Embarcados.com.br

Curso 02

Firmware - Programação de Sistemas Embarcados I

Projeto Final – Firmware - Programação de Sistemas Embarcados I

Rogerio da Silva Oliveira

Fortaleza - CE

2025

Sumário

1 Estufa Didática Rogério 1.0

Projeto submetido ao curso de "**Firmware - Programação de Sistemas Embarcados**" do Embarcados.com.br como atividade final por Rogerio da Silva Oliveira

O sistema monitora a temperatura, umidade e luminosidade de uma estufa através dos sensores DHT11 e LDR, integrantes do módulo de desenvolvimento LAB01. Os valores medidos são apresentados em tempo real num display OLED.

Um rele é acionado, pino GPIO 14 ou LED vermelho no LAB01, quando a temperatura lida no sensor é menor que o valor do Setpoint. Também esta indicação é apresentada no display na tela principal.

São três modos de controle da saída:

- Automático,
- Sempre ligado,
- Sempre desligado.

Os parâmetros do Setpoint e Modo de controle sempre tem uma cópia gravada na memória NVS, para manter a integridade numa falha de energia ou reset. Os valores são gravados quando são alterados ou na inicialização.

Os valores e parâmetros do sistema podem ser monitorados pela porta serial.

O registro de dados está num arquivo texto na memória NVS interna ao ESP32

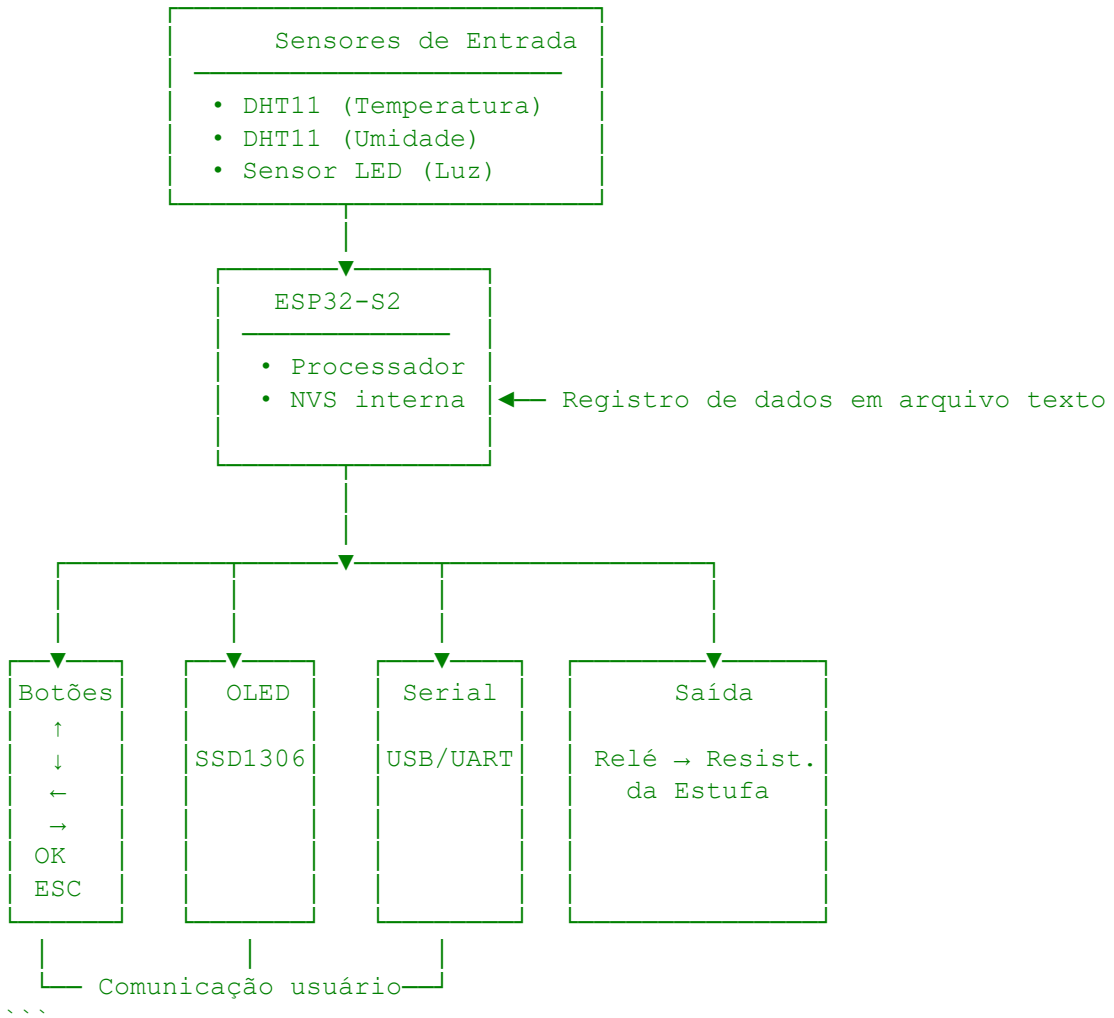
Figura 1. Elementos do sistema

1.1 Repositório no github

https://github.com/rgroliveira/estufa

2 Diagrama de Blocos do Hardware:

Diagrama de blocos de Sistema com um microcontrolador ESP32-S2. Na entrada tem três sensores, um sensor de temperatura DHT11, um sensor de humidade DHT11 e um sensor LED de luminosidade. Como interface tem um display OLED. Para configuração são seis botões, quatro de navegação (Esquerda, direita cima e baixo) e dois botões de decisão, OK e ESC. Tem uma comunicação pela porta serial para o usuário. A saída é de um relé para controle da resistência de uma estufa. O registro de dados está num arquivo texto na memória NVS interna ao ESP32.

```


```

3 Estrutura do Menu

Teclas de Navegação: Cima (BT1), Baixo (BT4), Esquerda (BT2) e Direita (BT3)

Tecla de comando: OK (BT5) e ESC (BT6)

Tecla de Controle do Setpoint, dentro do menu setpoint '+' (BT6) e '-' (BT5)

3.1 Telas do Menu:

Figura 2. Navegação das telas


```
┌──────────────────────┐
├─ 1 Principal         |
├     Estufa <versão>  |
├     Rogerio IFCE     |
├     Temperat. =      |
├     Umidade   =      |
├     Brilho    =      |
├                      |
├     Set point =      |
├     Rele: Desligado  |
├──────────────────────┤
├─ 2  ===SETPOINT===   |
├     +: Incrementa    |
├     -: Decrementa    |
├     Set Point = xx°C |
├──────────────────────┤
├─ 3  ===CONTROLE====  |
├     A: Automatico    |
├     L: Ligado        |
├     D: Desligado     |
├                      |
├     <OK> Para salvar |
├──────────────────────┤
├─ 4  ====ARQUIVO====  |
├     T: lista Tudo    |
├     Z: Apaga tudo    |
├     G: Ativa Gravac  |
├     g: Para Gravaca  |
├                      |
├     <Gravando>       |
└──────────────────────┘
```

