PLANO DE TESTES

NOME DA EQUIPE: Koala
PARTICIPANTES: Eduardo Izidorio, Kamila Leite, Lucas Anderson, Yhasmim Ferreira

Este modelo pode ser adaptado conforme necessário para atender aos requisitos específicos do projeto.

1. Introdução

Objetivo: Este documento define o plano de teste e os casos de teste para o Projeto MedBox: Sistema de compartimentos inteligentes, com o intuito de verificar a funcionalidade, desempenho, segurança e confiabilidade dos dispositivos e sistemas implementados.

Escopo: Os testes cobrem os principais componentes e funcionalidades do sistema, incluindo a integração de sensores, atuação de dispositivos e a comunicação entre a plataforma ESP32 e o servidor.

2. Estratégia de Teste

Metodologia: A metodologia utilizada será baseada em testes manuais e automáticos, com foco em testes funcionais, de integração, de desempenho e de segurança.

Ambiente de Teste:

- Dispositivos: ESP32
- Ferramentas:Arduino IDE

Responsáveis pelo teste: Eduardo Izidorio

3. Casos de Teste

#Caso de Teste 1: Sensor de Distância Ultrassônico (HC-SR04)

- **ID: CT-001**
- **Descrição:** Validar se o sensor de distância ultrassônico (HC-SR04) mede a distância corretamente e envia os dados para o ESP32.
- **Pré-condição:** Sensor HC-SR04 corretamente conectado ao ESP32 e configuração do código de leitura do sensor.
- **Passos de Teste:**
    1. Ligar o ESP32 e garantir que o sensor HC-SR04 esteja corretamente alimentado.
    2. Testar a medição de distância com o sensor (variando a distância do objeto).
    3. Verificar se a leitura de distância no console serial do :ESP32 corresponde à distância real medida com uma régua ou outro método de medição.
    4. Enviar os dados de distância para o servidor via Wi-Fi (conforme configurado no Caso de Teste 2).
    5. Verificar a recepção e a precisão dos dados de distância no servidor.
- **Resultado Esperado:** O sensor deve medir a distância corretamente, e os dados devem ser enviados sem erros para o servidor.
- **Resultado Real:** Erro nos pinos do código que não estava colocado de maneira correta. Após essa correção, tudo funcionou corretamente.
- **Status:** PASSOU

###Caso de Teste2: Sensor de Temperatura e Umidade DHT11

- **ID: CT-002**
- **Descrição:** Validar se o sensor DHT11 mede corretamente a temperatura e a umidade, e envia os dados para o ESP32.
- **Pré-condição:** Sensor DHT11 corretamente conectado ao ESP32 e configuração do código de leitura do sensor.
- **Passos de Teste:**
    1. Ligar o ESP32 e garantir que o sensor DHT11 esteja corretamente alimentado.

2. Testar a medição de temperatura e umidade com o sensor (variando as condições ambientais, como alterar a temperatura ou umidade).
3. Verificar se os valores de temperatura e umidade exibidos no console serial do ESP32 correspondem à medição real com um termômetro e higrômetro de referência.
4. Enviar os dados de temperatura e umidade para o servidor via Wi-Fi (conforme configurado no Caso de Teste 2).
5. Verificar a recepção e a precisão dos dados de temperatura e umidade no servidor.

- **Resultado Esperado:** O sensor deve medir a temperatura e umidade corretamente, e os dados devem ser enviados sem erros para o servidor.
- **Resultado Real:** Atendeu a todos os requisitos
- **Status:** PASSOU

###Caso de Teste 3: Buzzer e LEDs

- **ID: CT-003**
- **Descrição:** Validar o funcionamento do buzzer e dos LEDs, garantindo que acionem corretamente conforme comandos do ESP32.
- **Pré-condição:** Buzzer e LEDs corretamente conectados ao ESP32, e código de controle configurado.
- **Passos de Teste:**
    1. Ligar o ESP32 e garantir que o buzzer e os LEDs estejam conectados corretamente.
    2. Testar o acionamento do LED (ligar e desligar) em resposta a comandos do ESP32.
    3. Verificar se os LEDs acendem com a cor e intensidade esperadas quando o comando é dado (por exemplo, LED vermelho acende em caso de erro, LED verde para sucesso).
    4. Testar o acionamento do buzzer, verificando se ele emite som ao ser ativado e desativa quando o comando correspondente é dado.

