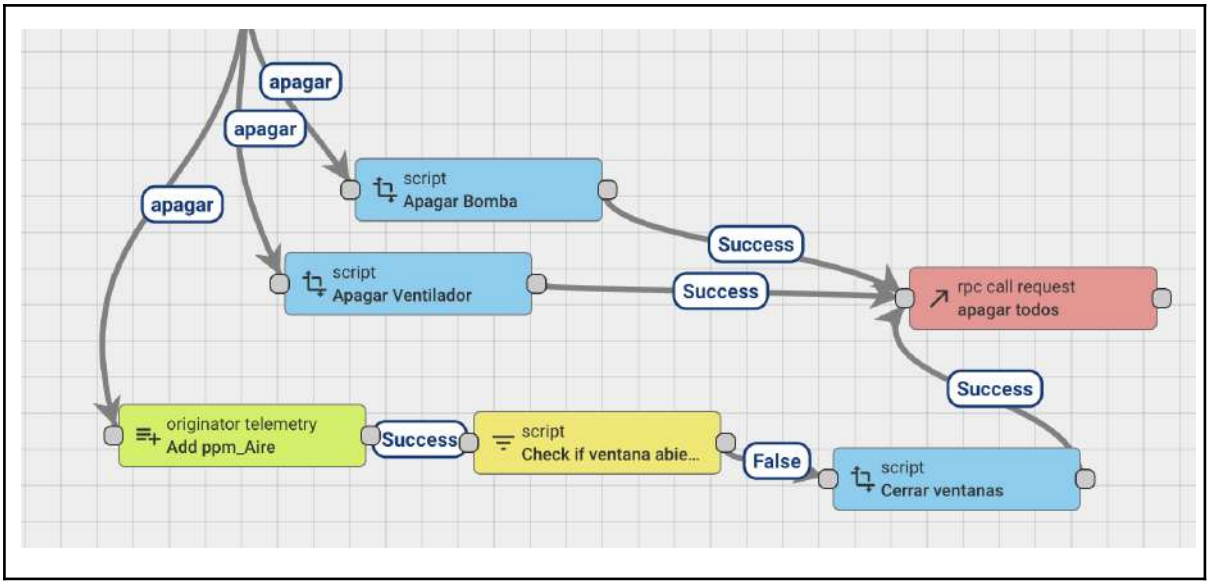
En el modo enfriar se consideraron los parámetros de temperatura exterior y lluvia para decidir qué método de refrigeración usar. Si llueve, no se abren las ventanas. Asimismo, si la temperatura exterior es mayor a la temperatura de confort, no se abren las ventanas sino que se prende el ventilador. Para poder integrar los parámetros extraídos por la API del clima es necesario cambiar el originador del mensaje al Asset Cowork, obtener la telemetría del originador y luego cambiar el originador de nuevo al Device ESP32_piso1 donde se va a ejecutar el llamado RPC.

Figura 26. Rule chain: Llamados a actuadores de climatización (Modo enfriar)



En el modo apagar, se agrega una condición al cerrado de ventanas, de que si estas estaban abiertas para ventilar debido a altas cantidades de CO2 en el aire, estas no se cierren.

Figura 27. Rule chain: Llamados a actuadores de climatización (Modo apagar)



6) Construcción de la maqueta final

Para la construcción de la maqueta lo primero que hicimos fue definir las medidas que queríamos teniendo en cuenta los componentes a utilizar. Para el subsuelo consideramos un tamaño razonable para la puerta y la caldera, dado que iban a ser los elementos más grandes, y para el piso superior tuvimos en cuenta el tamaño de las ventanas y la distribución de los sensores y actuadores.

