



Projeto Final – Unidade 7

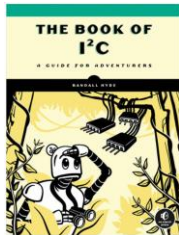
Aluno: *Antonio Carlos Ferreira de Almeida*

Data: 20/02/2025

Sumário

a) Escopo do projeto	4
1 – Apresentação do projeto.	4
2 – Objetivos do projeto.	5
3 – Principais requisitos.	6
4 – Descrição do funcionamento.	6
5 – Justificativa.	7
6 – Originalidade.	8
b) Especificação do hardware.	8
1 – Diagrama em blocos.	8
2 – Função de cada bloco.	9
3 – Configuração de cada bloco.	10
4 – Especificações.	13
5 – Lista de materiais.	14
6 – Descrição da pinagem.	15
7 – Circuito completo do hardware.	15
c) Especificação do firmware/software.	16
1 – Blocos funcionais.	16
2 – Descrição das funcionalidades.	16
3 – Definição das Variáveis e Constantes do Sistema.	16
4 – Fluxograma.	17
5 – Inicialização.	18
6 – Configurações dos registros.	18
7 – Estrutura e formato dos dados.	18
8 – Organização da memória.	19
9 – Protocolo de comunicação.	20
10 – Formato do pacote de dados.	21
d) Execução do projeto	21
1 – Metodologia.	21
2 – Testes de validação.	22
3 – Discussão dos Resultados.	23
4 – Link do Vídeo do projeto funcionando.	24
e) Referência Bibliográficas	25

1 - https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html#pico-1-family	25
2 - https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf	25
3 - https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf	25
4 - https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf	25
5 - https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf	25



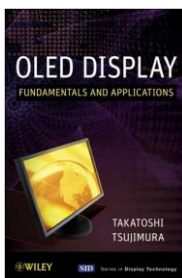
6 - https://learning.oreilly.com/library/view/the-book-of/9781098156596/	25
---	----



7 - https://learning.oreilly.com/library/view/raspberry-pi-pico/9781801814812/	25
---	----



8 - https://learning.oreilly.com/library/view/robotics-at-home/9781803246079/	25
---	----



9 - https://learning.oreilly.com/library/view/oled-display-fundamentals/9781118173077/	25
---	----

Índice de figuras

Figura 1: Limites de Controle de Temperatura	4
Figura 2: Custo Estimado	8

Figura 3: Diagrama de Blocos.....	8
Figura 4: GPIO Envolvidos Com Atuadores.....	10
Figura 5: Alimentação, GND e Controle na Bitdoglab.....	10
Figura 6: Sensor Interno RP2040.....	11
Figura 7: LM35.....	11
Figura 8: Hardware Relativo à Ventilação.....	11
Figura 9: Hardware relativo à lâmpada.....	12
Figura 10: Conexões de Hardware OLED.....	12
Figura 11: Fontes de Alimentação.....	13
Figura 12: Circuito completo (J1- LM35)	15
Figura 12: Circuito completo.....	15
Figura 13: Blocos Funcionais.....	16
Figura 14: Descrição das Funcionalidades.....	16
Figura 15: Fluxograma Eco-Piu-Piu.....	17
Figura 16: Inicialização do Software.....	18
Figura 17: Configurações dos Registros	18
Figura 18: Exemplo de Pacote de dados.....	21

Índice de Tabelas

Tabela 1: Lista de Materiais.....	14
Tabela 2: Pinagem.....	15
Tabela 3: Alimentação.....	15
Tabela 4: Variáveis do Sistema.....	16
Tabela 5: Constantes do Sistema.....	17

a) Escopo do projeto

1 – Apresentação do projeto.

O projeto idealiza uma incubadora, compacta para uso doméstico e pequenos criadores que se propõe a **controlar a temperatura** e garantir um ambiente ideal para a eclosão dos ovos.

Gerenciado por um **Raspberry Pi Pico W – Bitdoglab**, para monitoramento e controle da temperatura.

Consiste em acionar uma lâmpada resistiva para aquecimento e ventilação forçada para circulação de ar, consequentemente resfriando o ambiente.

Com capacidade para 36 ovos e medindo: 20cm de altura, 15cm de largura e 45cm de comprimento, a incubadora pode ser utilizada com diversos tipos de aves, mesclando patas e galinhas, por exemplo. Assim, aumentando a taxa de sucesso na incubação.

Através de um display OLED 128x64, exibi informações de temperatura, aproveitando o sensor que há na placa.

```

19 // Limites de temperatura com histerese
20 #define TEMP_ON 38.0 // Liga a ventoinha acima de 38°C
21 #define TEMP_OFF 36.0 // Desliga a ventoinha abaixo de 36°C


```

Figura 1: Limites de Controle de Temperatura.

Em resumo, manter a temperatura estável de forma automática e permanente dentro dos valores acima apresentados.

Considerações Importantes:

- ***O projeto foi testado com o sensor LM35 e com o sensor interno do RP2040. O LM35, sem dúvidas, é mais indicado, para o projeto final, entretanto, J1 que habilita GPIO28 deve estar na posição correta, como indicado no item: '7 – Circuito Completo do Hardware'.***
- ***Para testar com o sensor interno ao RP2040, apenas siga orientações contidas nos arquivos: 'adc_setup.c e temperature.c'.***
- ***O sensor LM35 apresentou uma temperatura, em média aritmética, em 2 amostragens de 10 medições, um valor cercado em 3,97 a 4,21 graus Celsius, inferior em relação ao sensor interno do RP2040.***
- ***O sensor interno ao RP2040 mede a temperatura do chip, consequentemente a do ambiente, guardadas as proporções.***
- ***A leitura pode ser influenciada pelo aquecimento do RP2040 em uso prolongado.***
- ***Para medições ambientais e de controle mais precisas, sensores externos como LM35, DHT22 ou DS18B20 são recomendados.***
- ***O propósito principal do projeto é aflorar a ideia central do controle e inércia de temperatura, relacionando software e hardware, através da placa Bitdoglab com algoritmo eficaz linguagem 'C'. E como foi mencionado nas especificações gerais do projeto, usar a Bitdoglab como base priorizando o não uso de elementos externos.***
- ***O emprego do sensor interno ao RP2040 pode não ser a melhor escolha para se obter precisão máxima. Contudo, enfatizando o uso da Bitdoglab, o projeto prova a capacidade do sistema como um todo.***

Título do projeto: - Eclo-Piu-Piu .

Link da versão final do algoritmo do projeto:

https://github.com/alfecjo/Embarcatech7/tree/main/ssd1306_oled_bdl_final .

2 – Objetivos do projeto.

Configurar o Bitdoglab para controlar a incubadora de forma eficiente usando C/C++.

Monitorar a temperatura em tempo real com sensor de alta precisão da própria placa.

Automatizar o controle térmico, acionando a resistência luminosa gerando aquecimento e a ventoinha para resfriamento no domínio da temperatura e do tempo.

Regular a umidade com troca constante de ar e ventilação.

Exibir dados no display OLED para facilitar o acompanhamento das condições da incubadora.

Implementar alertas visuais os quais podem apoiar na detecção de falhas ou condições críticas de desligamento através de LEDs.

Garantir uma interface intuitiva de configuração e operação do sistema automático com apenas liga-desliga.

Registrar na tela OLED dados de temperatura em tempo real para verificação do processo de incubação.

Criar uma lógica de segurança pelo software para evitar

superaquecimento ou resfriamento excessivo.

Permitir futuras expansões, como monitoramento remoto, configuração de temperatura via Wi-Fi e ajustes via aplicativo.

