Documentação do projeto realizado no curso de capacitação Embarcatech

Aluno : Ítalo Alves Tenório de Almeida

Sistema Dual-Module para resfriamento e monitoramento de intensidade luminosa em módulos fotovoltaicos.

Link do repositório :
https://github.com/Romulus09/Projeto_Embarca_Sistema_Dual_Module.git
Link do vídeo :
https://youtu.be/xTfbLNeej8Y

1.1- Introdução

O projeto proposto neste documento se trata de um sistema IOT para refrigeração e monitoramento de temperatura em paineis fotovoltaicos, além de monitoramento da luminosidade. O sistema é controlado centralmente pela placa Raspberry Pi Pico W, e consequentemente possui como unidade microcontroladora o RP2040, fazendo uso de periféricos pertinentes aos objetivos do projeto como o ADC, PWM,Timers e PIO, todos embarcados no próprio RP2040. O sistema consiste basicamente num conjunto de sensores, responsáveis por obter do ambiente dados que possibilitem o monitoramento da temperatura de um módulo fotovoltaico e o controle para que a mesma permaneça em um intervalo adequado predeterminado, além do sensoriamento da luminosidade e exibição desses dados para o usuário no sistema supervisor. O controle de temperatura será realizado pelo conjunto de atuadores responsáveis pelo ato da refrigeração em si, mais especificamente um sistema de sprinklers de água e um ventilador controlado por um motor DC, que serão ativados conforme o necessário, baseado nas leituras de temperatura recebidas pelo microcontrolador através dos sensores. É embutido no sistema também um sensor de proximidade infravermelho, que terá como função detectar quando a quantidade de água no reservatório estiver baixa, e consequentemente desativar a rotina dos sprinklers. Além disso, o projeto conta com um sistema supervisor através do display OLED 128x64, que possibilitará ao usuário acompanhar as leituras de temperatura e luminosidade, bem como possíveis alertas que poderão surgir caso o painel aqueça fora do range adequado, e a ação dos atuadores que serão utilizados para lidar com essas situações. Os sensores presentes no sistema, bem como alguns atuadores, serão simulados por componentes da placa de desenvolvimento BitDogLab.

1.2-Descrição do problema

Os painéis solares desempenham um papel crucial na nossa mudança para fontes de energia sustentáveis, mas a sua eficácia pode ser prejudicada pelas flutuações de temperatura (Srinivasarao et al, 2024). Aumentar a temperatura da superfície do módulo fotovoltaico em apenas 1°C pode diminuir a eficiência em aproximadamente 0,5%. Portanto, o aumento na temperatura faz com que nem toda a energia solar seja absorvida pelas células fotovoltaicas para se converter em energia elétrica. A energia solar restante é convertida em calor. Além disso, esse calor desperdiçado reduzirá a eficiência geral de conversão (Azmi et al, 2023). Para lidar com esse problema, diversas técnicas de

refrigeração e controle de temperatura são utilizadas, como sistemas passivos refrigerados a ar, sistemas de refrigeração a água em circuito fechado, sistema refrigerado a ar e sistema de borrifamento de água (Kamarudin et al, 2021) . Pesquisas (semelhantes ao projeto proposto) nessa área já propuseram soluções, inclusive com o uso de sistemas embarcados. Srinivasarao (2024) propôs uma solução inovadora utilizando a placa Arduino, sensores de temperatura e um ventilador ativado por motor DC. Já Azmi (2023) apresentou um sistema de refrigeração híbrido, também controlado por Arduino, consistindo de um ventilador acionado por motor DC e também de uma bomba d'água. O projeto proposto neste documento utiliza a placa Raspberry Pi Pico W, usando além de um motor DC, um sistema de sprinklers (irrigadores), e incrementa ao design um sistema supervisor através do display OLED presente na placa BitDogLab, além de um mecanismo para a desativação do sistema de irrigação quando o nível da água contida no reservatório estiver baixo. Também é adicionado um módulo para o sensoriamento e monitoramento da luz incidida sobre o módulo fotovoltaico através de um LDR.