5. Verificar a resposta do buzzer e dos LEDs ao envio de um sinal do servidor, por exemplo, ativar o buzzer quando um valor de temperatura fora do intervalo permitido for detectado.

- **Resultado Esperado:** O buzzer deve emitir som quando ativado e os LEDs devem acender corretamente de acordo com os comandos fornecidos pelo ESP32.
- **Resultado Real:** Um dos LEDs não acendeu durante o teste, ao verificar foi notado que o jumper não estava funcionando corretamente, foi trocado. Após isso ambos os LEDs funcionaram corretamente, assim como o buzzer.
- **Status:**PASSOU

###Caso de Teste 4: Conexão com Servidor Web

- **ID: CT-004**
- **Descrição:** Validar se o ESP32 consegue se conectar a um servidor web e enviar/receber dados de forma confiável.
  Pré-condição: Servidor web configurado e ativo, com o ESP32 conectado à rede Wi-Fi.
- **Passos de Teste:**
  1. Ligar o ESP32 e garantir que ele está conectado à rede Wi-Fi.
  2. Verificar a configuração do servidor web (por exemplo, um servidor MQTT local ou na nuvem) para garantir que ele está preparado para receber e responder a requisições.
  3. Enviar uma requisição MQTT do ESP32 para o servidor, usando um método GET ou POST (dependendo da configuração do servidor).
  4. Verificar se a resposta do servidor é recebida corretamente pelo ESP32 (por exemplo, código MQTT 200 de sucesso ou dados em formato JSON).
  5. Testar o envio de dados do ESP32 para o servidor, por exemplo, enviando dados de sensores (como temperatura e umidade) para o servidor via requisição POST.
  6. Verificar no servidor se os dados recebidos do ESP32 estão corretos e são processados de maneira adequada.

7. Realizar um teste de desconexão e reconexão com o servidor, simulando uma perda de rede.
8. Testar a reconexão automática do ESP32 ao servidor após uma falha de conexão.

- **Resultado Esperado:** O ESP32 deve ser capaz de se conectar ao servidor web, enviar e receber dados com sucesso, e reconectar automaticamente em caso de falha de conexão.
- **Resultado Real:** Atendeu a todos os requisitos
- **Status:** PASSOU

###Caso de Teste 5: Integração de LEDs, Sensor Ultrassônico e Buzzer

- **ID: CT-005**
- **Descrição:** Validar a integração do sensor ultrassônico com LEDs e buzzer, de modo que os LEDs e o buzzer acionem quando a lembrete do remédio for enviado do app mobile.
- **Pré-condição:** Sensor ultrassônico (HC-SR04), LEDs e buzzer corretamente conectados ao ESP32, e código configurado para ler dados do sensor e controlar os LEDs e o buzzer.
- **Passos de Teste:**
    - Ligar o ESP32 e garantir que o sensor ultrassônico, LEDs e buzzer estão corretamente conectados e configurados.
    - Testar a medição de distância com o sensor ultrassônico e verificar se o valor de distância é exibido corretamente no console serial do ESP32.
    - Configurar o código para acionar os LEDs e o buzzer com base no envio do lembrete pelo app mobile.
    - Exemplo:
    - Dado alerta de mensagem os leds e buzzer são acionados, e só são desligados a aproximação de 5cm do sensor ultrassônico.
    - Verificar se os LEDs acendem corretamente (nas cores definidas) e o buzzer emite os sons esperados conforme a distância medida.