3 – Principais requisitos.

- **Capacidade:**
 - Deve incubar o maior número de ovos ao mesmo tempo.
- **Tamanho:**
 - O produto deve ser compacto para uso doméstico.
- **Confiabilidade na Temperatura:**
 - O sistema deve manter a temperatura estável dentro da faixa ideal (entre 36°C e 38°C) com pequena variação (3 graus ><) na maior parte do tempo.
- **Ventilação:**
 - A chocadeira deve possuir ventilação forçada para manter a temperatura homogênea em todos os pontos.
- **Monitoramento e Interface:**
 - A chocadeira deve ter um visor digital para acompanhar a temperatura ao decorrer do tempo.
- **Consumo de Energia:**
 - Deve funcionar com tensão: 127V e 220V.
- **Durabilidade e Material:**
 - O material interno e externo deve ser resistente e fácil de higienizar.
- **Segurança:**
 - Deve haver controle que evite superaquecimento ou redução drástica da temperatura por software que possa prejudicar os ovos.
- **Preço e Garantia:**
 - O produto deve se pagar com uso em curto espaço de tempo e caso apresente defeito, a substituição do componente defeituoso deve ser capaz de resolver o problema.

4 – Descrição do funcionamento.

O sistema é baseado em interrupções de timer, onde:

1. Leitura dos Sensores

O software lê periodicamente o sensor de **temperatura interno** da placa Raspberry Pi Pico W - Bitdoglab. Os valores são processados e comparados com os parâmetros ideais definidos pelo software, para perfeita eclosão de ovos de diversos tipos de aves.

2. Controle de Temperatura

Se a temperatura estiver **abaixo** do valor desejado, o sistema ativa a **resistência de aquecimento através da lâmpada**.

Se a temperatura estiver **acima**, a **ventoinha de resfriamento** é acionada para dissipar o calor.

3. **Exibição no Display OLED**

Os valores de temperatura são **atualizados e exibidos** no display OLED.

4. **Tomada de Decisões e Ações**

O sistema verifica continuamente o sensor e **toma as decisões automaticamente** para manter o ambiente ideal e controlado no domínio do tempo. Interrupções de tempo **ajustam os atuadores principais do sistema (aquecer, ventilar)** conforme necessário.

5. **Registro e Monitoramento**

Os dados são apresentados e atualizados a cada 15 segundos.

Em versões futuras, pode ser implementado um **monitoramento remoto via Wi-Fi**. Não há armazenamento explícito ou implícito de dados.

6. **Segurança e Proteção**

O software escrito em linguagem C/C++ totalmente confiável com lógica clara e perfeita definição do que pode e não pode ocorrer com o sistema.

5 – Justificativa.

O projeto **Eclo-Piu-Piu** tem como objetivo desenvolver um sistema microcontrolado para incubadoras de ovos, garantindo o que há de melhor em tecnologia de controle atualmente.

Um ambiente ideal para o desenvolvimento embrionário e a eclosão dos pintinhos.

A automação desse processo é fundamental para aumentar a taxa de sucesso na incubação, reduzindo perdas e otimizando recursos, em consequência, liberando o proprietário para outras atividades e com um custo muito acessível.

Atualmente, muitas incubadoras domésticas e artesanais dependem de ajustes manuais, o que pode resultar em variações de temperatura e umidade que comprometem o desenvolvimento dos ovos. Um sistema automatizado como o **Eclo-Piu-Piu** melhora significativamente esse controle ao utilizar sensores e atuadores para manter as condições ideais, com necessidade de intervenção apenas para ligar e desligar, após uso.

Além disso, o projeto de software, com pequenas modificações, acrescentando atuadores e sensores, pode ser aplicado em outras áreas e ampliado, como: **estufas comerciais, criadouros e processos industriais que demandam controle térmico e até de umidade**, tornando-se uma solução versátil e escalável.

Dessa forma, a automação desse processo não só melhora a **eficiência e confiabilidade da incubação**, como também reduz o tempo e o esforço necessários para acompanhar o desenvolvimento dos ovos, tornando o processo mais acessível e eficiente para pequenos criadores e entusiastas.

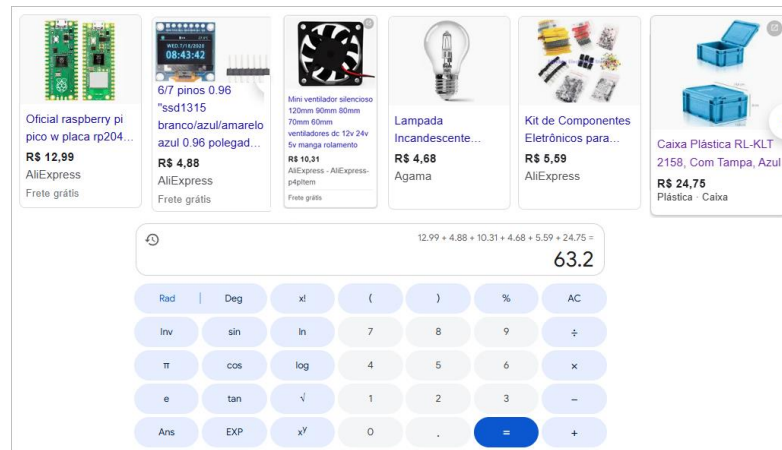


Figura 2: Custo Estimado.

6 – Originalidade.

O **Eclo-Piu-Piu** se destaca por integrar **controle automatizado de temperatura, garantindo a umidade** pela troca do calor do ambiente. Em uma pesquisa na internet não encontrei nada igual, no entanto há experimentos mais simples com μ controlador Arduino.

Diferente de soluções convencionais, que muitas vezes utilizam **termostatos analógicos** ou **controladores simples**, este projeto combina o sensor de precisão com um **sistema de atuação inteligente, desenvolvido em linguagem C, que se traduz em robustez e confiança** para manter as condições ideais de incubação. Além disso, a inclusão de um **display OLED** para exibição da temperatura assegura robustez no processo.

Outro diferencial do **Eclo-Piu-Piu** é sua **versatilidade**, podendo ser adaptado para diferentes aplicações com pequenas modificações, como **estufas, controle térmico industrial**. A modularidade do código e a estrutura aberta (Open Source) do projeto permitem **personalizações e melhorias futuras por qualquer indivíduo que se interessar em controle de temperatura simples, evidentemente, partindo de um código já testado e consolidado**, incentivando experimentação e inovação na área de automação para pequenos criadores e entusiastas da avicultura e até outras áreas.

b) Especificação do hardware.

1 – Diagrama em blocos.

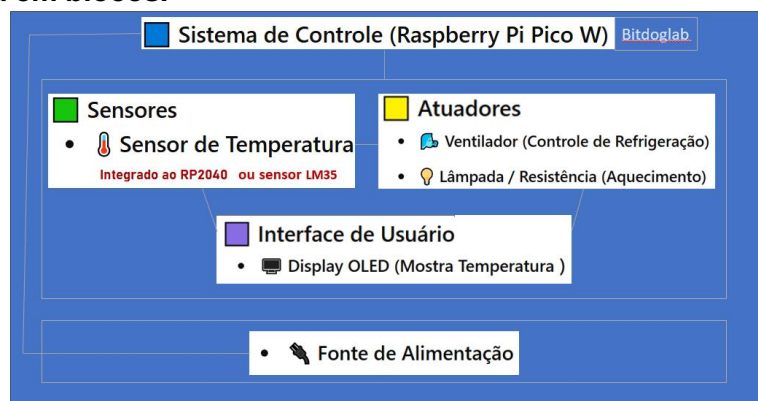


Figura 3: Diagrama de Blocos.