2.1-Diagramas e especificações de hardware

A figura 1 ilustra o diagrama de blocos do hardware do sistema.

Sensor Infravermelho (pino 5) : O sensoriamento infravermelho funciona da seguinte forma : dentro do reservatório da água que será direcionada aos sprinklers, existe uma pequena boia, presa ao fundo do reservatório por uma corda ou método similar. Próximo ao fundo do reservatório, está um sensor infravermelho que detectará quando a boia passar na sua frente, sinalizando que a boia desceu, e que consequentemente o nível da água no reservatório está baixo. O sensor de proximidade infravermelho, será simulado por um push button. Caso o botão seja pressionado, subentende-se que o sensor infravermelho detectou a presença da boia, e que consequentemente o nível da água está baixo.

Sensor de temperatura (pino 27): O sensor de temperatura, simulado pelo eixo Y do Joystick da placa bitdoglab será responsável por medir a temperatura dos módulos fotovoltaicos, enviando esse dado ao microcontrolador para que seja então determinado se o sistema de resfriamento do ventilador deverá ser acionado. Como já mencionado, as pesquisas realizadas informaram que a perda de eficiência dos módulos é de aproximadamente 0.5% para cada grau C de temperatura mais alto do que o limite máximo do range adequado para o funcionamento das placas. O consenso encontrado para o limite máximo do range adequado de funcionamento foi 35 C. Sendo assim, foi determinado no projeto que o sistema de refrigeração será acionado quando a leitura de temperatura superar os 40C, mantendo a perda de eficiência em no máximo 5%.

Motor DC (pino 12) : esse atuador será responsável por controlar a ação do ventilador, sendo controlado por modulação PWM, e simulado no projeto por um LED azul, que acenderá gradativamente até o brilho máximo, dependendo da leitura da temperatura. O brilho representa a velocidade com que o ventilador gira.

Resistor LDR (pino 26) : esse sensor será simulado pelo eixo X do joystick e representa a incidência de luz sobre as placas fotovoltaicas. Quanto menor a incidência de luz, maior a

resistência. O valor de resistência irá variar entre 3 e 1012 ohms, de acordo com o valor do ADC.

Buzzer (pino 21): esse atuador será responsável por emitir um alerta sonoro quando a resistência do LDR ultrapassar um valor de 900 ohms, sinalizando assim um nível de luminosidade baixa incidindo sobre as placas solares. É controlado por PWM.

Matriz RGB 5x5 : esse atuador servirá como um segundo feedback visual para o acionamento do ventilador do sistema, exibindo uma animação enquanto o mesmo estiver ativado.

Sprinklers (pino 13): esse atuador, simulado por uma modulação PWM do LED vermelho da placa bitdoglab, tem a função híbrida de auxiliar na limpeza periódica das placas solares, bem como no resfriamento das placas. Os irrigadores são temporizados e ativam periodicamente durante a duração do programa, a não ser que sua ativação seja inibida pelo sensor infravermelho, no caso do nível da água no reservatório estar baixo.

Display OLED (SCL pino 14, SDL pino 15) : esse atuador servirá como o sistema supervisor central do sistema, exibindo as leituras de temperatura das placas e da resistência do LDR, bem como alertas na ativação/desativação dos sprinklers de água.