    - Realizar testes de variação de distância (por exemplo, aproximando e afastando objetos) e verificar se a resposta dos LEDs e do buzzer é imediata e correta.
- **Resultado Esperado:** O ESP32 deve acionar os LEDs e o buzzer conforme o recebimento do lembrete no app mobile.
- **Resultado Real:** Atendeu a todos os requisitos
- **Status:** PASSOU

###Caso de Teste 6: Conexão do app mobile com o servidor na nuvem

ID: CT-006

Descrição: Validar a integração do aplicativo mobile com o servidor MQTT, garantindo que os dados sejam enviados e recebidos corretamente, com acionamento de LEDs e buzzer conforme o envio do lembrete de remédio.

Pré-condição: O aplicativo mobile deve estar configurado para se conectar ao servidor MQTT, com as credenciais e tópicos corretamente configurados. O dispositivo deve estar conectado ao servidor MQTT, com a assinatura do tópico correspondente.

Passos de Teste:

1. **Ligar o servidor MQTT**:
    - O servidor MQTT deve estar funcionando e acessível.
2. **Configurar e enviar o lembrete do app mobile**:
    - Enviar o lembrete de remédio para o tópico MQTT configurado.
3. **Verificar a recepção da mensagem no dispositivo receptor**:
    - O dispositivo (ESP32) deve receber a mensagem do lembrete corretamente.
4. **Acionar LEDs e buzzer**:
    - LEDs devem acender e o buzzer deve emitir som conforme a mensagem recebida.
5. **Testar variação de distância do sensor ultrassônico**:
    - Aproximar e afastar objetos do sensor e verificar a resposta dos LEDs e buzzer.
6. **Testar reconexão ao servidor MQTT**:

- Simular falha de conexão e verificar se o dispositivo reconecta automaticamente.

7. **Verificar o tempo de resposta**:
   - Medir o tempo entre o envio do lembrete e o acionamento dos LEDs/buzzer.

Resultado Esperado:

- O sistema deve enviar, receber dados e acionar LEDs/buzzer corretamente.
- O tempo de resposta deve ser inferior a 3 segundos.
- O dispositivo deve reconectar automaticamente após falha.

Resultado Real: O teste atendeu a todos os requisitos. O aplicativo mobile enviou o lembrete com sucesso, o servidor MQTT processou a mensagem corretamente, e o dispositivo (ESP32) acionou os LEDs e buzzer conforme esperado. A reconexão foi bem-sucedida após falha de conexão e o tempo de resposta foi inferior a 3 segundos.
STATUS: PASSOU

###Caso de Teste 7: Integração do app com o banco de dados
ID: CT-007
Caso de Teste: Integração do App com o Banco de Dados
Descrição: Validar a integração do banco de dados com o servidor MQTT na nuvem, garantindo que o ESP32 consiga consumir os dados do banco de dados para a emissão dos alertas de remédios.
Pré-condição: O aplicativo mobile deve estar configurado para se conectar ao servidor MQTT e ao banco de dados. O banco de dados deve estar acessível e corretamente configurado para armazenar e fornecer os lembretes. O dispositivo ESP32 deve estar conectado ao servidor MQTT e assinado no tópico correspondente.
Passos de Teste:
1. Ligar o servidor MQTT: O servidor MQTT deve estar funcionando e acessível.
2. Conectar o app mobile ao servidor MQTT: Verificar se a conexão é estabelecida corretamente.
3. Cadastrar um lembrete de remédio no app mobile: O lembrete deve ser salvo no banco de dados e enviado para a nuvem.

4. Consumir os dados do banco de dados pelo ESP32: O dispositivo deve acessar o banco de dados e receber os lembretes corretamente.

5. Verificar a recepção da mensagem no dispositivo receptor: O ESP32 deve processar a mensagem corretamente.

6. Acionar LEDs e buzzer: Os LEDs devem acender e o buzzer emitir som conforme a mensagem recebida.

7. Testar reconexão ao servidor MQTT: Simular uma falha de conexão e verificar se o dispositivo reconecta automaticamente.

8. Verificar o tempo de resposta: Medir o tempo entre o cadastro do lembrete e o acionamento dos LEDs/buzzer.

Resultado Esperado: O usuário cadastra o remédio via app mobile, o lembrete é salvo no banco de dados e enviado à nuvem. O dispositivo receptor (ESP32) consome os dados e aciona os LEDs e buzzer corretamente. O tempo de resposta deve ser inferior a 3 segundos. O dispositivo deve reconectar automaticamente após falha.

