Projeto de Sistema Embarcado

1. Escopo do Projeto

Apresentação do Projeto

O projeto é sobre manter uma estufa funcionando sem a necessidade constante de intervenção humana, medindo a temperatura e umidade do ambiente e controlando um umidificador, um ventilador, um desumidificador e uma bomba de água para manter o ambiente devidamente climatizado e funcional para o crescimento correto das plantas, emitindo um alerta sonoro caso o sistema não controlando corretamente e permitindo que o usuário configure e adapte o sistema de acordo com a época, plantio e suas condições.

Título do Projeto

Sistema para controle e monitoramento de estufas

Objetivos do Projeto

- Manter a umidade e temperatura dentro de uma faixa estabelecida pelo usuário.
- Mostrar e enviar as informações captadas de umidade e temperatura para os usuários, de forma que consigam adaptar conforme necessário.
- Informar os usuários sobre estado do nível de umidade e temperatura conforme foi configurado, entre baixa, normal e alta.
- Fazer um alarme sonoro, caso o sistema não esteja conseguindo manter o ambiente devidamente climatizado.
- Permitir que o usuário altere a qualquer momento as configurações sem precisar alterar o firmware.

Descrição do Funcionamento

- O microcontrolador virá configurado com valores padrões para a umidade mínima e máxima, temperatura mínima e máxima, velocidade média do ventilador.
- O usuário poderá configurar esses esses valores padrões para o que desejar através de seu computador utilizando o Wifi .
- Se a umidade do ambiente estiver abaixo do configurado, é ativado o umidificador. Se ela estiver acima do configurado é ativado o desumidificador. Se estiver dentro da faixa, nada acontece.
- Se a temperatura do ambiente estiver abaixo do configurado, é desativado o ventilador. Se ela estiver acima do configurado é ativado o ventilador em intensidade máxima. Se estiver dentro da faixa, o ventilador opera em uma velocidade proporcional ao valor da temperatura.

- As informações de umidade e temperatura do ambiente são indicadas no display Oled e enviadas via Wifi.
- Caso se passe um tempo determinado e o sistema continue com condições de temperatura ou ambiente fora da faixa programada, um alarme sonoro será emitido.

Justificativa

A automação de estufas aumenta a eficiência agrícola, economizando tempo, água e energia. Além disso manter a umidade e temperatura no nível ideal é essencial para o desenvolvimento das plantas, pois alta umidade fornece um ambiente propício para o desenvolvimento de patógenos causadores de doenças, já a baixa umidade pode causar estresse na planta e prejudicar o rendimento dela, e a temperatura alta ou baixa pode paralisar o crescimento e até mesmo causar a morte da planta. Portanto o sistema apresentado é capaz de solucionar esses problemas e auxiliar os agricultores, permitindo que dediquem seu tempo para outras demandas e saibam das condições que se encontram sua estufa.