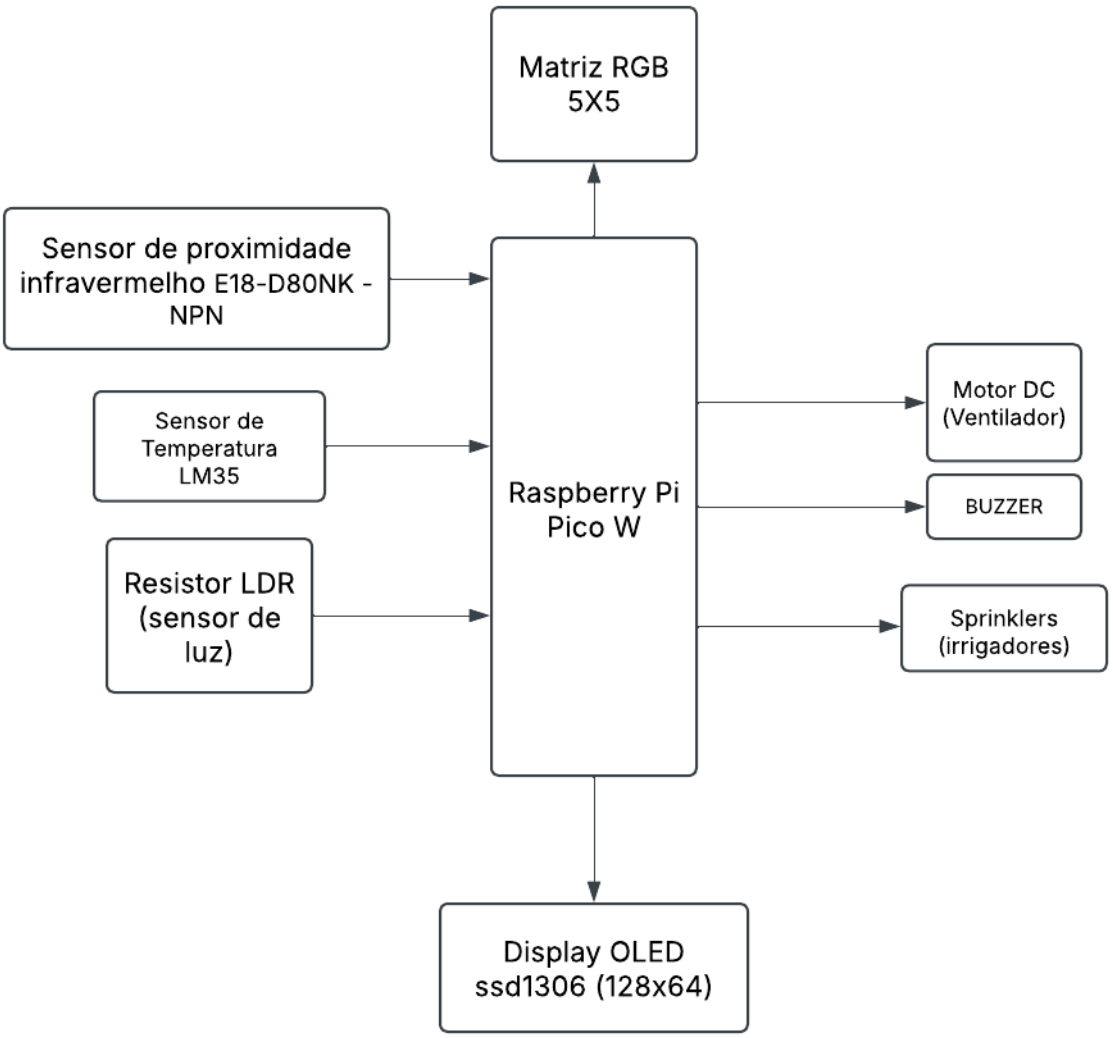

Figura 1: Diagrama de Blocos do Hardware

A figura 2 ilustra o circuito do sistema,com seus componentes simuladores, através do diagrama feito no ambiente de simulação virtual Wokwi :