Resultado Real: O banco de dados não estava registrando os remédios corretamente, causando falhas no cadastro dos lembretes. Houve problemas com dependências no desenvolvimento do app, dificultando a comunicação com o banco de dados. Após corrigir e atualizar as versões das dependências, conseguimos estabelecer a comunicação correta, garantindo o registro e envio dos dados ao servidor.

STATUS: PASSOU

##Caso de Teste 8: Integração do Sensor com o App Mobile

ID: CT-008

Caso de Teste: Integração do Sensor com o App Mobile

Descrição: Verificar se os dados coletados pelo sensor conectado ao ESP32 são enviados corretamente para o aplicativo mobile via servidor MQTT, garantindo o monitoramento em tempo real.

Pré-condições:

- O sensor deve estar corretamente conectado ao ESP32 e funcional.
- O ESP32 deve estar conectado ao servidor MQTT e ao tópico correto.

- O aplicativo mobile deve estar configurado para receber dados do tópico MQTT.

Passos de Teste:

1. Ligar o ESP32 com o sensor acoplado e garantir sua inicialização correta.
2. Estabelecer a conexão do ESP32 com o servidor MQTT.
3. Simular a leitura de dados pelo sensor (ex: temperatura, umidade, ou outro parâmetro).
4. Verificar o envio dos dados do sensor via MQTT para a nuvem.
5. Confirmar o recebimento dos dados pelo app mobile em tempo real.
6. Validar se os dados recebidos são exibidos corretamente na interface do app.
7. Simular uma perda de conexão com o servidor MQTT e testar a reconexão automática.
8. Verificar a atualização contínua dos dados após a reconexão.

Resultado Esperado:

- Os dados do sensor são enviados corretamente pelo ESP32 via MQTT.
- O aplicativo mobile recebe e exibe os dados em tempo real.
- Em caso de perda de conexão, o sistema reconecta automaticamente e retoma o envio dos dados.
- O intervalo de atualização não deve exceder 2 segundos após a leitura do sensor.

Resultado Real:

- A conexão inicial entre o sensor e o ESP32 foi bem-sucedida.
- Os dados estavam sendo enviados ao servidor MQTT, mas o app não exibia as informações.
- Após ajustes no mapeamento do tópico no app e correção de parâmetros de leitura, os dados passaram a ser recebidos corretamente.
- A reconexão automática funcionou conforme o esperado, com recuperação dos dados em tempo real

STATUS: PASSOU

Caso de Teste 9: Respostas dos LEDs às Mensagens do MQTT

ID: CT-009

Caso de Teste: Respostas dos LEDs às Mensagens do MQTT

Descrição: Validar se os LEDs do dispositivo ESP32 respondem corretamente às mensagens recebidas via protocolo MQTT, de acordo com os comandos e identificações dos lembretes de medicamentos.

Pré-condições:

- O ESP32 deve estar conectado ao servidor MQTT e ao tópico de escuta adequado.
- O aplicativo mobile deve estar funcional e capaz de publicar mensagens no tópico correto.
- Os LEDs devem estar corretamente conectados às saídas digitais do ESP32.

Passos de Teste:

1. Estabelecer conexão do ESP32 ao servidor MQTT.
2. Acessar o aplicativo mobile e cadastrar um lembrete com identificação específica de compartimento.
3. Enviar manualmente uma mensagem MQTT simulando o acionamento de um compartimento específico (ex: compartimento 2).
4. Observar a resposta do LED correspondente no ESP32.
5. Repetir o envio para diferentes compartimentos e verificar se os LEDs corretos são acionados.
6. Simular o envio de uma mensagem inválida ou fora do padrão.
7. Verificar se o sistema ignora ou trata a mensagem sem acionar LEDs indevidos.
8. Medir o tempo de resposta entre o envio da mensagem e o acionamento do LED.

