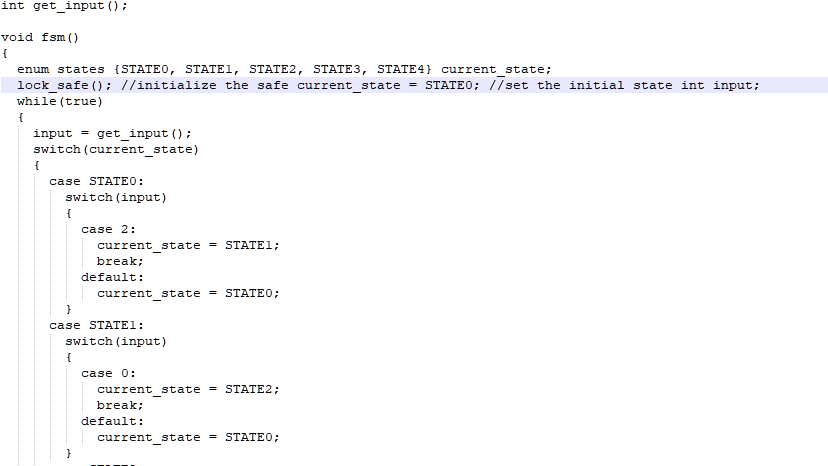
# Case Test ESP32

Este documento contiene los casos de pruebas realizados en ESP32

## **Enunciado:**

* Desarrollar un sistema embebido para ESP32 que utilice el concepto de Internet de las Cosas. Debe contener lo siguiente:
  + Utilizar al menos 2 sensores (1 analogico y 1 digital)
  + Utilizar al menos 2 actuadores (por lo menos 1 que use pwm y otro puede ser digital)
  + Utilizar el concepto de máquina de estados en el diseño del código. La máquina de estados debe seguir el patron switch anidados, en donde. Un switch por para cada estado y dentro de cada estado un switch para cada evento. Tal como se muestra en la figura.



* + Los eventos solo se deben obtener dentro de la función get\_input() y solo se pueden cambiar de estados dentro de la switch de la maquina de estados.
  + La maquina de estados debe tener 3 estados como mínimo.
  + Usar Logs para informar los cambios de estados
  + Se debe hacer el código descriptivo a documentando a nivel de funciones
  + Se debe usar el concepto de temporizadores por hardware o software
  + Se debe utilizar FreeRTOS para realizar tareas concurrentes para leer los sensores y activar actuadores
  + Debe poder enviar comandos datos de sensores a un bróker mqtt cada determinado tiempo
  + Debe poder recibir comandos desde un bróker mqtt que realice una acción en el embebido, preferentemente que active un actuador
  + La lógico de la comunicación mqtt debe cumplir el patron de máquina de estados.

## **PROMPTS UTILIZADOS:**

* **LENGUAJE DE PROGRAMACION**:
  + Utilizar lenguaje Wiring para el Ide Arduino
* **PLATAFORMA:**
  + Desarrollar un sistema embebido ESP32 que utilice el concepto de Internet de las Cosas.
* **FUNCIONALIDAD A IMPLEMENTAR:**
  + Desarrollar un sistema embebido que controle una casa inteligente que controle la temperatura ambiente
* **REQUISITOS ESPECIFICOS:**
  + se deben usar un sensor de temperatura analogico, dos pulsadores, un rele que controle un ventilador y un buzzer pasivo
  + El sistema debe tener dos modos de uso: uno manual y otro automatico
  + Si el sistema se encuentra en modo manual, al presionar el pulsador 1 se debe cambiar a modo automatico
  + Si el sistema se encuentra en modo automatico, al presionar el pulsador 1 se debe cambiar a modo manual
  + Cuando el sistema esta modo automatico, se debe encender el rele si la temperatura supera determinado umbral
  + Cuando el sistema esta modo automatico, se debe apagar el rele si la temperatura es menor a determinado umbral
  + Cuando el sistema esta modo manual, se debe encender el rele al presionar el pulsador 2
  + Cuando el sistema esta modo manual, se debe apagar el rele al presionar el pulsador 2.
  + Cada vez que se enciende el rele se debe reproducir una melodía usando el buzzer
  + Utilizar Tareas FreeRTOS para leer los sensores y activar actuadores
  + Utilizar temporizadores por hardware o software para reproducir la melodía en una tarea de FreeRTOS
  + Utilizar el concepto de máquina de estados en el diseño del código
  + La maquina de estados debe tener 3 estados como mínimo.
  + Usar Logs para informar los cambios de estados
  + Se debe hacer el código descriptivo a documentando a nivel de funciones
* **RESTRICCIONES:** 
  + La máquina de estados debe seguir el patron switch anidados, en donde exista un switch para cada estado y dentro de cada estado un switch para cada evento. Tal como se muestra en la figura anidada
  + Los eventos solo se deben obtener dentro de la función get\_input() y solo se pueden cambiar de estados dentro de la switch de la maquina de estados.
  + Los sensores solamente se deben leer dentro de la función get\_input y no debe haber sentencias if dentro de los switch de la maquina de estados