2 – Função de cada bloco.

- **Sistema de Controle (Raspberry Pi Pico W) Bitdoglab:** é um projeto Escola 4.0 que democratiza o ensino em μ controladores usando hardware aberto, idealizado pelo Prof. Dr. Fabiano Fruett e sua equipe.
 - Responsável por processar os sinais e controlar os periféricos através de software escrito em linguagem C/C++, pois é através desta lógica que todo o processo de controle é executado.

Os periféricos utilizados na solução são listados a ver:

- **ADC (Conversor Analógico-Digital):** Lê o sinal do sensor de temperatura (interno ao RP2040 ou LM35).
 - **I²C:** Comunicação com o display OLED para exibição das informações ao usuário.
 - **GPIO:** Controle do ventilador, lâmpada de aquecimento.
 - Embora haja outros periféricos na placa de aprendizagem **Bitdoglab**, apenas uns poucos foram utilizados no intuito de alcançar a simplicidade e eficiência num equilíbrio harmônico entre as partes.
- **Sensores**
 - O sensor de temperatura interno ao RP2040 foi utilizado, bem como o LM35, otimizando a solução para uso da Bitdoglab e demonstração da eficácia do sistema, contudo, o código está preparado para receber o sensor LM35, se tornando mais eficiente. Leituras mais realísticas são perfeitamente possíveis com o emprego de outros sensores sem perda do contexto geral ao qual a solução foi produzida.
 - O **RP2040** possui um **sensor de temperatura interno** embutido em seu **Conversor Analógico-Digital (ADC)**. Esse sensor mede a temperatura do próprio chip e fornece uma leitura analógica que pode ser convertida para temperatura em graus Celsius, também apresenta uma leitura mais elevada, em torno de 3.5 a 4.5 graus Celsius, pois acumula a temperatura do chip.
 - **Atuadores**
 - A Ventoinha tem a função de troca e circulação de ar saturada a temperatura limite e fazer a substituição de calor com o ambiente, assim que necessário. O processo se inicia sempre que a temperatura não satisfaz as exigências preestabelecidas. Isso acontece no domínio do tempo.
 - Lâmpada têm a função de inserir calor até a temperatura limite, elevando a temperatura do ambiente, assim que necessário. O processo se inicia sempre que a temperatura não satisfaz as exigências preestabelecidas. Isso acontece no domínio do tempo.
 - **Interface do Usuário**
 - **OLED 128x64:** Interface com usuário de modo ativo. Responsável em fornecer visualmente o valor da temperatura

ao usuário afim de informar as ocorrências do processo. Foi utilizado o OLED que vêm com a Bitdoglab sem prejuízo ao projeto:

- **Fontes de Alimentação**

- No processo de prova de amostra do projeto foram utilizadas uma entrada da concessionária AC 127 Vac:
 - ✓ Fonte 5Vcc em um adaptador comum, o qual alimentou a placa Bitdoglab,
 - ✓ Fonte 12Vcc em um adaptador comum, o qual alimentou a ventoinha,
 - ✓ Utilizado como fornecimento de energia a lâmpada.

3 – Configuração de cada bloco.

Sistema de Controle (Raspberry Pi Pico W) Bitdoglab

Configurado para orquestrar toda a aplicação, bem como fornecer sinal digital de controle ao atuador FAN_PIN (ventoinha), idem ao segundo atuador LIGHT_PIN (lâmpada), através da GPIO 8 e 9.

```
15 // Definição dos pinos de saída aos atuadores
16 #define FAN_PIN 8
17 #define LIGHT_PIN 9
```

Figura 4: GPIO Envolvidos Com Atuadores.

Além disso fornece 3.3Vcc e GND, a ambos sistemas de acionamentos, lâmpada e ventoinha, disparados por circuito discreto envolvendo resistores e transistores BC337.

- A seguir é apresentada as conexões que saíram do IDC – 14PIN, como demonstrado na figura abaixo:

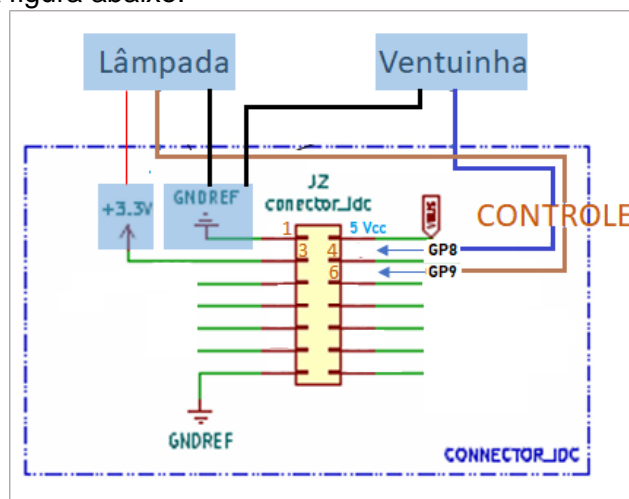


Figura 5: Alimentação, GND e Controle na Bitdoglab.

Sensores

- Sensor de temperatura interno ao RP2040 **não requer conexões externas**, pois já está embutido no microcontrolador. No entanto,

para garantir medições mais estáveis, algumas configurações podem ser aplicadas:

ADC_VREF	Alimentação de referência do ADC (3,3V por padrão).
GND	Referência de terra para medições.
Canal 4 do ADC	Entrada do sensor de temperatura interno (configurado via software).

Figura 6: Sensor Interno RP2040.

- **LM35**

- Para usar o **LM35** no lugar do sensor interno do RP2040, verificar posição de **J1**, também, foram feitas e comentadas algumas alterações no código, pois o **LM35** fornece uma saída analógica proporcional à temperatura (**10mV/°C**), enquanto o sensor interno do RP2040 usa uma equação específica baseada na tensão interna. Verificar os arquivos 'adc_setup.c e temperature.c', seguindo as orientações contidas neles. (**LM35 padrão**: $\pm 0,5^{\circ}\text{C}$ a 25°C e $\pm 1^{\circ}\text{C}$ na faixa de -55°C a 150°C - Datasheet)

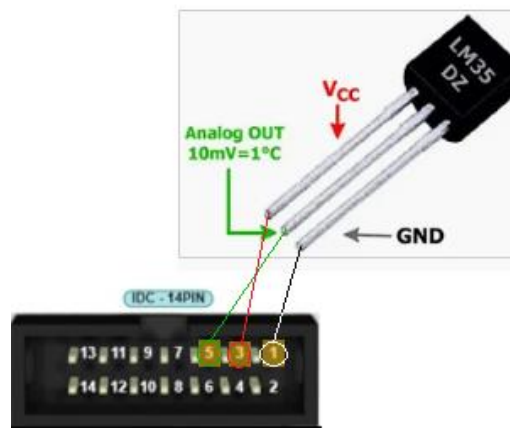


Figura 7: LM35.

Atuadores

- A configuração da ventilação consiste em um circuito de componentes discretos a saber:

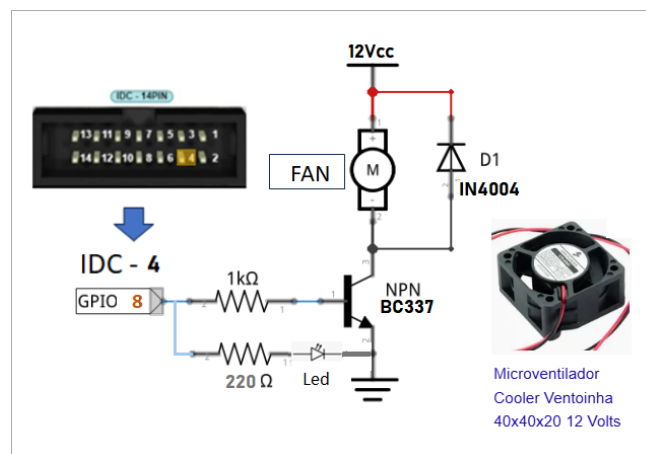


Figura 8: Hardware Relativo à Ventilação.