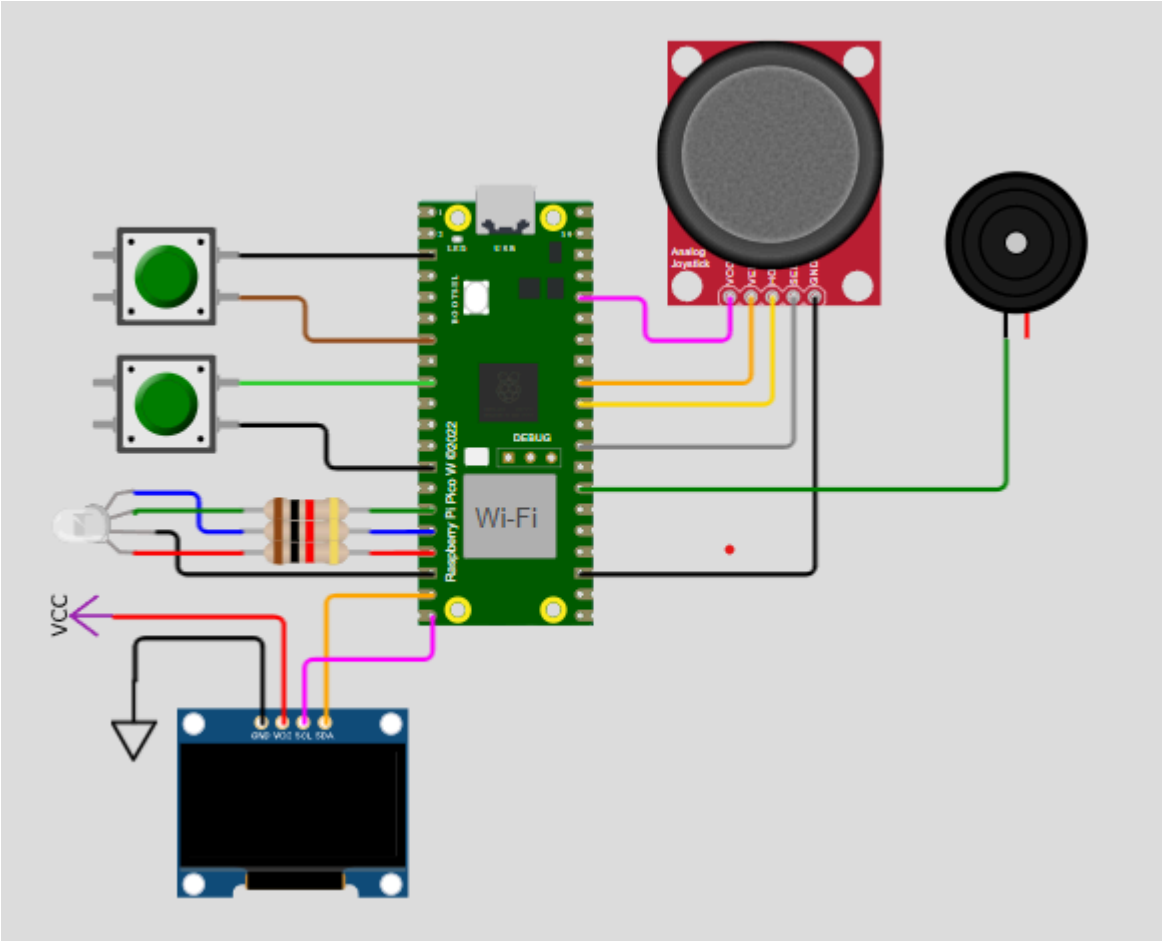

Figura 2 - Circuito do Projeto

2.2- Diagramas e especificações de software

2.2.1 - Fluxograma

A figura 3 abaixo ilustra o fluxograma geral do programa :