3.2 Fotos da tela OLED

Figura 3. Tela inicial

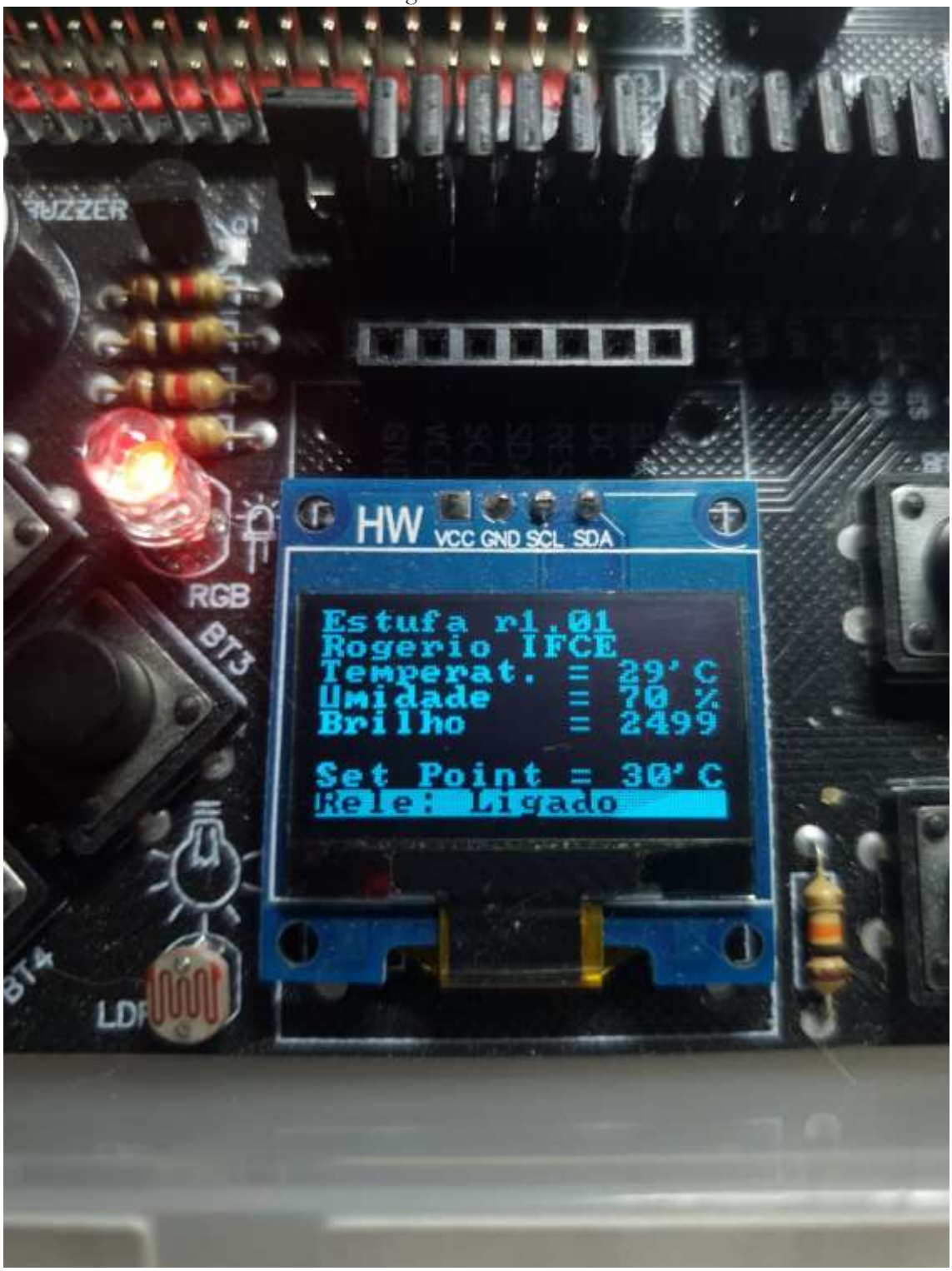