- A configuração da lâmpada consiste em um circuito de componentes discretos a saber:

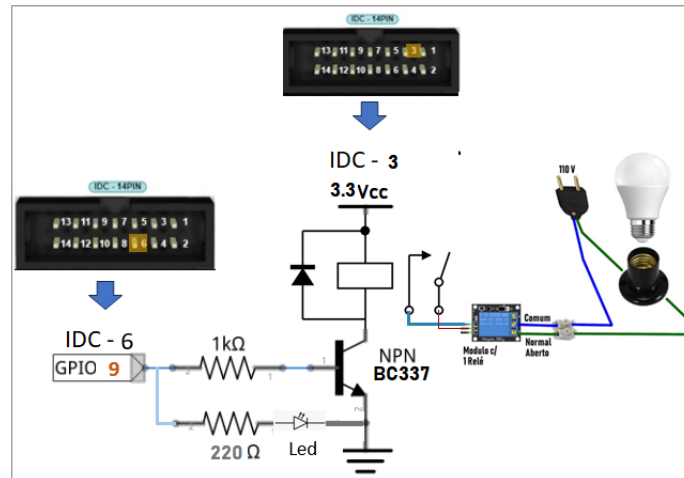


Figura 9: Hardware relativo à lâmpada.

Interface do Usuário

- **Conexões:**
 - **SDA:** Conectado a GP14.
 - **SCL:** Conectado a GP15.
 - **VCC:** Alimentado com 3.3Vcc.
 - **GND:** Conectado ao GND.

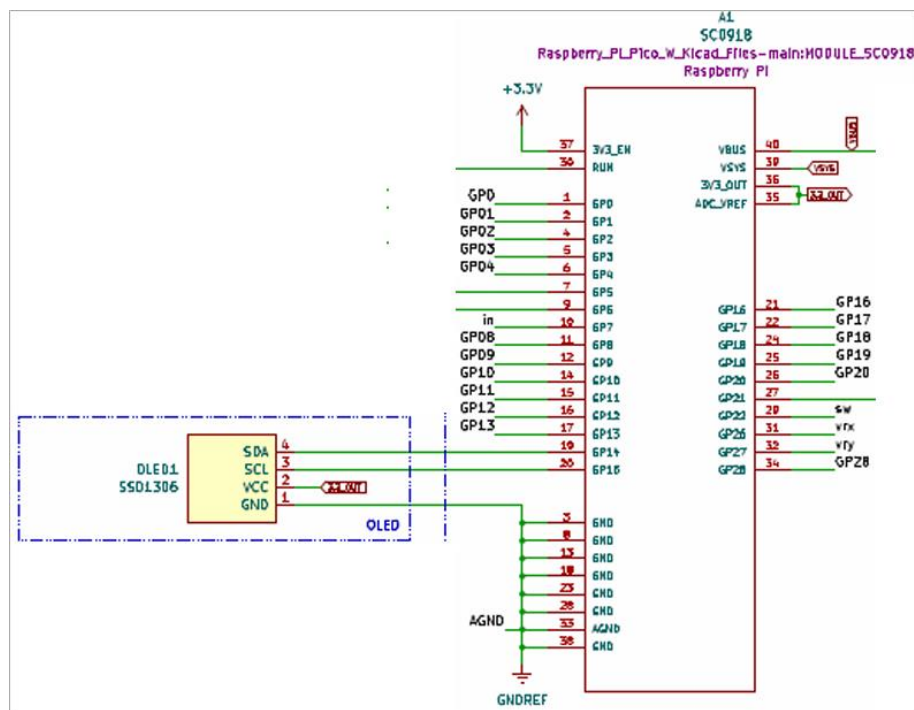


Figura 10: Conexões de Hardware OLED.

Fontes de Alimentação

- O sistema pode, futuramente, ser alimentado por fonte única, contudo, na configuração de prova de projeto, foram utilizadas configuração para alimentação do sistema, fontes comerciais que fornecem 5Vcc e 12 Vcc. 5Vcc foi utilizado através de conector USB, o qual alimentou a Bitdoglab e 12 Vcc foi utilizado na alimentação da ventoinha. Os esquemáticos de pinagem encontram-se no item: “3. Configuração de cada bloco – Atuadores”.

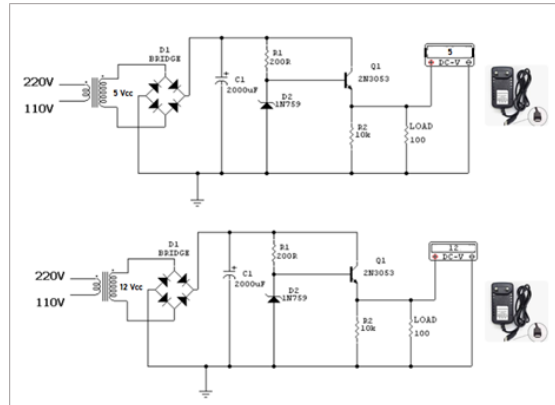


Figura 11: Fontes de Alimentação.

4 – Especificações.

- **Capacidade:**

- A chocadeira tem capacidade para 36 ovos, permitindo a incubação simultânea. Isso atende tanto consumidores domésticos quanto pequenos produtores.

- **Tamanho:**

- Medindo: 20cm de altura, 15cm de largura e 45cm de comprimento, têm design **compacto e otimizado**, ocupando pouco espaço e sendo ideal para uso doméstico, sem comprometer a eficiência na distribuição térmica e na ventilação.

- **Confiabilidade na Temperatura:**

- O sistema de controle de temperatura utiliza como amostra o sensor interno do **RP2040** ou o **sensor LM35**, para garantir uma precisão de **±0,5°C a 1,0 °C**, mantendo a faixa ideal entre **37,5°C e 38°C em média**. A regulação é automática, ajustando-se para evitar variações bruscas que poderiam comprometer o desenvolvimento dos embriões.

- **Ventilação:**

- A chocadeira conta com um **sistema de ventilação forçada**, utilizando um **cooler de 100 mm** de diâmetro para distribuir uniformemente o calor no interior da câmara. Isso evita áreas de superaquecimento ou resfriamento, garantindo a uniformidade da incubação.

- **Monitoramento e Interface:**

- A temperatura é exibida em **tempo real no display OLED**, permitindo que o usuário acompanhe facilmente o funcionamento da chocadeira. LEDs **alertam** visualmente para indicar parada no funcionamento.
- **Consumo de Energia:**
 - O sistema opera com **127V e 220V** (bivolt automático), garantindo compatibilidade com qualquer rede elétrica. O consumo energético é otimizado para ser econômico, funcionando de maneira eficiente sem desperdício de energia.
- **Durabilidade e Material:**
 - A estrutura é feita de **plástico resistente ao calor e fácil de higienizar**, reduzindo o acúmulo de umidade e evitando a proliferação de fungos e bactérias. Isso aumenta a vida útil do equipamento e melhora as condições sanitárias para os ovos.
- **Segurança:**
 - O controle de temperatura possui **proteção contra superaquecimento** por software que evita variações bruscas de temperatura. Caso ocorra alguma falha, os LEDs deixam de piscar, alertando **visualmente** o usuário para a necessidade de intervenção.
- **Preço e Garantia:**
 - O projeto foi desenvolvido para oferecer **um ótimo custo-benefício**, garantindo que o investimento se pague rapidamente com uma alta taxa de eclosão. Além disso, o sistema modular permite fácil manutenção e troca de componentes caso necessário. Pois o único componente que guarda estado é a Bitdoglab. Caso ocorra problemas em outros componentes, a substituição e ligação de forma idêntica resolve o caso.