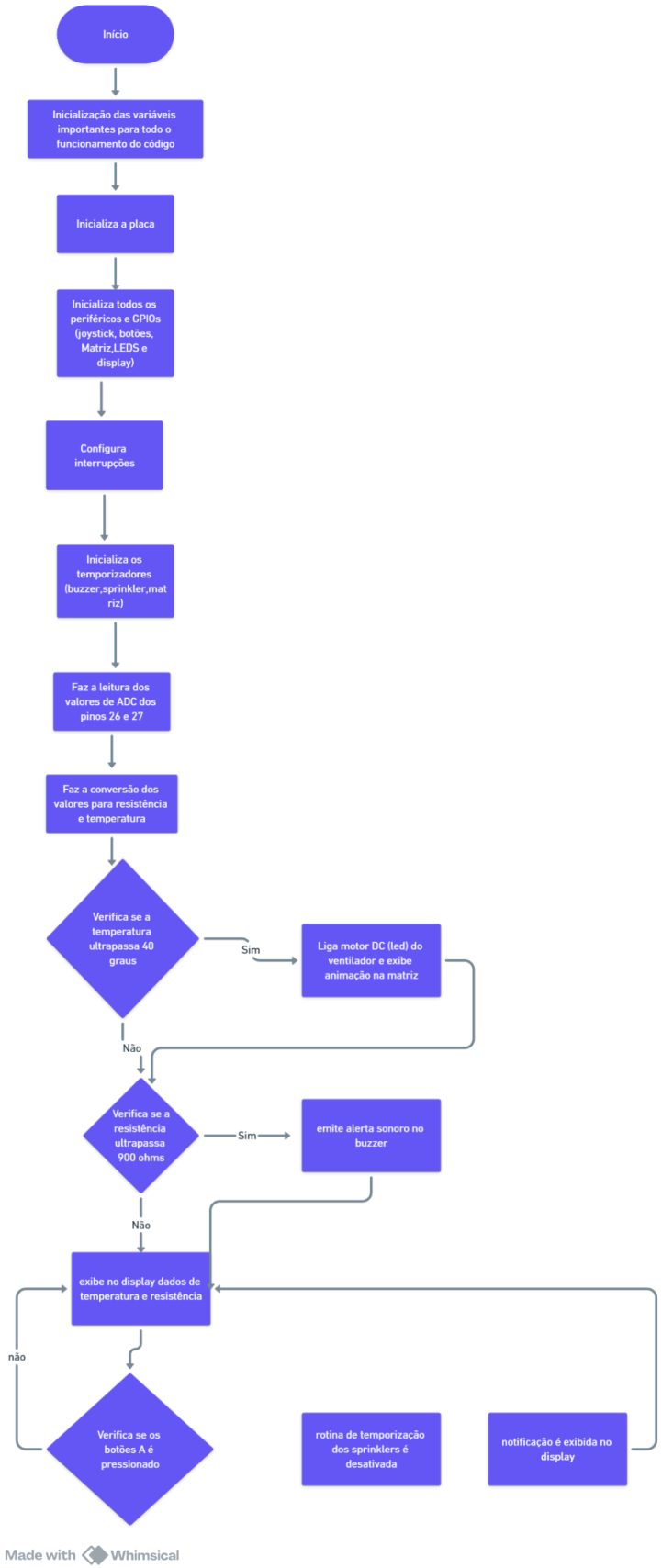

Figura 3 : Fluxograma do firmware

2.2.2- Diagrama de Camadas

A figura 4 ilustra o modelo em camadas do firmware do sistema.

Camada de Sensoriamento : responsável pela obtenção dos dados lidos pelos diversos sensores do sistema.

Camada de Processamento : responsável pela pela conversão dos dados ADC lidos pelos sensores para as unidades de medidas pertinentes, dependendo do sensor. Além disso, é nessa camada que ocorre todas as configurações de hardware necessárias para o funcionamento do programa, tais como inicializações de GPIO, configurações de saídas PWM, inicialização e configuração do protocolo I2C, utilizado para a comunicação com o display OLED.É também nessa camada onde ocorre o controle das saídas PWM e a configuração dos Timers do sistema, bem como a definição de suas respectivas funções de callback. A configuração da Matriz RGB 5x5 é também feita aqui.

Camada de Atuadores : responsável pela ação dos atuadores do sistema. Exibição de mensagens no display, exibição de padrões na matriz 5x5, modulação das saídas PWM dos sprinklers, buzzer e do motor DC.

Figura 4 : Diagrama em camadas de software

Abaixo estão definidas as principais funções utilizadas no programa :

void set_one_led(uint8_t r, uint8_t g, uint8_t b, bool matriz[]) : essa função exibe na matriz de LEDS 5x5 a configuração especificada pela matriz[] passada como parâmetro, com valores de RGB também passados via parâmetro.

void pwm_setup_fan(): Essa função configura as funcionalidades PWM do led (pino 12) responsável por simular a ação do motor DC que controlará o ventilador. O pino é

configurado como saída pwm e seu slice correspondente é definido com o divisor de clock e o período de wrap especificado.