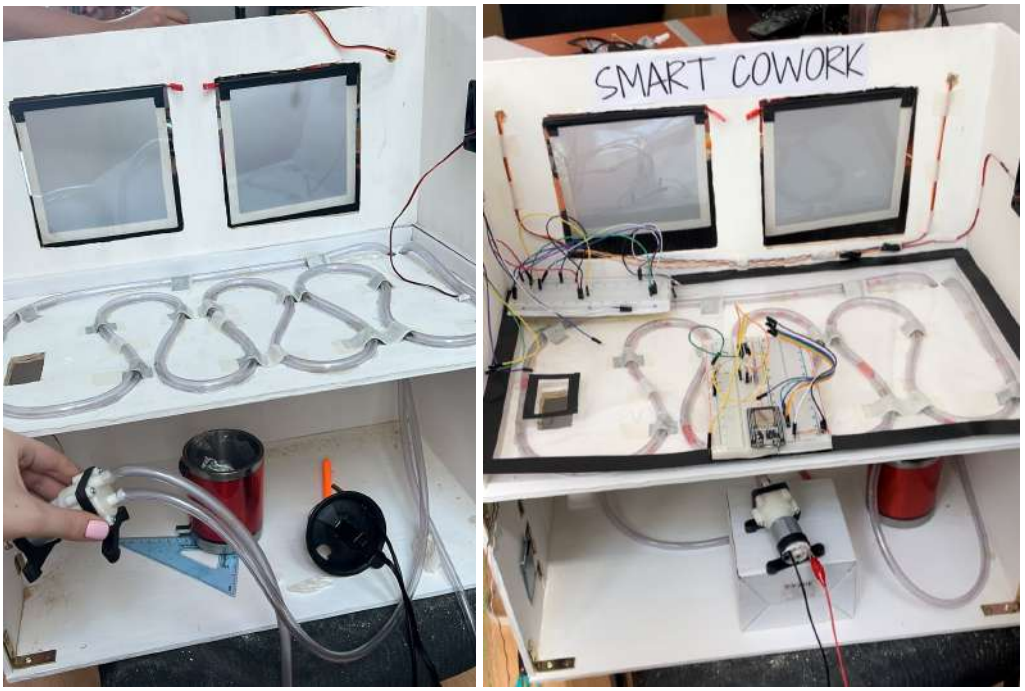
Una vez definidas las medidas, hacemos los orificios necesarios para las ventanas y puerta, luego pintamos las paredes de blanco y le dimos forma encastrando todas las partes, formando un espacio de doble piso. Posteriormente comenzamos a colocar los componentes y haciendo agujeros para que encastren en caso de ser necesario, como para los servo o el caño que va de un piso a otro.

Figura 28. Armado de la estructura

Instalados los componentes hicimos las conexiones tratando de priorizar un correcto funcionamiento y la prolijidad a la vista de alguien externo al proyecto.

Por último agregamos detalles estéticos como el cartel con el nombre del proyecto, el indicador de RFID, y los elementos que simulan una oficina.

Figura 29. Instalación de componentes



Posibles mejoras

Cuando hablamos de posibles mejoras hacemos referencia a implementar funcionalidades o mejorar las realizadas para que complementen las funcionalidades actuales con el fin de obtener un mejor resultado.

Iluminación

- Teniendo en cuenta la estación del año sabemos que en invierno amanece más tarde y oscurece más temprano, podríamos aplicar el uso de fotorresistencias con la idea de que al ingresar gente al cowork y aún no haya amanecido o esté oscureciendo las luces se enciendan automáticamente.

Ventilación

- Cuando implementamos el ventilador a nuestra maqueta nos dimos cuenta que al ponerlo sobre una de las paredes el aire que ventila es mucho menor

al que podría ventilar si lo hubiéramos colocado donde no quede cubierto. Esto se podría mejorar con la implementación de un ventilador más grande, sin salirnos de la escala, o haciendo un agujero detrás del ventilador permitiendo que se ventile más aire.

Calefacción y refrigeración

- Consideramos que la losa radiante es un buen método de calefacción, sin embargo podríamos implementar algún otro actuador que cumpla la función de aumentar la temperatura en el cowork, como una estufa eléctrica o aire acondicionado con calor.
- Se podría haber seleccionado otro tipo de material para el caño que nos permita una mejor distribución del mismo, logrando que abarque todo el piso y sea mejor la calefacción.
- Si hiciéramos que la temperatura de confort dependiera de la estación podríamos ahorrar energía y además simplificar algunos procedimientos.
- Teníamos la intención de implementar una cadena de reglas que encendiera la calefacción previo a la apertura del cowork y que la apague próximo a su cierre.

Control de acceso:

- La idea principal para el control de acceso era que además de controlar el acceso con el módulo RFID la puerta se abriera sola automáticamente, sin embargo esta opción era complicada de implementar por el tamaño y el espacio que disponíamos. Por ende una posible mejora es mejorar el funcionamiento de la puerta.
- Asimismo, en vez de esperar un tiempo definido arbitrariamente (30 segundos) se podría instalar un sensor de proximidad que cense cuando la puerta se abrió y cuando se volvió a cerrar, y que en ese momento pase la tranca. De esta forma se asemejaría más a la funcionalidad real.