Figura 4. Tela de ajuste do setpoint

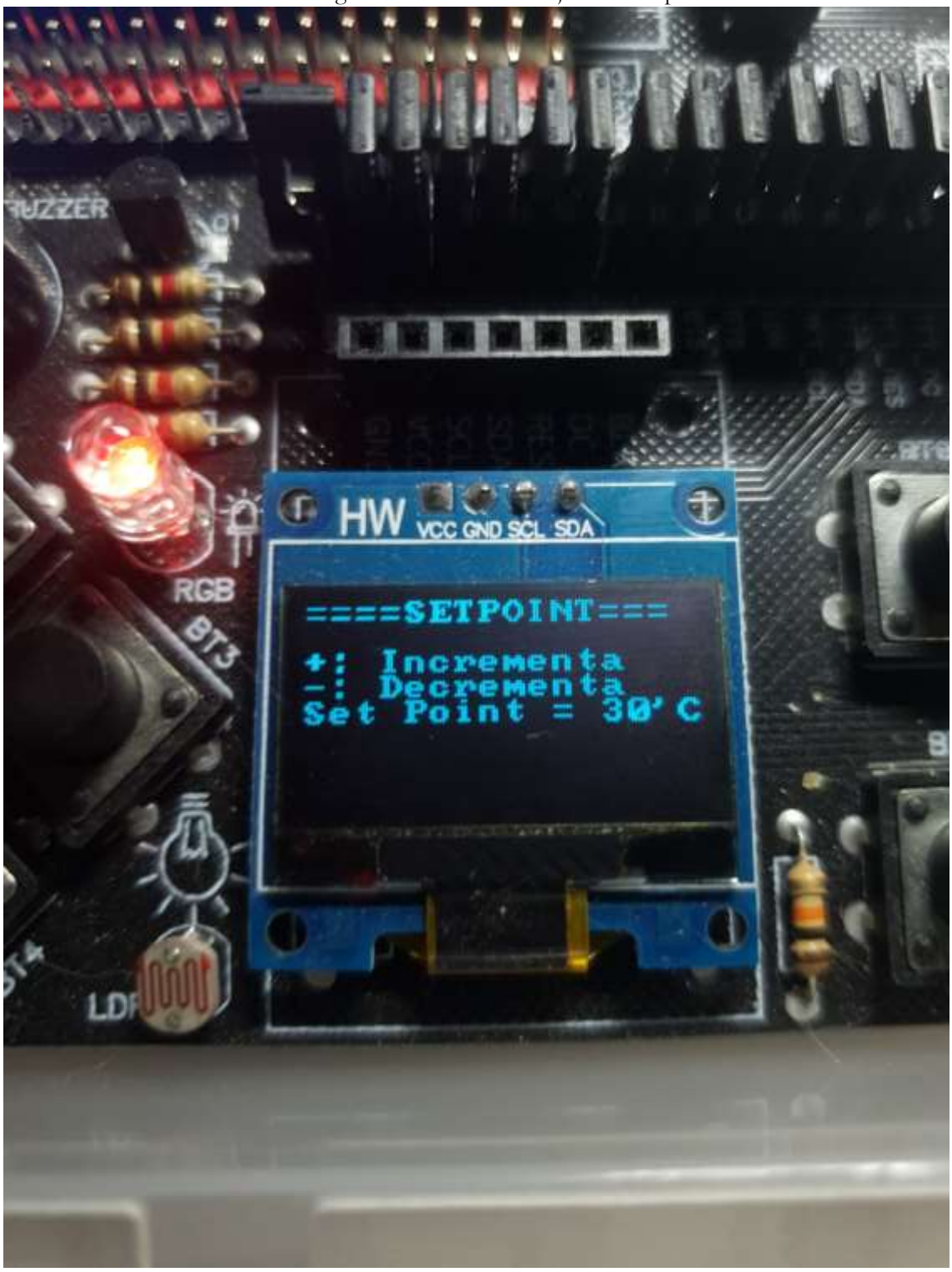