Originalidade

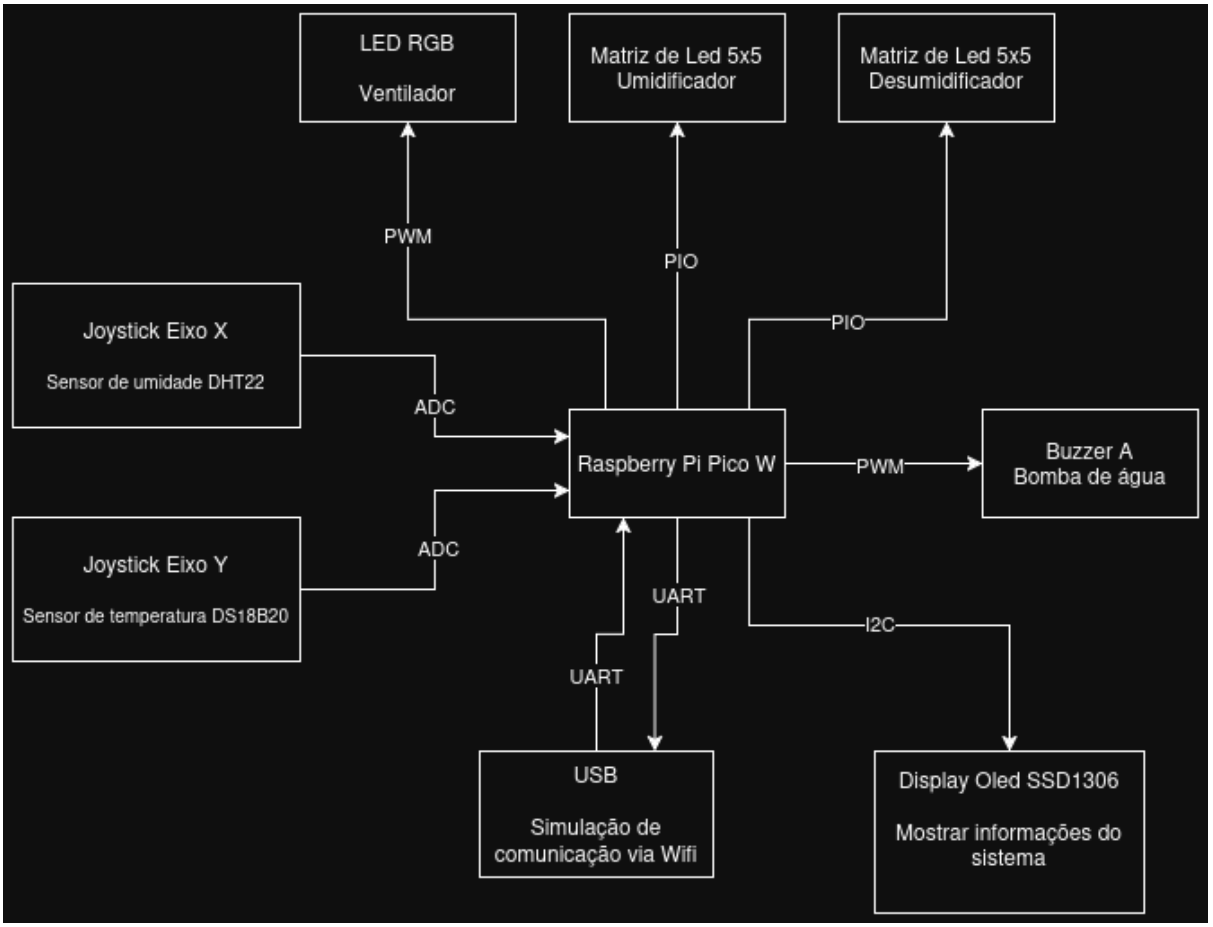
Foram encontrados alguns projetos correlatos, porém nenhum deles integrou umidificadores, desumidificadores, ventiladores, além de permitir uma nova configuração sem precisar de desligar o sistema com uma comunicação à distância e emitir um alerta sonoro caso o sistema não consiga controlar o ambiente.

Projetos encontrados:

- https://github.com/4ngelica/estufa-automatizada/blob/master/README.md
    - Apresenta uma solução eficaz, mas se utiliza de um ventilador para controlar a temperatura e uma bomba de água para a umidade.
- https://revista.uemg.br/index.php/intercursosrevistacientifica/article/view/7335
    - Tem uma proposta diferente: irrigação automatizada, temperatura e iluminação da estufa.
- https://lcv.fee.unicamp.br/images/BTSym18/Papers/016.pdf
    - É o que mais se assemelha, porém utiliza o sensor de umidade no solo para irrigar a plantação e não no ambiente para mantê-lo climatizado como no projeto que desenvolvi.

2. Especificação do Hardware

Diagrama em Bloco



Função de Cada Bloco

- **Raspberry Pi Pico W:** Microcontrolador, para processar os dados dos sensores, controlar os atuadores, gerenciar a comunicação.
- **Joystick Eixo X:** Simular a entrada de dados do sensor de umidade.
- **Joystick Eixo Y:** Simular a entrada de dados do sensor de temperatura.
- **Led RGB:** Simular o ventilador, variando sua intensidade conforme a rotação do ventilador.
- **Matriz de Led 5x5:** Mostrar um símbolo, representando a ativação do umidificador ou do desumidificador.
- **Buzzer A:** Disparar um alarme sonoro caso o ambiente esteja fora da faixa esperada por determinado tempo.
- **USB:** Simular a comunicação via wifi, enviando informações sobre a umidade e temperatura, um menu para configuração, e recebendo os dados e comandos para configurar o sistema.
- **Display Oled SSD1306:** Mostrar as informações do sistema: Temperatura, umidade e seus respectivos níveis.

Configuração de Cada Bloco

- **Raspberry Pi Pico W:** Clock de 125 Mhz e protocolos I2C, PWM, PIO habilitados

- **Joystick Eixo X:** ADC para simular umidade (0-100%)
- **Joystick Eixo Y:** ADC para simular temperatura (-10°C a 85°C)
- **Led RGB:** Controlado por PWM
- **Matriz de Led 5x5:** Controlado via PIO
- **Buzzer A:** Controlado por PWM
- **USB:** Interface UART
- **Display Oled SSD1306:** 128X64 pixels. Interface I2C.

