- CAPA

RESUMO

Este trabalho de desenvolvimento apresenta um projeto unindo o Free RTOS ( Real Time Operating System) e conectividade IOT (Internet of Things).

Foi desenvolvido um regador automatizado, com interface de comunicação via Wi-fi, utilizando o protocolo MQTT (Message Queue Telemetry Transport) para envio de informações para uma plataforma IOT.

O trabalho conta também com o fluxo de tarefas no sistema operacional, utilizando os conceitos de tasks, semáforos e filas, além de sensores de monitoramento de umidade e ligação do regador.

Palavras-chave: Regador, IoT, ESP32, MQTT, Free-RTOS, Wi-fi, Monitoramento

ABSTRACT

INTRODUÇÃO

Com a industrialização e urbanização das cidades, um tema que vem sendo cada vez mais estudado a cada ano é o desperdício e a limitação dos recursos naturais.

Dentre os recursos naturais mais abundantes do planeta, a água, na condição de potável, é também um dos recursos mais que vem diminuindo a cada dia, por vários fatores, entre eles:

* ocupação de áreas de manancial;
* desmatamento;
* degradação das nascentes
* efeitos climáticos.

Tanto no meio urbano, quanto no rural, uma forma de conter este desperdício é controlando o uso da água, de forma racional.

A irrigação é um dos grandes utilizadores da água no meio rural e também está presente no meio urbano, em jardins públicos e privados, campos de futebol, hortas e pomares.

Por outro lado, com o surgimento de novas tecnologias, surgem novos pontos de vista sobre hábitos e costumes que não acompanharam a modernidade, com novas soluções para o que antes não era nem considerado um problema.

Por meio do desenvolvimento e prova de conceito de um regador que se comunica com um broker MQTT através da rede Wi-fi, utilizando placa de desenvolvimento ESP32 com o sistema operacional FREE-RTOS, e passando informações relevantes para serem analisadas, este trabalho tem como finalidade a diminuição do uso desnecessário de água, aliada com novos aprendizados e novas tecnologias.

METODOLOGIA

Os materiais utilizados para o desenvolvimento do regador

Placa ESP32

Display

Relé

Bomba?

Sensor de Umidade do Solo

Principais softwares e bibliotecas utilizados no desenvolvimento

MQTT

LWIP

FREE-RTOS

ESP-IDF ou VS Code

WI-FI Manager

Linguagem C

Windows

CONCEITOS DE FREE-RTOS

Desenvolvido por volta de 2003, o Free-RTOS é hoje um dos principais sistemas operacionais para microprocessadores e microcontroladores. Por ser uma ferramenta de código aberto, também é um dos mais confiáveis e fáceis de usar.

Uma das principais características deste sistema operacional é seu diagrama de estados, controlado pelo agendador de tarefas.

O agendador de tarefas, ou escalonador, decide qual tarefa será executada, examinando a prioridade que foi atribuída a cada uma.

A imagem a seguir mostra como funciona o fluxo de tarefas no Free-RTOS.

Diagrama

Descrição gerada automaticamente

Diagrama de transição de estado entre as tarefas. Fonte: Site FreeRTOS.org

Uma tarefa pode fluir entre um dos seguintes estados:

* Ready – Quando a tarefa está pronta para ser executada e está na fila;
* Running – Quando a tarefa está sendo executada e está utilizando o processador;
* Blocked – Quando a tarefa está esperando um evento externo ou em modo de espera, iniciaod por um temporizador. Tarefas no estado de bloqueio normalmente tem um período de timeout, e são desbloqueadas depois desse período. Não podem ser selecionadas para entrar no estado de “Running”;
* Suspended – Tarefas que são suspendidas por meio de comandos como vTaskSuspend().

Estas tarefas neste estado não tem um tempo para expirar, saindo do estado somente com o comando vTaskResume()

Num ambiente de desenvolvimento de software, a primeira tarefa a ser executada, depois da inicialização, é a função principal. Dentro dela,

são criadas outras tarefas com prioridade mais alta, o que garante que ea só será executada uma vez.

Cada tarefa criada já entra em execução automaticamente e é importante equilibrar as prioridades entre as tarefas para que os recursos seja distribuídos entre elas de acordo com

a necessidade que cada uma tem de ser executada no tempo certo.

Uma simples inserção de atraso dentro da tarefa é importante para que o agendadorde tarefas a coloque em bloqueio por aquele período, possibilitando a chamada de outras tarefas e evitando a monopolização dos recursos para umaúnica tarefa.