5 – Lista de materiais.

Quantidade	Descrição
01	Raspberry Pi Pico W - Bitdoglab
01	OLED 128x64 0.96" I ² C SSD1306 - Bitdoglab
01	Case para comportar os ovos
01	Fonte 127 Vac ~ 12 Vcc 2 A - comum
01	Ventoinha 12 Vcc
01	Fonte 127 Vac ~ 5 Vcc 2 A - comum
01	Lâmpada filamento resistivo
02	Resistores 220Ω 1/4W
02	Resistor 1KΩ 1/4W
01	LM35 – sensor de temperatura
02	Transistores BC337
01	Diodo IN4004
01	Conector IDC Macho 2x7 180° - Bitdoglab
02	Led 1W - verde
01	Relé 3Vcc – 10 A 127 Vac ~250 Vac
01	Tomada 10 A
01	Bocal para Lâmpada
0,5 m	Fios d 2 mm

Tabela 1: Lista de Materiais.

6 – Descrição da pinagem.

IDC-14PIN	Bitdoglab	Descrição
VCC - 3	3.3V	Alimentação do circuito da lâmpada
GND - 1	GND	GND - Ambos atuadores
4	GP8 Controle FAN_PIN 8 - Ventoinha	GPIO 8 – Controle por software
6	GP9 Controle LIGHT_PIN 9 - Lâmpada	GPIO 9 – Controle por software
5	GP28 (Verificar: J1)	Sensor Temperatura LM35
N/A	GP14 SDA - OLED	Linha de dados
N/A	GP15 SCL - OLED	Linha de clock
3	3.3 Vcc - OLED	Alimentação OLED
1	GND - OLED	Comum OLED
N/A	J1 – Verificar posição	Figura 12: Circuito completo (J1- LM35)

Tabela 2: Pinagem.

Alimentação	Origem	Descrição
127 Vac	Concessionária	Alimentação Lâmpada
127 Vac	Concessionária	Fonte comum 5 Vcc
127 Vac	Concessionária	Fonte comum 12 Vcc
5 Vcc	Fonte comum	Alimentação Bitdoglab
12 Vcc	Fonte comum	Alimentação Ventoinha.
3.3 Vcc	Bitdoglab	Usado no circuito lâmpada

Tabela 3: Alimentação.

7 – Circuito completo do hardware.

- Todas as referências com relação a pinagem, encontram-se no item: “6 - Descrição da Pinagem - Tabela 2: Pinagem”, neste documento.

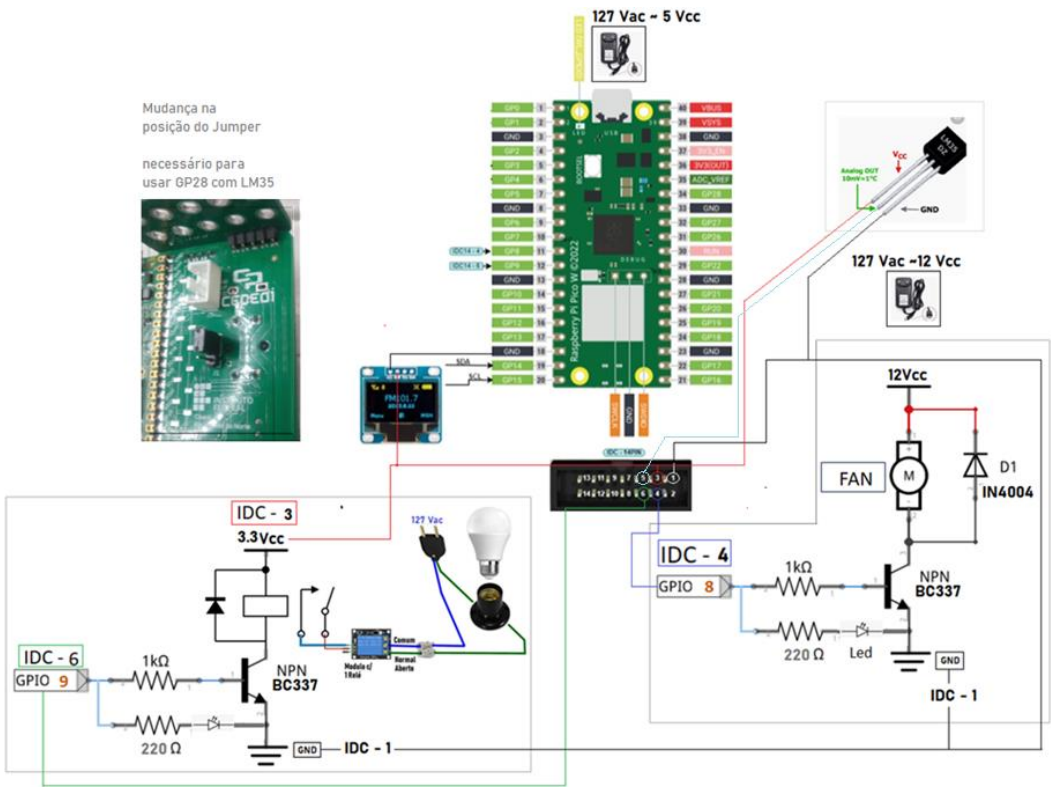


Figura 13: Circuito completo.

c) Especificação do firmware/software.

1 – Blocos funcionais.

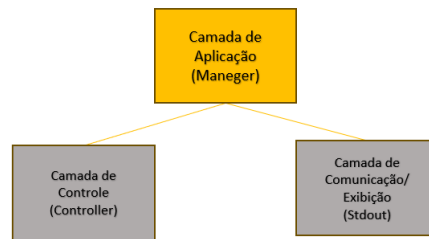


Figura 14: Blocos Funcionais.

2 – Descrição das funcionalidades.

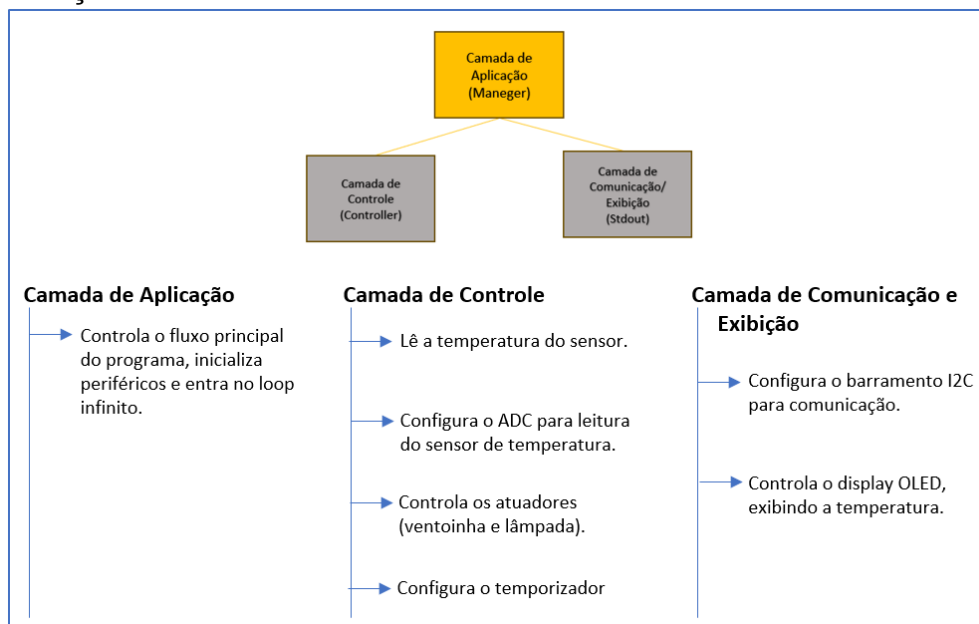


Figura 15: Descrição das Funcionalidades.