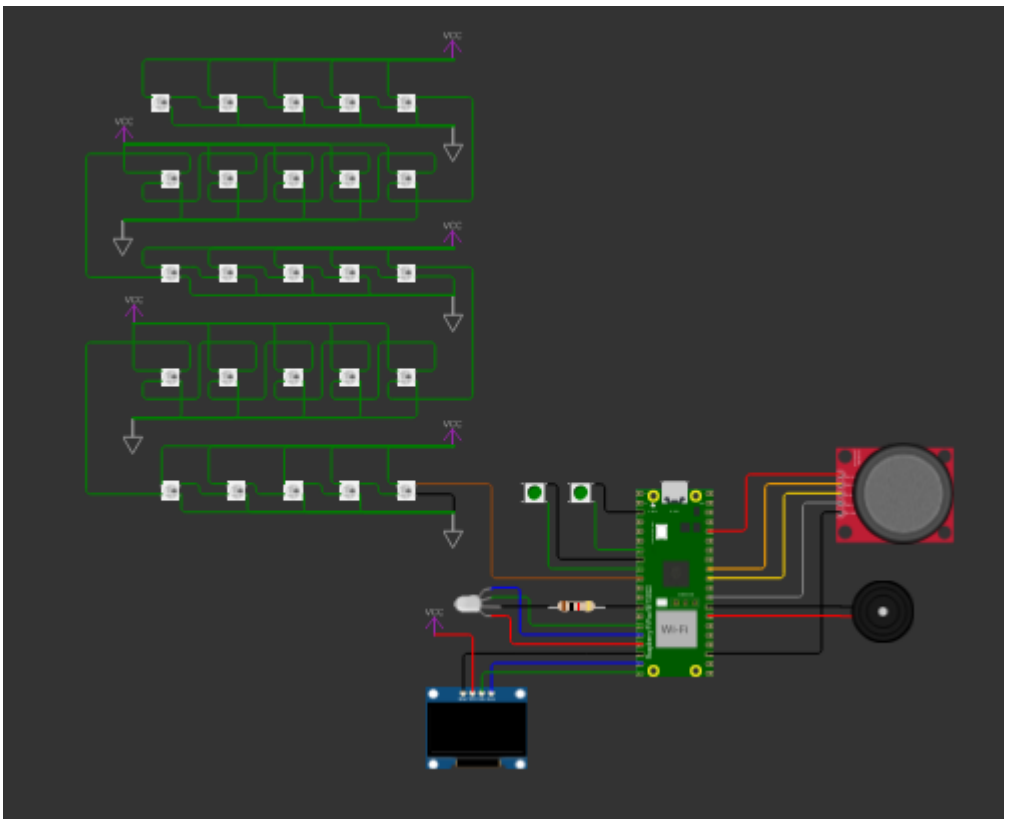
Comandos e Registros Utilizados

- **ADC (Sensores):** adc_init(), adc_gpio_init().
- **PWM (Led):** gpio_set_function(), pwm_gpio_to_slice_num(), pwm_set_wrap(), pwm_set_gpio_level(), pwm_set_enabled().
- **PWM (Buzzer):** pwm_set_wrap(), pwm_gpio_to_slice_num(), pwm_set_gpio_level(), pwm_set_clkdiv(), pwm_set_enabled().
- **PIO (Matriz de Led):** pio_add_program(), pio_claim_unused_sm(), animacao_matriz_program_init().
- **I2C (Display OLED):** i2c_init(), gpio_set_function(), gpio_pull_up(), ssd1306_init(), ssd1306_config(), ssd1306_fill() ,ssd1306_send_data().
- **USB UART:** stdio_init_all().

Descrição da Pinagem Usada

- **Joystick Eixo X:** GP27 - Leitura analógica de umidade simulada
- **Joystick Eixo Y:** GP26 - Leitura analógica de temperatura simulada
- **Led PWM:**
    - GP13 - Acender a cor vermelha,
    - GP12 - Acender a cor azul,
    - GP11 - Acender a cor verde.
- **Matriz de Led 5x5:** GP7 - Controlar os leds endereçáveis via PIO
- **Buzzer A:** GP21 - Gerar sonoridade através de PWM
- **Display Oled SSD1306:**
    - GP14 - linha de dados serial SDA
    - GP15 - linha de clock serial SCL