Resultado Esperado:

- O LED correspondente ao compartimento indicado na mensagem MQTT acende corretamente.
- Mensagens inválidas não acionam LEDs ou geram notificações de erro no log.

- O tempo de resposta entre o envio da mensagem e o acionamento do LED deve ser inferior a 2 segundos.

Resultado Real:

- Os LEDs foram acionados corretamente de acordo com as mensagens válidas recebidas via MQTT.
- Mensagens malformadas foram corretamente ignoradas.
- O tempo médio de resposta registrado foi de 1,2 segundos, dentro do esperado.
- O comportamento do sistema permaneceu estável mesmo após múltiplas mensagens sequenciais.

STATUS: PASSOU

Caso de Teste 10: Integração do Smartwatch com o Broker MQTT (HiveMQ)

ID: CT-010

Caso de Teste: Integração do Smartwatch com o Broker MQTT (HiveMQ)

Descrição: Validar se o smartwatch está corretamente integrado ao broker MQTT (RiveMQ), sendo capaz de receber e interpretar mensagens de lembretes enviadas pelo servidor e/ou aplicativo mobile.

Pré-condições:

- O smartwatch deve estar emparelhado com o smartphone (caso necessário).
- O smartwatch deve ter conectividade ativa (Wi-Fi ou Bluetooth com acesso à internet).
- O broker MQTT RiveMQ deve estar ativo e acessível.
- O aplicativo mobile deve estar funcional e configurado para publicar mensagens MQTT no tópico adequado.

Passos de Teste:

1. Garantir que o smartwatch esteja conectado à rede e com o app receptor de mensagens MQTT em execução.
2. Acessar o aplicativo mobile e cadastrar um lembrete com horário e identificação.

3. Enviar uma mensagem MQTT simulando o horário de acionamento do lembrete.
4. Verificar se o smartwatch recebe a notificação no tempo esperado.
   Avaliar a exibição da mensagem no display do smartwatch (texto correto, clareza, etc.).
5. Testar o envio de múltiplas mensagens seguidas (para lembretes diferentes).
6. Verificar o comportamento do smartwatch em caso de perda momentânea de conexão (por exemplo, desligar Wi-Fi e religar em seguida).
7. Enviar uma mensagem MQTT fora do padrão esperado e observar a reação do sistema.

Resultado Esperado:

- O smartwatch recebe e exibe corretamente as mensagens MQTT referentes aos lembretes.
- Mensagens inválidas são ignoradas ou geram notificações de erro visíveis no app ou log do sistema.
- O tempo entre o envio da mensagem e a notificação no smartwatch deve ser inferior a 3 segundos.
- Em caso de reconexão à rede, o smartwatch deve retomar a escuta do tópico MQTT automaticamente.

Resultado real

Durante a execução do Caso de Teste 10, que visa validar a integração do smartwatch com o broker MQTT (RiveMQ), foram encontrados alguns problemas iniciais que impediram o sucesso do teste logo de início. A função responsável por receber as mensagens MQTT no smartwatch estava desatualizada e incompatível com a versão atual do protocolo implementado. Além disso, o smartwatch estava descarregado, o que impediu o recebimento das notificações no momento da primeira tentativa. Também foi identificado que a biblioteca MQTT utilizada pelo aplicativo apresentava instabilidade temporária, ocasionando falhas na publicação das mensagens para o broker.

Após a identificação dos problemas, as correções necessárias foram aplicadas: a função de recepção MQTT no smartwatch foi atualizada, o dispositivo foi carregado adequadamente, e a biblioteca instável foi substituída por uma versão estável. Com essas correções, o teste foi repetido e o smartwatch passou a receber e exibir corretamente todas as mensagens dos lembretes, dentro do tempo esperado (inferior a 3 segundos). Mensagens fora do padrão também foram ignoradas corretamente, conforme o comportamento esperado.

STATUS: PASSOU

Caso de Teste 11: Integração do Protótipo com o Aplicativo

ID: CT-011

Caso de Teste: Integração do Protótipo com o Aplicativo

Descrição: Verificar se o protótipo físico (ESP32 com LEDs e demais componentes) está se comunicando corretamente com o aplicativo mobile, garantindo a troca de informações via protocolo MQTT e o funcionamento esperado do sistema de lembretes.