Aprendizaje

En cuanto al aprendizaje podemos decir que fue notorio en muchos aspectos, principalmente a la hora de hacer las conexiones con Thingsboard, la creación de una página web, trabajar con las APIs, y el uso de GitHub. No solo aprendimos qué son teóricamente, sino que tuvimos la oportunidad de poner en práctica el conocimiento y enfrentarnos con los desafíos que eso conlleva.

En cuanto a los aspectos que no salieron como esperaban podemos hablar de los tiempos, cuando hicimos el diagrama de Gantt para el anteproyecto estimamos tiempos muy optimistas. En especial, no tuvimos en cuenta el tiempo que nos iba a llevar aprender a trabajar de forma eficiente en Thingsboard ni cuán intenso en términos de tiempo fue el testeo de los subsistemas. Asimismo, no consideramos que al correr del semestre nuestra disponibilidad temporal varía conforme se

presentan otras evaluaciones por lo que ,en retrospectiva, consideramos que estos periodos debieron haber sido tomados en cuenta como de menor actividad en la planificación del desarrollo del proyecto.

A su vez, nos sucedió que habíamos programado la funcionalidad del apagado de la climatización para las horas donde el cowork no estuviera activo en clase, pero debido a causas de fuerza mayor este código no quedó guardado en la nube. A causa del corto tiempo que quedaba una vez que lo notamos, no se pudo volver a implementar. Sin embargo, lo utilizamos como un método de aprendizaje para proyectos futuros de que más vale ser precavidos y siempre guardar un respaldo de todo el trabajo que se va haciendo.

En cuanto al uso de componentes electrónicos, pudimos superar los desafíos que se nos presentaron menos el principal que se había delineado en el anteproyecto: el control del Smart Film. Asumimos erróneamente que este componente iba a ser más fácil de controlar de lo que fue, y comenzamos a trabajar con él relativamente tarde en el transcurso del proyecto. Esto causó que no lo pudiéramos controlar como fue planteado anteriormente.

También subestimamos el armado de la maqueta, el hecho de hacer una estructura de MDF nos complicó más tiempo del pensado lograr encontrar la forma de poder lograr las paredes y los dos pisos que habíamos pensado. De la misma forma nos pasó con los agujeros que debíamos hacer para las conexiones de los componentes, o el caño que pasaba de un piso hacia otro.

Página web

https://pilibus12.wixsite.com/smart-cowork

Repositorio en GitHub

https://github.com/Conectando-Cosas-2022/Smart-Cowork

Bibliografía

Aosong Electronics Co. (n.d.). *DHT22 datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/1132459/ETC2/DHT22.html

Beegee-Tokyo. (2021, February 19). *DHTESP/examples/dht_esp32 at master · Beegee-Tokyo/DHTESP*. GitHub. Retrieved December 11, 2022, from https://github.com/beegee-tokyo/DHTesp/tree/master/examples/DHT_ESP32

DALLAS SEMICONDUCTOR. (n.d.). *DS18B20 Datasheet*. ALLDATASHEET. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html

García, S. (2020, June 17). *Cómo conectar un sensor de temperatura DS18B20 a Arduino*. 330ohms. Retrieved December 11, 2022, from https://blog.330ohms.com/2020/06/04/como-conectar-un-sensor-de-temperatura-ds18b20-a-arduino/

Mario's Ideas. (n.d.). *Tutorial on how to control 12V devices with Arduino*. Arduino Project Hub. Retrieved December 11, 2022, from https://create.arduino.cc/projecthub/mdraber/tutorial-on-how-to-control-12v-devices-with-arduino-e5b416?ref=user&ref_id=1474727&offset=6

NXP Semiconductors. (1999). *IRFZ44 datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/17808/PHILIPS/IRFZ44.html

NXP Semiconductors. (n.d.). *MFRC522 Datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/227839/NXP/MFRC522.html

Thingsboard. (2022). *Weather reading using REST API calls*. ThingsBoard. Retrieved December 12, 2022, from https://thingsboard.io/docs/user-guide/rule-engine-2-0/tutorials/get-weather-using-rest-api-call/

Thingsboard. (n.d.). *Enrichment nodes*. ThingsBoard. Retrieved December 12, 2022, from https://thingsboard.io/docs/user-guide/rule-engine-2-0/enrichment-nodes/#device-attributes

Xukyo. (2021, May 3). *Using an RFID module with an ESP32*. AranaCorp. Retrieved December 11, 2022, from https://www.aranacorp.com/en/using-an-rfid-module-with-an-esp32/

Zhengzhou Winsen Electronics Technology Co. (2015). *MQ135 datasheet*. Alldatasheet.com. Retrieved December 11, 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/1307647/WINSEN/MQ135.html