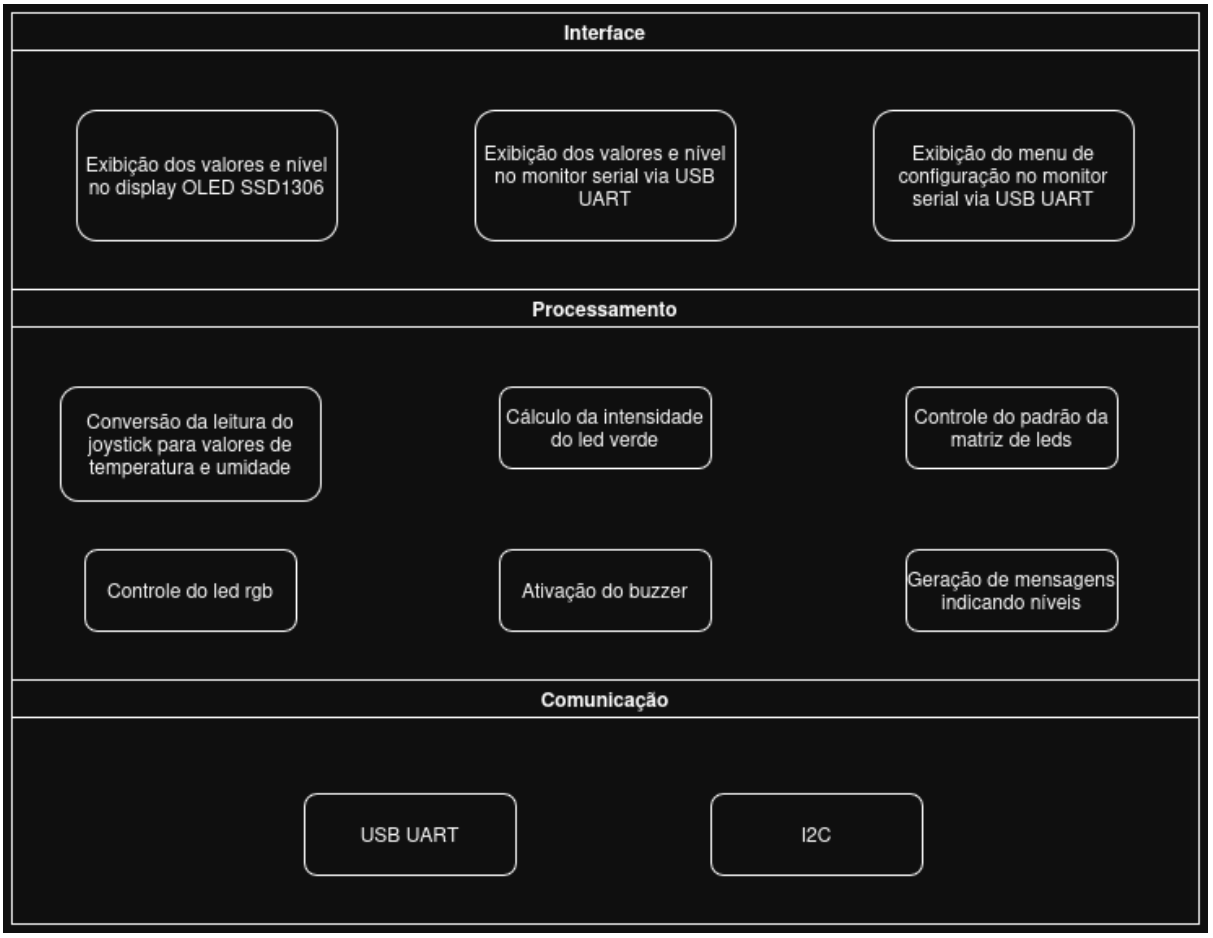
Circuito Completo do Hardware



controle_estufa - Wokwi ESP32, STM32, Arduino Simulator. Disponível em: <https://wokwi.com/projects/423834179903621121>. Acesso em: 26 fev. 2025.

3. Especificação do Firmware

Blocos Funcionais



Descrição das Funcionalidades

- **Interface:**
  - **Exibição dos valores e nível no display OLED SSD1306:**

    Exibe no display oled os valores lidos pelos sensores referentes a temperatura e umidade e seus respectivos níveis: alta, normal e baixa.

  - **Exibição dos valores e nível no monitor serial via USB UART:**

    Exibe no monitor serial os valores lidos pelos sensores referentes a temperatura e umidade e seus respectivos níveis: alta, normal e baixa.

  - **Exibição do menu de configuração no monitor serial via USB UART:**

    Exibe no monitor serial o menu de configuração para que o usuário seja capaz de escolher qual configuração deseja alterar.

- **Processamento:**
  - **Conversão da leitura do joystick para valores de temperatura e umidade:**

Responsável por converter os dados lidos pelo joystick do eixo x para a faixa de valores entre 0 e 100, simulando a umidade e os dados lidos no eixo y para a faixa entre -10 e 85, simulando a temperatura.

- **Cálculo da intensidade do led verde:**

  Responsável por fazer o led rgb variar entre a intensidade mínima e máxima enquanto a temperatura está dentro da faixa configurada, atingindo a intensidade máxima quando a temperatura está igual a máxima definida e mínima quando a temperatura está igual a mínima definida.

- **Controle do padrão da matriz de leds:**

  Responsável por fazer o padrão que irá representar o ativamento do umidificador ou do desumidificador, a depender do nível de umidade. Mantendo desligado quando está dentro da faixa configurada.

- **Controle do led rgb:**

  Responsável por fazer o led desligar quando a temperatura estiver abaixo da mínima, variar de acordo com o cálculo da intensidade caso esteja entre a mínima e máxima, e acender a luz vermelha caso esteja acima da máxima.