**AGREGADO DE CODIGO MQTT EN EL RESULTAOD OBTENIDO PARA ESP32**

modifica el código generado para que realice lo siguiente:

* + El sistema se debe conectar a un bróker mqtt
  + La lógica de la comunicación mqtt debe cumplir el patron de máquina de estados implementado .
  + Se debe suscribir al topico /casa/cmd
  + Dependiendo del valor recibido en el topico /casa/cmd,se encenderá o apagara el rele
  + Cada determinado tiempo se debe publicar la temperatura medida del sensor, en el topico /casa/temperatura del borker mqtt

## RESULTADOS OBTENIDOS CON CHATGPT

1. En la primera iteración se observo que no se sigue correctamente el patron de maquina de estados ya que se generaban leen los estados dentro del get\_event y se genran if dentro de los switch.
2. Además se. Se generó el código pero sin que aplique mecanismo de sincronización en la variables compartidas en entre las tareas. Por lo que hubo que refinar el prompt solicitándole que le agregue esa mecanismo.
3. En la segunda iteración se noto que el código de pwm se generó usando la versión de la biblioteca de ledcwrite menor a 2.0 por lo que wokwi no esta preparado para eso. Por hubo que volver a refinar el código solictandole que lo actualice. Dado que chatgpt no lo realizaba correctamente hubo que realizar la correción a mano.
4. Por lo que hubo que volver a realizar una refinación porque no genero la función loop()
5. Otro de los problemas es la tarea de reproducir el buzzer lo genero de manera bloqueate en la tarea principal. Por lo que bloquea todo el código al reproducir la melodía. Por lo que hubo que volver a realizar otra refinación del código generado.
6. Luego de varias refinaciones se pude conseguir que funcione el sistema pero no se respetan el patron de maquina de estados solicitado.

**RESULTADO OBTENIDO AL AGREGAR LA UTILIZACIÓN DE MQTT**

Se genero el código modificado con mqtt, pero presenta determinados retrasos en el envio al bróker mqtt y al la recpcion de datos del mismo. Que dificulta el uso del mismo

Para realizar las pruebas se utilizo el bróker de emqx

El código generado se encuentra en

<https://wokwi.com/projects/434925575898614785>

## RESULTADOS OBTENIDOS CON GITHUB COPILOT

1. Genera un código más performante que chatgpt, pero sigue siendo no optimo. ya que genera código bloqueante y no sigue el patron de maquina solicitado, siguiendo. Habiando if dentro de loc cases de la switch.
2. Elcodigo sin mqqt funciona bien en Wowki pero con esos problemas de performance, pero en el sistema físico no funciona del todos bien. Además no genera de forma optima el código.

**RESULTADO OBTENIDO AL AGREGAR LA UTILIZACIÓN DE MQTT**

1. Al realizar el código con mqtt no funciona bien el código fuente, debido a que genera diversos problemas en la comunicación con el bróker.
2. Luego de muchos refinamientos se pudo hacer funcionar a medias, dado que aveces genera problemas de conectividad.

## RESULTADOS OBTENIDOS CON CURSOR

1. Genera un código aceptable sin mqtt funcional, que hubo que refinar varias veces. Siguio el patron de máquina de estados solicitados. Pero no genero mecanismos de sincronización entre las variables globales usadas en las diferentes tareas. Por lo que hubo que solictarlas.
2. Al aplicar la conexión con el bróker mqtt, se generaron varios problemas que dejaron de funcionar lo que anteriormente funcionaba. Por lo que hubo que hacer varios refinamientos para que vulvan a funcionar en conjunto con la conectivada de mqtt.
3. Si bien se pudo hacer que funcione la conectividad con mqtt y que se reciban y envíen comandos. Se debe realizar muchos refienamientos para funcionen adecuadamente.

## RESULTADOS OBTENIDOS CON WINDSURF

1. Genera un código aceptable sin mqtt funcional, que hubo que refinar varias veces. Siguio el patron de máquina de estados solicitados. Pero no genero mecanismos de sincronización entre las variables globales usadas en las diferentes tareas. Por lo que hubo que solictarlas.
2. Su realización de código funciono mejor que con otro llm tales como cursor, githubcopilot. Si bien hay aspectos que falla con respecto al uso del patron de maquina de estados. Su código es funcional
3. Al agregar el código de conexión mqtt su generación de código resulto mejor que con cursor. Ya que funciono de una y hubo que hacer muy poco refinamiento. Aparte el funcionamiento del programa con mqtt funciona mejor que con cursor
4. Se utilizo el model SWE-1 (free limited time)