3 – Definição das Variáveis e Constantes do Sistema.

Variável	Tipo	Descrição
fan_on = false;	bool	Estado inicial da ventoinha
light_on = false;	bool	Estado inicial da lâmpada
temperatura	float	Lê a temperatura
timer	repeating_timer	Manipula a estrutura repeating_timer
disp	ssd1306_t	Manipula a estrutura ssd1306_t, relativo ao display
raw	uint16_t	Lê o valor bruto do ADC
voltage	float	Converte para tensão

Tabela 4: Variáveis do Sistema.

Constantes	Tipo	Descrição
------------	------	-----------

FAN_PIN	#define FAN_PIN 8	Definição dos pinos de saída aos atuadores - Ventoinha
LIGHT_PIN	#define LIGHT_PIN 9	Definição dos pinos de saída aos atuadores - Lâmpada
TEMP_ON	#define TEMP_ON 38.0	Limite SUPERIOR de temperatura com histerese
TEMP_OFF	#define TEMP_OFF 36.0	Limite INFERIOR de temperatura com histerese
conversion_factor	float	Utilizado como: Fator de conversão para 12 bits

Tabela 5: Constantes do Sistema.

4 – Fluxograma.

- “Eclo-Piu-Piu”

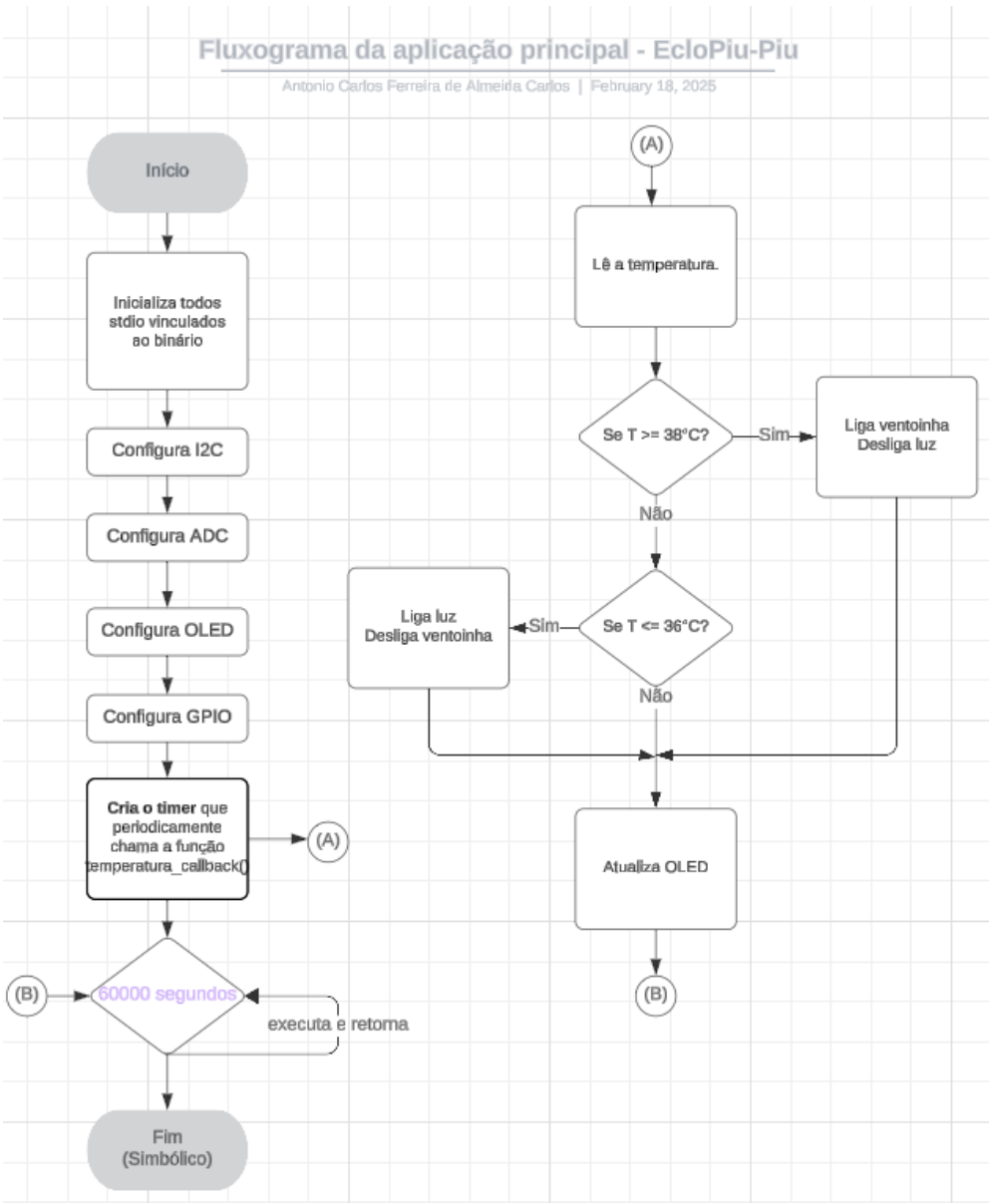


Figura 16: Fluxograma Eclo-Piu-Piu.

5 – Inicialização.

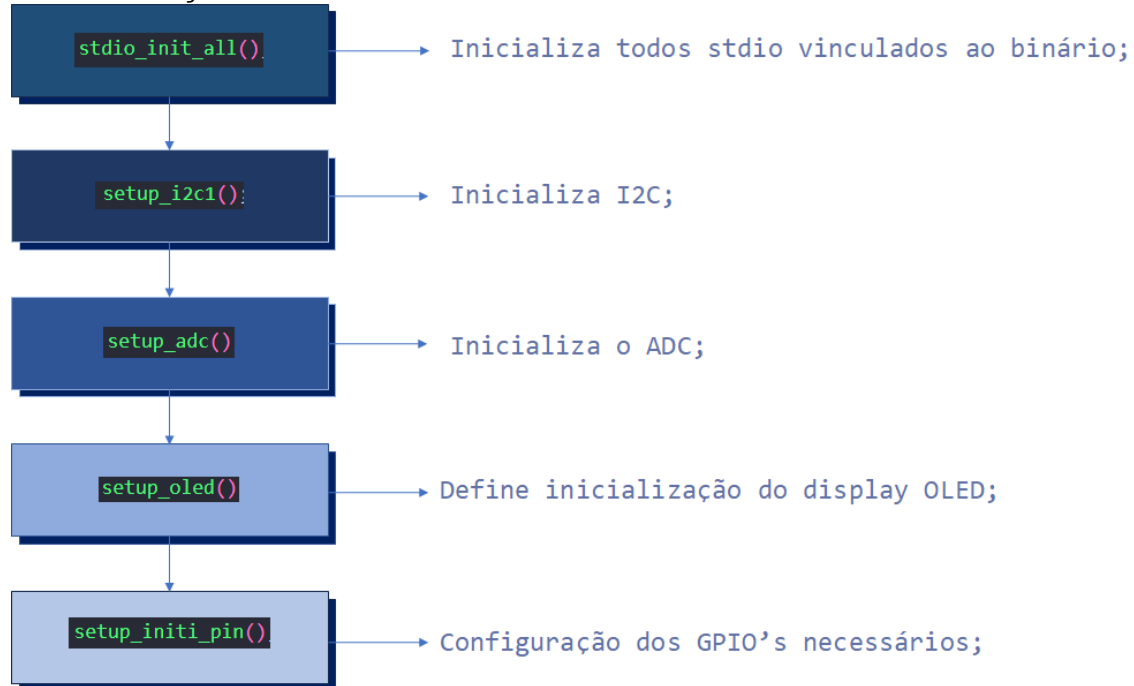


Figura 17: Inicialização do Software.