Pré-condições:

- O ESP32 deve estar conectado ao broker MQTT.
- O aplicativo deve estar funcional e conectado ao mesmo broker MQTT.
- A conexão entre o ESP32 e o hardware (LEDs, botões, sensores, etc.) deve estar configurada corretamente.
- O dispositivo mobile deve ter conexão à internet e permissões adequadas.

Passos de Teste:

1. Iniciar o protótipo e o aplicativo mobile.
2. Garantir a conexão de ambos com o broker MQTT.
3. Cadastrar um lembrete de medicamento no aplicativo com horário e compartimento definidos.
4. Aguardar o envio automático da mensagem MQTT no horário agendado.

5. Verificar se o compartimento correspondente no protótipo é acionado corretamente (ex: LED aceso, motor vibrando, etc.).

6. Confirmar no aplicativo que a ação foi executada (ex: status atualizado ou confirmação de recebimento).
7. Interagir com o protótipo (por exemplo, pressionar botão para confirmar o uso do medicamento).
8. Verificar se o aplicativo recebe o feedback do protótipo corretamente.
9. Simular perda de conexão do dispositivo mobile e verificar se há reconexão automática e recuperação da comunicação.

Resultado Esperado:

- O protótipo responde corretamente aos comandos enviados pelo aplicativo.
- A troca de mensagens entre aplicativo e protótipo ocorre sem falhas ou atrasos superiores a 2 segundos.
- Feedbacks físicos (como LEDs, motores, botões) funcionam conforme o esperado.
- O aplicativo reconhece ações realizadas no protótipo e atualiza o status do lembrete.
- O sistema lida corretamente com desconexões temporárias, retomando a comunicação assim que possível.

Resultado real

- Mensagem MQTT enviada e recebida com sucesso, mas os sensores (ou atuadores, como LEDs/motores) não foram ativados.
-

4. Critérios de Aprovação

- Funcionalidade: Todos os casos de teste funcionais devem ser aprovados.
- Desempenho: O tempo de resposta dos dispositivos não deve exceder o limite especificado.

- Segurança: Nenhuma vulnerabilidade crítica deve ser encontrada.
- Resiliência: O sistema deve retomar a comunicação após falhas de rede sem perda de dados.

5. Conclusão

- **Resumo dos Resultados**

Durante a execução dos testes do Projeto MedBox, foram identificadas algumas falhas que foram corrigidas ao longo do processo. No teste do Sensor Ultrassônico (CT-001), houve um erro nos pinos do código, mas após a correção, o sensor funcionou corretamente. O Sensor de Temperatura e Umidade (CT-002) atendeu a todos os requisitos sem problemas. No teste do Buzzer e LEDs (CT-003), um LED não acendeu devido a um jumper defeituoso, que foi substituído para resolver o problema.

A Conexão com o Servidor Web (CT-004) ocorreu conforme esperado, garantindo a comunicação do ESP32 com o servidor. Já no teste de Integração do Sensor Ultrassônico, LEDs e Buzzer (CT-005), o sistema respondeu corretamente ao envio do lembrete, acionando os alertas adequadamente.

Na Conexão do App Mobile com o Servidor MQTT (CT-006), todos os requisitos foram atendidos, garantindo que os dados fossem enviados, recebidos e processados corretamente, com tempo de resposta inferior a 3 segundos e reconexão automática após falhas.

Por fim, na Integração do App com o Banco de Dados (CT-007), houve falhas iniciais no registro de lembretes devido a problemas com dependências do aplicativo. Após a atualização das versões e ajustes na comunicação com o banco de dados, os lembretes passaram a ser cadastrados corretamente, garantindo que o ESP32 consumisse os dados e acionasse os alertas conforme esperado.

No geral, todos os testes foram bem-sucedidos após as correções necessárias, garantindo a funcionalidade e confiabilidade do sistema.