O FREE-RTOS possui uma infinidade de API's que dão um toque especial ao sistema. Algumas das API's utilizadas são mostradas a seguir:

xTaskCreate

vTaskDelete

vTaskDelay

xQueueCreate

xQueuePeek

xQueueSend

xSemaphoreTake

xSemaphoreGive

vSemaphoreDelete

xSemaphoreCreateBinary

xSemaphoreGiveFromISR

xEventGroupCreate

xTaskCreatePinnedToCore

Protocolo MQTT

O MQTT (Message Queue Telemetry Transport) é um protocolo utilizado para comunicação de uma máquina para outra. Sua utilização vem crescendo a cada dia,

principalmente com o crescimento da Internet das Coisas.

O protocolo permite a troca de mensagens entre dispositivos, possibilitando o envio de informações e de comandos de um para o outro.

Os dispositivos precisam ser acesso a um servidor MQTT, que intermedia a comunicação, conhecido como Broker.

Sendo assim, um dispositivo pode fazer tanto um envio de informações para o Broker, como o assinar recebimento de informações.

As informações são enviadas e recebidas através de canais, que são criados por tópico. O dispositivo que quer receber, faz a assinatura do canal naquele tópico,

enquanto que o dispositivo que quer enviar, simplesmente envia as informações.

O conteúdo da mensagem pode ser a leitura de um sensor, informações de estado de um interruptor, por exemplo, ou até mesmo um comando para ligar/desligar

determinado periférico do dispositivo.

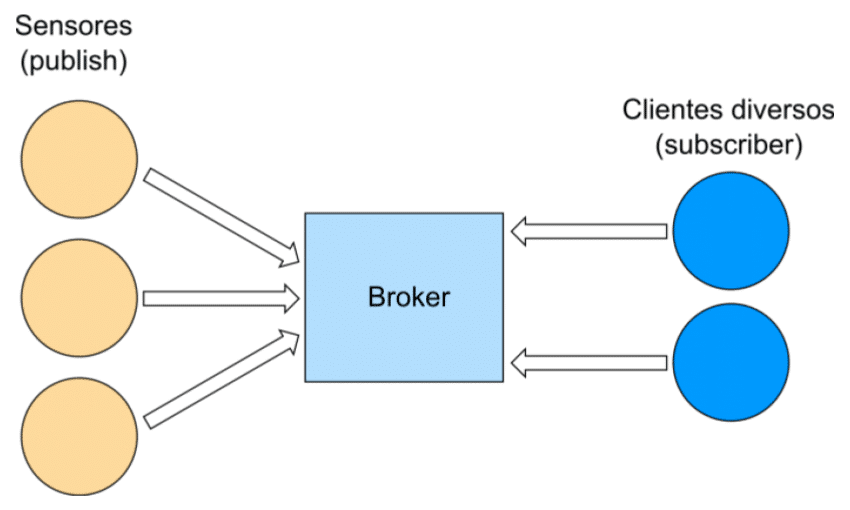
As informações do broker também pode ser acessadas por outros dispositivos, como computadores, que podem fazer o tratamento desses dados e inserí-los numa interface homem-máquina, de modo que facilite a manipulação, visualização e armazenamento dos dados, além de possibilitar uma maior

interação do seu operador, através da inserção de comandos, com a visualização rápida das respostas, de acordo com o intervalo das mensagens enviadas e recebidas.

Um dos servidores MQTT mais conhecidos é o Mosquitto, que foi utilizado na neste trabalho como parte dos testes locais.

Como se trata de transferência de dados basicamente através do stack http, grande parte da mensagem é trafegada utilizando a formatação Json e a comunicação

também pode ser feita de forma segura, com certificados SSL/TLS e autenticação no servidor.



Fonte: Site Embarcados

publish subscribe example

Fonte:Site CloudMQTT

Biblioteca WI-FI Manager

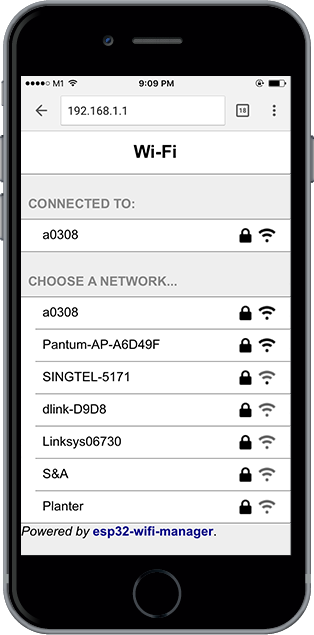
A biblioteca Wi-Fi Manager foi utilizada para facilitar a conexão do usuário, através do ESP32, à rede wi-fi, sem a necessidade da inserção da rede do usuário no código fonte.