- **Ativação do buzzer:**

  É ativado caso o sistema não esteja conseguindo manter a temperatura e umidade controladas na faixa configurada por mais de 10 segundos, com uma tolerância de até 6 segundos que é o tempo de verificação dos sensores.

- **Geração de mensagens indicando níveis:**

  As mensagens para exibir no display e monitor serial são geradas de acordo com a temperatura e umidade.

- **Comunicação:**
  - **USB UART:**

    Utilizado para enviar os valores e níveis de temperatura e umidade, exibir o menu de configuração e ler os dados para reconfigurar a forma como o sistema funciona.

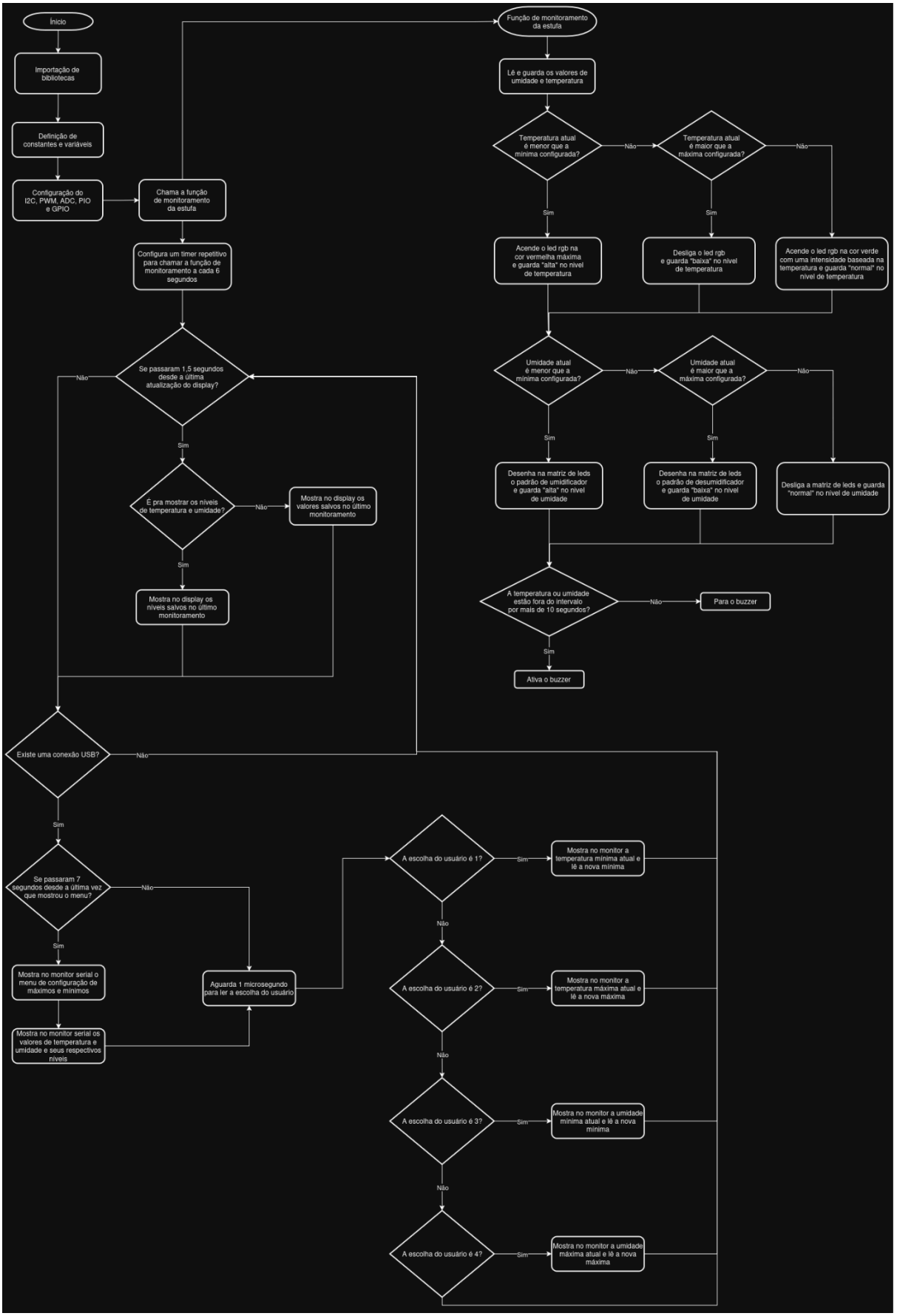
  - **I2C:**

    Utilizado para controlar o display OLED SSD1306, permitindo que os valores e níveis de mensagem sejam exibidos corretamente.