void pwm_setup_sprinkler():análoga à função anterior, esta tem como objetivo configurar as funcionalidades PWM do led (pino 13) responsável por simular a ação dos irrigadores (sprinklers) de água.O pino é configurado como saída pwm e seu slice correspondente é definido com o divisor de clock e o período de wrap especificado.

void pwm_fan_control(uint16_t analog input) : Liga o ventilador (led) caso a leitura do sensor de temperatura (parâmetro) supere os 40 graus celsius.

void pwm_sprinkler_ctrl() : controla a ação dos sprinklers, simulados por um led que tem seu brilho aumentado gradativamente até o máximo, e em seguida diminui o brilho até apagar, repetidamente.

bool sprinkler_callback(struct repeating_timer *t) : função de callback do temporizador que controla a ação dos sprinklers. De acordo com a temporização especificada (5 minutos desligado e 1 minuto desligado no projeto real, porém 3 segundos desligados e 6 segundos ligados no código, para fins de demonstração), essa função será chamada por um timer periodicamente, chamando a função pwm_sprinkler_ctrl() cada vez para ajustar o brilho do led via pwm e simular a ação dos sprinklers.

bool fan_animation_callback(struct repeating_timer *t) : função de callback do temporizador t que controla a ação da matriz de leds 5x5. A matriz exibirá uma animação enquanto os ventiladores estiverem ligados (temperatura > 40), que servirá como um outro feedback visual da ativação do ventilador.

void setup_gpio() : função que inicializa e configura todas as GPIO's necessárias para a execução do programa.

void setup_irq() : função que configura todas as interrupções utilizadas no programa.

void setup_display() : função que configura e inicializa todas as funcionalidades I2C do display OLED ssd1306, preparando o componente para comunicação.

Abaixo estão especificadas as principais variáveis utilizadas para a execução do firmware:

Variáveis de Controle do Sistema

- sprinkler_active (**bool**) → Indica se os sprinklers estão ativados.
- fan_active (**bool**) → Indica se os ventiladores estão ativados.
- sprinkler_control (**int**) → Controla o tempo de funcionamento do sprinkler.
- fan_animation_control (**int**) → Controla os frames da animação do ventilador na matriz de LEDs.
- sprinkler_level (**int**) → Controla o brilho do LED que simula os sprinklers.
- up (**bool**) → Direção do ajuste de brilho do LED do sprinkler (aumentando ou diminuindo).

Variáveis dos Sensores (Simulados)

- JOYSTICK_X_PIN (**GPIO 26**) → Entrada analógica para eixo X do joystick.
- JOYSTICK_Y_PIN (**GPIO 27**) → Entrada analógica para eixo Y do joystick.
- JOYSTICK_PB (**GPIO 22**) → Botão do joystick.
- adc_temp_read (**uint16_t**) → Valor lido do ADC simulando a temperatura.
- temperature_celsius (**int**) → Temperatura convertida a partir do ADC.

Variáveis dos Atuadores (Simulados)

- LED_BLUE (**GPIO 12**) → LED azul (simula o ventilador).
- LED_PISCA (**GPIO 13**) → LED vermelho (simula o sprinkler).

Configuração de PWM

- WRAP_PERIOD_FAN (**uint16_t**) → Define o valor máximo do contador PWM para o ventilador.
- WRAP_PERIOD_SPRINKLER (**uint16_t**) → Define o valor máximo do contador PWM para os sprinklers.
- PWM_DIVISER (**float**) → Define o divisor do clock para o PWM.

Variáveis de Exibição (OLED)

- ssd (**ssd1306_t**) → Estrutura para controle do display OLED.
- I2C_PORT (**i2c1**) → Porta I2C usada para o display OLED.
- I2C_SDA (**GPIO 14**) / I2C_SCL (**GPIO 15**) → Pinos de comunicação I2C do OLED.