O usuário se conecta ao ponto de acesso wi-fi criado pelo microcontrolador e acessa o endereço padrão configurado. No caso, foi utilizado o endereço de ip: 10.10.0.1 para se ter acesso à página html do wi-fi manager hospedada pelo microcontrolador.

A partir daí, o usuário seleciona a rede de acesso, entra com as credenciais, e o MCU se conecta diretamente à rede.

O Wi-Fi Manager foi incorporado ao projeto, como um componente externo.

Tela de celular

Descrição gerada automaticamente 

Fonte: Biblioteca Wi-Fi Manager

RESULTADOS E DISCUSSÃO

Este trabalho foi dividido em basicamente duas etapas.

Na primeira etapa, foram verificados os modos de funcionamento do sensor de umidade, o display e o motor. Para cada um desses periféricos, foi implementada uma tarefa.

|  |  |
| --- | --- |
| Tarefa | Prioridade |
| sensor\_display\_task | 10 |
| sensor\_umidade\_task | 10 |
| motor\_task | 10 |

Tabela: Prioridade das tarefas iniciais

Para evitar complicações, as tarefas foram criadas com a mesma prioridade inicial.

Foi criada uma Queue do tipo Mailbox

Foi criado um semáforo

Ordem inicial de execução das tarefas

Sensor\_display\_task

Pega o primeiro valor armazenado da queue, transforma para valores reais em unidade de percentual de umidade, aplica a lógica para ligação do motor e envia as informações do sensor para o display.

Devido ao atraso inserido e à prioridade igual às outras tarefas, a próxima tarefa é criada e entra em execução, após o bloqueio da tarefa atual.

Sensor\_umidade\_task

Configura o conversor analógico-digital, lê o valor analógico e envia para a fila xQueueSensor. Também, devido ao atraso inserido e à prioridade igual, a próxima tarefa é criada e entra em execução, após o bloqueio da tarefa atual.

Motor\_task

Espera a liberação do semáforo para acionar o motor. Entra em estado de bloqueio devido ao atraso inserido, o que garante a continuidade do fluxo das tarefas.

Na segunda etapa, foi inserida a conectividade ao projeto. Utilizamos uma biblioteca do Wi-Fi Manager incorporada ao projeto, seguindo as instruções descritas na biblioteca.

Também foi inserida a biblioteca “mqtt\_client”, para o envio de mensagens através do protocolo mqtt.

Em ambos os casos, foram utilizadas variáveis de “event group” para sinalizar alguns eventos importantes, como wi-fi conectado ou mqtt.

A primeira informação necessária na função main foi a inicialização da memória não volátil, necessária para o armazenamento das informações de credenciais da rede wi-fi escolhida.

A rede wi-fi é iniciada através da inicialização do wi-fi manager, o que possibilita a conexão à rede.

A inicialização do wi-fi manager implica em utilização de queue, semáforo mutex, entre outros, e na criação da tarefa “wifi\_manager”, que no caso está configurada com prioridade 5, não oferecendo risco às demais tarefas, até porque a biblioteca já está implementada utilizando-se o FREE-RTOS.

Depois da conexão wi-fi, a rede mqtt é iniciada e configurada.

Foi utilizado o broker da Thingspeak, que possibilita a implementação de dashboards de acompanhamento. Em uma situação de implementação em ambiente de produção, se utilizaria um broker que faria somente papel de broker, e uma aplicação para se conectar com o broker e fazer os dashboards necessários. Nessa implementação não foram utilizadas subscrições, somente publicações no broker, devido a limitações da plataforma e conta.

O projeto envia para o broker mqtt as informações de umidade e estado do motor, e os dashboards apresentam estas informações. Há a possibilidade de armazenar as informações para estudo ou verificar o comportamento do regador ao longo do tempo.

A biblioteca “mqtt\_client” também cria uma tarefa, “mqtt\_task”, cuja prioridade padrão também é igual a 5. Isso possibilita que todas as tarefas tenham chances de serem chamadas.

CONCLUSÃO

REFERÊNCIAS

Free-RTOS

https://www.freertos.org/

The Free-RTOS Reference Manual - API functions and configuration options.

Amazon Web Services

Copyright (C) 2017 Amazon.com, Inc. or its affiliates

Site CloudMQTT

<https://www.cloudmqtt.com/docs/index.html>

Site Embarcados

<https://www.embarcados.com.br/mqtt-protocolos-para-iot/>

Site Free-RTOS.org

<https://www.freertos.org/RTOS-task-states.html>

Biblioteca Wi-Fi Manager

<https://github.com/tonyp7/esp32-wifi-manager>