Definição das Variáveis

- **last_time_display:** Para armazenar o tempo da última vez que atualizou o display.
- **last_time_usb:** Para armazenar o tempo que a última mensagem foi enviada pela UART USB.
- **last_time_temp_normal:** Para armazenar o tempo da última vez que a temperatura estava dentro da faixa programada.
- **last_time_umid_normal:** Para armazenar o tempo da última vez que a umidade estava dentro da faixa programada.
- **current_time_display:** Para armazenar o tempo da atual e poder comparar com a última vez que atualizou o display.
- **current_time_usb:** Para armazenar o tempo da atual e poder comparar com a última vez que a última mensagem foi enviada pela UART USB.
- **current_time_normal:** Para armazenar o tempo da atual e poder comparar com a última vez que a temperatura ou umidade estavam dentro da faixa programada.
- **temp_min:** Para definir qual a temperatura mínima aceitável pelo ambiente.
- **temp_max:** Para definir qual a temperatura máxima aceitável pelo ambiente.
- **temp_atual:** Para guardar qual a temperatura lida atualmente pelos sensores.
- **umid_min:** Para definir qual a umidade mínima aceitável pelo ambiente.
- **umid_max:** Para definir qual a umidade máxima aceitável pelo ambiente.
- **umid_atual:** Para guardar qual a umidade lida atualmente pelos sensores.
- **choice:** Para capturar a escolha do usuário no menu de configuração
- **msg:** Struct para armazenar os valores de umidade e temperatura formatados em uma string e seus respectivos níveis.
- **mostrar_nivel:** Alternar entre mostrar ou não o nível no display.
- **padrao_led:** Matriz para armazenar o desenho que será disposto na matriz de leds

Fluxograma


Início
Importação de bibliotecas
Definição de constantes e variáveis
Configuração do I2C, PWM, ADC, PIO e GPIO
Chama a função de monitoramento da estufa
Configura um timer repetitivo para chamar a função de monitoramento a cada 6 segundos
Função de monitoramento da estufa
Lê e guarda os valores de umidade e temperatura
Temperatura atual é menor que a mínima configurada?
Não
Temperatura atual é maior que a máxima configurada?
Não
Sim
Sim
Acende o led rgb na cor vermelha máxima e guarda "alta" no nível de temperatura
Desliga o led rgb e guarda "baixa" no nível de temperatura
Acende o led rgb na cor verde com uma intensidade baseada na temperatura e guarda "normal" no nível de temperatura
Umidade atual é menor que a mínima configurada?
Não
Umidade atual é maior que a máxima configurada?
Não
Sim
Sim
Desenha na matriz de leds o padrão de umidificador e guarda "alta" no nível de umidade
Desenha na matriz de leds o padrão de desumidificador e guarda "baixa" no nível de umidade
Desliga a matriz de leds e guarda "normal" no nível de umidade
A temperatura ou umidade estão fora do intervalo por mais de 10 segundos?
Não
Para o buzzer
Sim
Ativa o buzzer
Se passaram 1,5 segundos desde a última atualização do display?
Não
Sim
É pra mostrar os níveis de temperatura e umidade?
Não
Mostra no display os valores salvos no último monitoramento
Sim
Mostra no display os níveis salvos no último monitoramento
Existe uma conexão USB?
Não
Sim
Se passaram 7 segundos desde a última vez que mostrou o menu?
Não
Sim
Mostra no monitor serial o menu de configuração de máximos e mínimos
Mostra no monitor serial os valores de temperatura e umidade e seus respectivos níveis
Aguarda 1 microsegundo para ler a escolha do usuário
A escolha do usuário é 1?
Sim
Mostra no monitor a temperatura mínima atual e lê a nova mínima
Não
A escolha do usuário é 2?
Sim
Mostra no monitor a temperatura máxima atual e lê a nova máxima
Não
A escolha do usuário é 3?
Sim
Mostra no monitor a umidade mínima atual e lê a nova mínima
Não
A escolha do usuário é 4?
Sim
Mostra no monitor a umidade máxima atual e lê a nova máxima

Inicialização

O sistema inicializa configurando a comunicação I2C para o display OLED, em seguida configura o PIO para a comunicação com a matriz de leds. Após isso, configura os pinos do led rgb com PWM e configura os eixos do joysticks como ADC.