Matriz de LEDs RGB (WS2812)

- NUM_PIXELS (**25**) → Número de LEDs na matriz RGB (5x5).
- WS2812_PIN (**GPIO 7**) → Pino de controle da matriz WS2812.
- led_r, led_g, led_b (**uint8_t**) → Intensidades das cores vermelho, verde e azul.
- desliga_led[], frame_1[], frame_2[], frame_3[] → Arrays que definem os frames da animação do ventilador.

Outras Variáveis

- last_time (**volatile uint32_t**) → Armazena o tempo do último evento de interrupção (usado para debouncing).
- timer, timer2 (**struct repeating_timer**) → Timers para controlar sprinklers e animação do ventilador.

3. Testes e resultados

3.1-Testes

Foram realizados os seguintes testes para a validação das funcionalidades do sistema.
Obs: todos os testes foram realizados na placa bitdoglab.

Teste 1 : o programa foi iniciado, com o timer dos sprinklers desligado. Primeiro, o eixo Y do joystick foi variado pouco a pouco, para verificar a funcionalidade do monitoramento da temperatura e do acionamento dos motores. As leituras no display foram acompanhadas, com atenção aos limites que causavam o acionamento do LED azul e da matriz de LEDS

Teste 2 : o programa foi iniciado, com o timer dos sprinklers desligado. Primeiro, o eixo X do joystick foi variado pouco a pouco, para verificar a funcionalidade do monitoramento da resistência do LDR. As leituras no display foram acompanhadas, com atenção aos limites que causavam o acionamento do alerta sonoro do buzzer.

Teste 3 : o programa foi iniciado, com o timer dos sprinklers ligado, com a finalidade de testar a rotina periódica dos sprinklers (representada pelo piscar do led vermelho por 6 segundos, seguido de um período apagado de 3 segundos. A interrupção do botão A foi testada, pressionando o botão várias vezes, alternando o estado de ativação dos sprinklers para verificar se de fato os sprinklers seriam apropriadamente desativados com o pressionamento do botão,e reativados com outro pressionamento, e se os alertas pertinentes seriam exibidos no display.

Teste 4 : o programa foi iniciado com todas as suas funcionalidades habilitadas. As ações realizadas durante os 3 últimos testes foram repetidas e executadas em diversas combinações e sequências, com a finalidade de determinar se os diferentes módulos do sistema causavam algum tipo de interferência indesejada entre si.

3.2- Resultados e conclusão

Em raras ocasiões, o pressionamento do botão A causava exibições errôneas e caóticas no display. A causa pode estar relacionada a uma combinação específica de fatores , como a taxa de transferência do protocolo I2C, bem como a desarmonização entre chamadas de funções para o envio de dados ao display em diferentes partes do código (loop principal e em rotinas de interrupção). O evento , contudo, é raro e não interfere no funcionamento do sistema na maioria significativa do tempo. O sensoriamento dos diversos sensores e acionamento dos atuadores não são afetados por esse fenômeno, sendo apenas um problema visual no conteúdo do display que, ademais, é consideravelmente raro por si só.

No geral, os testes demonstraram que o programa possui consistência e confiabilidade em diferentes instâncias de execução, e que todas as funcionalidades pretendidas foram simuladas com sucesso na maioria significante do tempo. A exibição de dados no display é coerente com o esperado de acordo com a movimentação do joystick e a ativação de interrupções,e os atuadores são também acionados de forma coerente e todas as temporizações funcionam adequadamente.

Referências

Behera, Anshu, and Prakash S. Kulkarni. "Smart temperature-dependent cooling of solar panel using Arduino." *2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*. IEEE, 2023.

Azmi, M. S. F. M., et al. "Hybrid cooling system for solar photovoltaic panel." *Journal of Physics: Conference Series*. Vol. 2550. No. 1. IOP Publishing, 2023.

Kamarudin, Muhammad Nizam, Sahazati Md Rozali, and Mohd Saifuzam Jamri. "Active cooling photovoltaic with IoT facility." *International Journal of Power Electronics and Drive Systems* 12.3 (2021): 1494.

https://components101.com/sites/default/files/component_datasheet/LDR%20Datasheet.pdf

https://www.ti.com/lit/ds/symlink/lm35.pdf