6 – Configurações dos registros.



Figura 18: Configurações dos Registros

7 – Estrutura e formato dos dados.

✓ **I2C (OLED SSD1306):**

Comandos e dados são enviados em **bytes** via I2C.

Endereço do dispositivo, comando de controle, e dados.

✓ **ADC:**

Manipula valor digital entre 0 e 4095 (12 **bits**) correspondente à leitura do sinal analógico e para isso se utiliza de uma constante do tipo: float (conversion_factor).

Se utiliza do tipo: unsigned short para ler o valor bruto do ADC (raw), bem como o tipo: float (voltage), após converter para tensão.

✓ **OLED:**

Manipula o OLED, fornece valor em hexadecimal ao acessar seu endereço (0x3C). Utiliza uma variável de estrutura do tipo: `ssd1306_t` (disp) para acessar seus membros.

Utiliza uma matriz do tipo: char (temp_str) que armazena a string que irá para o display.

✓ **GPIO:**

Foram utilizados valores digitais do tipo: int (0 ou 1) para controle de pinos de entrada e saída.

8 – Organização da memória.

➤ Endereços de memória utilizados indiretamente:

- Periférico I2C1 (OLED SSD1306)
 - O barramento **I2C1** foi utilizado para comunicação com o display **OLED SSD1306**, que possui o endereço **0x3C** no barramento I2C.
 - endereço de base:
 - **I2C1_BASE = 0x40044000**
 - registros principais + offset:
 - **IC_CON (0x40044000 + 0x00)** → Configuração do barramento I2C
 - **IC_TAR (0x40044000 + 0x04)** → Endereço do dispositivo alvo (0x3C)
 - **IC_DATA_CMD (0x40044000 + 0x10)** → Envio/recebimento de dados
- Periférico ADC
 - endereço de base:
 - **ADC_BASE = 0x4004C000**
 - registros principais + offset:
 - **CS (0x4004C000 + 0x00)** → Controle do ADC
 - **RESULT (0x4004C000 + 0x04)** → Resultado da conversão ADC (12 bits)
 - **FCS (0x4004C000 + 0x08)** → Configuração do FIFO do ADC
- GPIO (Pinos 8 e 9)
 - endereço de base:
 - **SIO_BASE = 0xD0000000**

- registros principais:
 - **GPIO_OUT (0xD0000010)** → Define o estado (0 ou 1) dos pinos
 - **GPIO_OE (0xD0000020)** → Configura se o pino é entrada ou saída
 - **GPIO_CTRL (0x40014000 + (N * 8))** → Configurações individuais para cada GPIO
- Endereços de memória utilizada diretamente, mas que são alocados pelo compilador:
 - **Variáveis globais:**
 - “disp” - Estrutura global do display
 - **Variáveis de buffer:**
 - “temp_str” – Relacionado a função para atualizar a temperatura no OLED
 - **A utilização da memória pelas variáveis de retorno de função,** são utilizadas diretamente, contudo, são autoexplicativas aos tipos de retorno em relação ao papel que a função presta a solução. A intenção de confronta-las exige antes uma verificação da lógica empregada e de que forma o tipo de retorno agrega na continuação do fluxo do ponteiro de programa.

9 – Protocolo de comunicação.

- I2C (Inter-Integrated Circuit) é um protocolo serial síncrono que permite a comunicação entre múltiplos dispositivos usando apenas dois fios:
 - SDA (Dados, GPIO 14)
 - SCL (Clock, GPIO 15)
 - Funcionamento do I2C com o OLED SSD1306
 - O RP2040 atua como mestre e o OLED SSD1306 como escravo, com endereço 0x3C.
 - A comunicação ocorre em pacotes de bytes enviados pelo barramento I2C, organizados em:
 - ❖ Byte de Endereço: O primeiro byte identifica o dispositivo no barramento.
 - ❖ Byte de Controle: Define se os dados seguintes são comandos ou dados gráficos.
 - ❖ Bytes de Dados: Podem ser comandos de configuração (como ligar/desligar) ou bytes de pixels.
- Na aplicação o uso do **ADC (Analog-to-Digital Converter)** converte um sinal analógico (tensão variável) em um valor digital de **12 bits**.
 - Fluxo de Comunicação com o ADC
 - Configurar o ADC e selecionar o canal correto.
 - Acionar a conversão (start).
 - Aguardar a conversão ser concluída.
 - Ler o valor convertido.
- Os **GPIOs (General-Purpose Input/Output)** fazem parte dos atuadores
 - Modos de Operação
 - Entrada (Input) → Lê o estado lógico do pino (0 ou 1).
 - Saída (Output) → Define o nível lógico do pino (0 ou 1).

10 – Formato do pacote de dados.

- Pacote I2C - Comunicando com o OLED SSD1306
 - [Start] -> [Endereço do OLED (0x3C)] -> [Byte de Controle] -> [Dados/Comandos] -> [Stop]
 - Start (1 bit) → Indica o início da comunicação.
 - Endereço do Dispositivo (7 bits) → Endereço do OLED (0x3C).
 - Bit R/W (1 bit) → Define se a operação é de leitura (1) ou escrita (0).
 - ACK/NACK (1 bit) → Confirmação do recebimento do dado.
 - Byte de Controle (8 bits) → Define se os bytes seguintes são comandos ou dados gráficos.
 - ❖ 0x00 → Envio de comandos (configuração do display).
 - ❖ 0x40 → Envio de dados gráficos (pixels na tela).
 - Bytes de Dados (N bytes) → Sequência de comandos ou pixels.
 - Stop (1 bit) → Finaliza a comunicação.
- Pacote ADC – Conversor Analógico-Digital
 - Formato dos Dados do ADC
 - Valor bruto (Raw Data, 12 bits): 0b0000_0000_0000 a 0b1111_1111_1111 (0 a 4095).
 - Valor convertido para tensão: tipo: float armazenado em 32 bits (IEEE 754).
 - ❖ [Canal Selecionado (8 bits)] -> [Valor Bruto (12 bits)] -> [Conversão para tipo: float (32 bits)]
- Pacote GPIO – Comunicação Digital
 - Formato dos Dados GPIO
 - [Endereço do Pino (8 bits)] -> [Estado (1 bit: 0 ou 1)]
 - ❖ [0000_1000] -> [1] (Pino 8, estado HIGH)
 - ❖ [0000_1001] -> [0] (Pino 9, estado LOW)

Periférico	Formato do Pacote	Exemplo
I2C (OLED SSD1306)	[Start] + [0x3C] + [0x00 ou 0x40] + [Dados] + [Stop]	[Start] -> [0x3C] -> [0x00] -> [0xAF] -> [Stop]
ADC (Conversor Analógico-Digital)	[Canal] + [Valor Bruto (12 bits)] + [Valor Convertido (32 bits)]	[0000_0000] -> [0000_1010_1100] -> [1.38V]
GPIO (Entrada/Saída Digital)	[Endereço (8 bits)] + [Estado (1 bit)]	[0000_1000] -> [1]

Figura 19: Exemplo de Pacote de dados.

d) Execução do projeto

1 – Metodologia.