Configurações dos Registros

- **ADC (Sensores):**
    - **adc_init():** Para inicializar o conversor analógico digital.
    - **adc_gpio_init():** Para inicializar determinado pino GPIO como uma entrada analógica, no nosso caso o 26 e 27.
- **PWM (Led):**
    - **gpio_set_function():** Define a função do pino para que opere no modo PWM.
    - **pwm_gpio_to_slice_num():** Retorna o número do "slice" (grupo PWM) associado ao pino, permitindo a configuração do PWM naquele grupo.
    - **pwm_set_wrap():** Define o valor máximo do contador (wrap) para o PWM, determinando o período do sinal PWM.
    - **pwm_set_gpio_level():** Configura o duty cycle do PWM, ou seja, o nível médio do sinal, que controla a intensidade luminosa dos LEDs.
    - **pwm_set_enabled():** Habilita ou desabilita o funcionamento do PWM para o slice configurado.
- **PWM (Buzzer):**
    - **pio_set_function():** Define a função do pino para que opere no modo PWM.
    - **pwm_gpio_to_slice_num():** Retorna o número do "slice" (grupo PWM) associado ao pino, permitindo a configuração do PWM naquele grupo.
    - **pwm_set_wrap():** Define o valor de wrap para o PWM do buzzer, que determina o período do sinal e, em conjunto com o divisor, a frequência (tom) do som.
    - **pwm_set_gpio_level():** Ajusta o duty cycle do sinal PWM, que influencia o volume (intensidade sonora) do buzzer.
    - **pwm_set_clkdiv():** Configura o divisor do clock para o slice PWM, possibilitando ajustar a frequência do som emitido pelo buzzer.
    - **pwm_set_enabled():** Habilita ou desabilita o funcionamento do PWM para o slice configurado.
- **PIO (Matriz de Led):**
    - **pio_add_program():** Carrega o programa PIO (no nosso caso, o programa para controle da LED Matrix) na memória do PIO.
    - **pio_claim_unused_sm():** Reserva uma "State Machine" (SM) do PIO que ainda não está em uso, para ser utilizada no controle dos LEDs.
    - **animacao_matriz_program_init():** Inicializa o programa PIO com os parâmetros necessários (como offset e pino de saída) para controlar a matriz de LEDs de acordo com os padrões desejados.
- **I2C (Display OLED):**
    - **i2c_init():** Inicializa o barramento I2C do RP2040, definindo a velocidade de comunicação

    - **gpio_set_function():** Configura os pinos designados para a comunicação I2C (SDA e SCL) para a função I2C.
    - **gpio_pull_up():** Ativa os resistores de pull-up nos pinos SDA e SCL.
    - **ssd1306_init():** Inicializa o display OLED (SSD1306), configurando parâmetros como tamanho, endereço e modo de operação.
    - **ssd1306_config():** Envia os comandos de configuração iniciais para o display OLED, preparando-o para receber dados.
    - **ssd1306_fill():** Limpa o display, para começar inicialmente apagado.
    - **ssd1306_send_data():** Envia os dados para o display, atualizando a imagem exibida.
- **USB UART:**
    - **stdio_init_all():** Inicializa todas as interfaces padrão, incluindo a comunicação USB (UART) do RP2040, permitindo a depuração e a interação com o usuário via terminal.

Estrutura e Formato dos Dados

- **msg_t:**
    - Um struct que armazena 4 strings, **nivel_temp** e **nivel_umid** com um tamanho de 20 caracteres e **string_temp_atual** e **string_umid_atual** com um tamanho de 6 caracteres.
    - **nivel_temp e nivel_umid** são utilizados para armazenar os níveis entre alto, normal e baixo.
    - **string_temp_atual** e **string_umid_atual** são utilizados para armazenar o valor lido dos eixos dos joysticks, depois que eles são corretamente formatados para o uma string.
    - Esse tipo de dado é necessário para a função de callback poder manipular e guardar esses valores, para que quando chegue no display ou na comunicação UART, seja possível exibir corretamente.
- **last_time_display:** uint32_t
- **last_time_usb:** uint32_t
- **last_time_temp_normal:** uint32_t
- **last_time_umid_normal:** uint32_t
- **current_time_display:** uint32_t
- **current_time_usb:** uint32_t
- **current_time_normal:** uint32_t
- **temp_min:** int
- **temp_max:** int
- **temp_atual:** volatile uint16_t - Valores inteiros representando temperatura (-10°C a 85°C) e umidade (0-100%).
- **umid_min:** uint
- **umid_max:** uint
- **umid_atual:** volatile uint16_t - Valores inteiros representando temperatura (-10°C a 85°C) e umidade (0-100%).
- **choice:** bool

Protocolo de Comunicação

Foi utilizado o protocolo UART, para se comunicar com a placa através do USB. Para enviar os comandos basta se conectar através do monitor serial com um baud raute de 115200 e com o final de linha LF.

Formato do Pacote de Dados

Texto não criptografado. Valores entre 1 a 4 para escolha do menu seguidos do final de linha LF. Para a configuração tem que ser um inteiro para a temperatura e um inteiro sem sinal para a umidade, ambos seguidos de um final de linha.

4. Execução do Projeto

Metodologia

- **Pesquisa e levantamento**

  Foi realizada uma pesquisa sobre funcionamento de estufas e como automatizá-las, sensores utilizados, ventiladores, umidificadores, desumidificadores. Também foi buscado projetos correlatos e foi feito um estudo de como eles operacionalizam, identificando pontos fortes e lacunas para poder implementar minha solução.