- Basicamente, o projeto cresceu empregando uma abordagem Top-Down e para o desenvolvimento da incubadora automatizada, foram realizadas pesquisas sobre os parâmetros ideais para incubação de ovos, incluindo temperatura, umidade e ventilação. Além disso, foram analisadas abordagens técnicas para o controle inercial térmico automatizado, inércia considerando o

uso de microcontroladores e sensores apropriados. Estudos sobre a implementação de sistemas embarcados também foram conduzidos, permitindo a escolha adequada da plataforma de desenvolvimento e dos componentes necessários para a automação.

O hardware central escolhido para o projeto foi o Raspberry Pi Pico W – Bitdoglab, devido à sua capacidade de processamento eficiente, suporte a comunicação e compatibilidade com sensores e atuadores, essenciais para o controle térmico. Testes demonstraram uma boa reação ao controle e resposta rápida, sem perder a eficiência, excitando aquecimento com uma lâmpada resistiva, enquanto a ventoinha foi integrada para a circulação de ar e resfriamento do ambiente. Um display OLED 128x64 foi incorporado para exibição das informações de temperatura, aproveitando o sensor embutido na placa.

As funcionalidades do software foram definidas com base nos requisitos do projeto e no funcionamento esperado de uma incubadora eficiente, devidamente documentados no item: “3 – Principais Requisitos”.


A implementação do software foi realizada utilizando a **IDE Visual Studio Code**, configurada para suportar o desenvolvimento em **C/C++ (GCC)**. O ambiente foi preparado com a instalação do **CMake**, essencial para a compilação e gerenciamento do projeto. Além disso, foram utilizadas bibliotecas específicas para comunicação com os sensores, display e controle dos atuadores devidamente documentados (todos de fácil localização na internet).

Durante o desenvolvimento, adotamos práticas de **Test-Driven Development (TDD)**, garantindo que cada funcionalidade fosse testada individualmente antes da integração. Testes unitários foram escritos para validar a lógica de acionamento da lâmpada e da ventoinha, além da leitura precisa dos sensores.

A depuração foi realizada utilizando ferramentas do **Visual Studio Code**, incluindo debug direto no hardware via **pico-debug** e monitoramento serial para análise dos dados do sensor em tempo real. Para a depuração de software, foi empregada uma **sonda de depuração**, permitindo verificar o comportamento do código e detectar falhas no desenvolvimento. Já para a depuração de hardware, foram utilizados **multímetro, osciloscópio e fonte CC**, garantindo a verificação precisa dos sinais elétricos e do funcionamento correto dos componentes físicos do sistema. Foram testadas diferentes condições de temperatura e acionamento dos componentes para assegurar a confiabilidade do sistema.

2 – Testes de validação.

Abaixo encontram-se os testes e validações de software. A documentação com o código encontra-se no repo: <https://github.com/alfecjo/Embarcatech7>.

-  Adicionando teste0001 Bitdoglab - blink_bdl
 - Este teste consiste em fazer um LED piscar utilizando a Raspberry Pi Pico - Bitdoglab. A prova foi essencial para o reconhecimento da pinagem na simulação de utilizar os GPIO'S que deverão acionar a LED_PIN_LIGHT (iluminação geradora de calor) e a LED_PIN_FAN (ventilação retirando ar quente do ambiente).

- Código completo em C: blink_bdl.c Arquivo CMake: Cmakefiles.txt
- Adicionando teste0002 Bitdoglab - button_buzzer_bdl
 - Este teste foi essencial ao reconhecimento do canal ADC, embora simples, várias conclusões de hardware foram alcançadas e consiste na prática do Botão A que aciona o Buzzer.
 - Código completo em C: button_buzzer_bdl.c Arquivo CMake: Cmakefiles.txt

No **repo** acima, estão contidos mais testes, executados principalmente **antes, durante** e **depois** da criação do projeto, os quais, para cumprir o requisito de **25 páginas**, ficaram de fora.

Os testes desempenharam um papel fundamental no reconhecimento da pinagem da Bitdoglab e na construção das funcionalidades, através da abordagem Test-Driven Development (**TDD**).

Com os testes estabelecidos, as funcionalidades foram desenvolvidas se submetendo e confirmando as restrições estabelecidas pelos mesmos. Assim, foi possível validar o comportamento esperado antes mesmo da implementação completa, economizando tempo e aumentando assertividade.

Essa metodologia garantiu maior confiabilidade ao sistema, reduzindo erros e facilitando a manutenção e evolução do projeto.

A confirmação da aplicação da abordagem TDD, fica explícita ao notar a disposição dos arquivos .c e .h e a clara separação de responsabilidades contidas em cada um. Sem esta estratégia de modularização, a aplicação da técnica fica impraticável.

Funcionalidades modularizadas em arquivos que atendem a responsabilidade única podem receber testes que vão direto ao ponto, sem conflito e sem malabarismos desnecessários.

3 – Discussão dos Resultados.

- Objetivos alcançados:
 - ✓ **A Bitdoglab** foi configurada para controlar a temperatura da incubadora de forma eficiente usando C/C++.
 - ✓ **A funcionalidade para monitorar a temperatura** em tempo real com sensor de alta precisão da própria placa foi alcançado.
 - ✓ **Como consequência, automatizar o controle térmico**, de forma eficiente acionando a resistência luminosa gerando aquecimento e a ventoinha para resfriamento controlando a grandeza temperatura no domínio do tempo.
 - ✓ A variação e as trocas sistemáticas de arejamento tornam possível manter a **umidade** em igual índice do ambiente, o que é favorável a eclosão de ovos.
 - ✓ **Ao exibir dados no display OLED**, mantém o usuário atualizado e facilita o acompanhamento das condições da incubadora.
 - ✓ **A indicação luminosa** alerta visualmente o usuário das condições de trabalho ou falha do sistema.

- ✓ **O sistema como um todo garante uma interface intuitiva** com operação automática com apenas liga-desliga.
- ✓ **Escrito em linguagem C/C++, que por si só, dispensa comentários, possui lógica de segurança** por software para evitar superaquecimento ou resfriamento excessivo.
- ✓ **Da forma como está, a solução permiti futuras expansões**, como monitoramento remoto, configuração de temperatura via Wi-Fi e ajustes via aplicativo, rotacionamento dos ovos e muito mais.

4 – Link do Vídeo do projeto funcionando.

Link do YouTube, não listado: <https://www.youtube.com/watch?v=fSJytcEURIA>.



e) Referência Bibliográficas

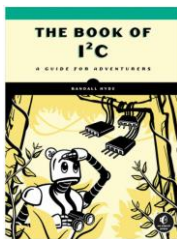
1 - <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html#pico-1-family>

2 - <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>

3 - <https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf>

4- <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

5 - <https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf>



6 -

<https://learning.oreilly.com/library/view/the-book-of/9781098156596/>



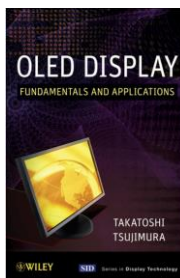
7 -

<https://learning.oreilly.com/library/view/raspberry-pi-pico/9781801814812/>



8 -

<https://learning.oreilly.com/library/view/robotics-at-home/9781803246079/>



9 -

<https://learning.oreilly.com/library/view/oled-display-fundamentals/9781118173077/>