- **Escolha do Hardware:**

  Os sensores de umidade e temperatura foram escolhidos graças a sua precisão e durabilidade, devido ao fato de a estufa ser um ambiente muito úmido e poder afetar esses equipamentos eletrônicos. Os outros atuadores podem ser escolhidos de acordo com a demanda e tamanho da estufa, por isso foram colocados de forma genérica.

- **Desenvolvimento e integração:**

  O primeiro passo foi configurar a IDE Visual Studio Code, instalando a extensão oficial da raspberry pi pico e a extensão do wokwi. Logo após, a extensão do raspberry foi utilizada para criar a base do projeto. Em seguida foi realizada a configuração do CMakeLists para incluir todas as bibliotecas necessárias para o projeto e habilitando a comunicação UART USB. Os arquivos referentes a simulação com o wokwi foram criados localmente, primeiramente criando o diagram.json no site oficial do wokwi e depois transferindo localmente.

  As funcionalidades do software foram estruturadas de forma que se adquira periodicamente os dados dos sensores, liberando mais tempo para o processador atualizar o display e se comunicar com a interface UART.

Os tempos de verificação e ativação do buzzer são para nível de simulação, a depender do local e usuário isso deve ser alterado.

Testes de Validação

Para os joysticks foram realizados testes para ver quais números máximos e mínimos retornavam em ambos os eixos e após validado, esses dados foram capturados e convertidos para as faixas de valores de umidade e temperatura esperados, faixas essas que também foram testadas se atingiam os valores correspondentes.

Para a matriz de leds foi verificado se a cor e os leds estavam acendendo da forma correta, como foram programados.

Para o display foi modificado diversas posições do joystick e analisado se estava com o nível e valor correto.

Para o buzzer foi colocado valores fora da faixa com o joystick e verificado se o tempo até emitir o som, era o mesmo do de verificação e do tempo programado, levando em conta que o sensor é lido a cada 6 segundos.

Discussão dos Resultados

Os testes realizados demonstraram que o sistema cumpre as funções de monitoramento e controle de uma estufa de forma eficaz:

- **Desempenho:**
  O firmware atualiza os valores dos sensores de forma periódica, permitindo que a comunicação via UART USB funcione perfeitamente, lendo quando necessário e mostrando tanto no monitor serial quanto no display as informações capturadas. Além de que o sistema reage rapidamente às alterações do ambiente, acionando os atuadores e o alarme sonoro quando necessário.
- **Confiabilidade:**
  A integração entre os diversos módulos (ADC, PWM, I2C, PIO e USB) apresentou resultados consistentes. O uso de um timer repetitivo e a verificação de tempo para as condições fora do intervalo garantiram que o alarme (buzzer) só fosse ativado em situações críticas.
- **Interatividade:**
  A interface via USB permite ao usuário ajustar os parâmetros do sistema sem a necessidade de reprogramar o firmware, agregando flexibilidade e praticidade.

5. Referências

LOPES, J. **Plantfort - Transformando o Cultivo de Mudas de Cacau com Tecnologia de Estufas**. Disponível em: https://plantfort.ind.br/blog/89-transformando-o-cultivo-de-mudas-de-cacau-com-tecnologia-de-estufas. Acesso em: 26 fev. 2025.

Controle de Microclima e Irrigação em Estufas - elysios. Disponível em: https://elysios.com.br/controle-de-microclima-de-irrigacao-em-estufas/. Acesso em: 26 fev. 2025.

4NGELICA. **estufa-automatizada/README.md at master · 4ngelica/estufa-automatizada**. Disponível em: https://github.com/4ngelica/estufa-automatizada/blob/master/README.md. Acesso em: 27 fev. 2025.

LOPES SILVA, KATIA; MOREIRA FRANCO, Matheus; OLIVEIRA FERRARI, HÉLIO; HEMERLY GAZZANI, MAURO. DESENVOLVIMENTO DE CONTROLES AUTOMATIZADOS EM ESTUFAS AGRÍCOLAS. **Intercursos Revista Científica**, *[S. l.]*, v. 22, n. 1, p. 86–107, 2023. Disponível em: https://revista.uemg.br/index.php/intercursosrevistacientifica/article/view/7335. Acesso em: 26 fev. 2025.

DE, V. et al. **Sistema Embarcado para Aplicação em Monitoramento e Controle de Temperatura e Umidade**. [s.l: s.n.]. Disponível em: https://lcv.fee.unicamp.br/images/BTSym18/Papers/016.pdf. Acesso em: 26 fev. 2025.

Links do vídeo e repositório

Link do vídeo: https://youtu.be/slr7jHzEJRI

Link do repositório: https://github.com/yuriccosta/Controle_estufa