Figura 5. Tela de controle do relé

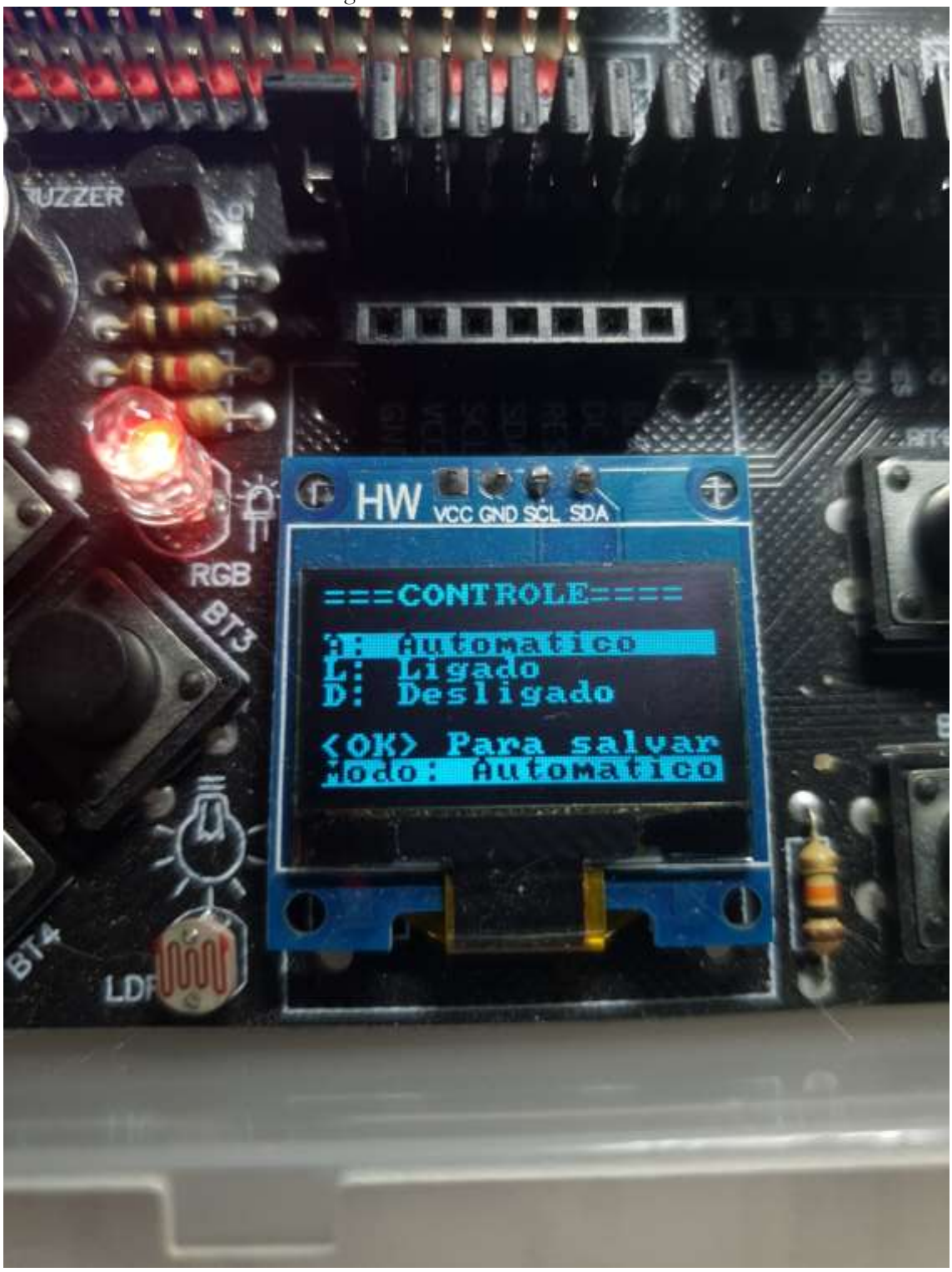
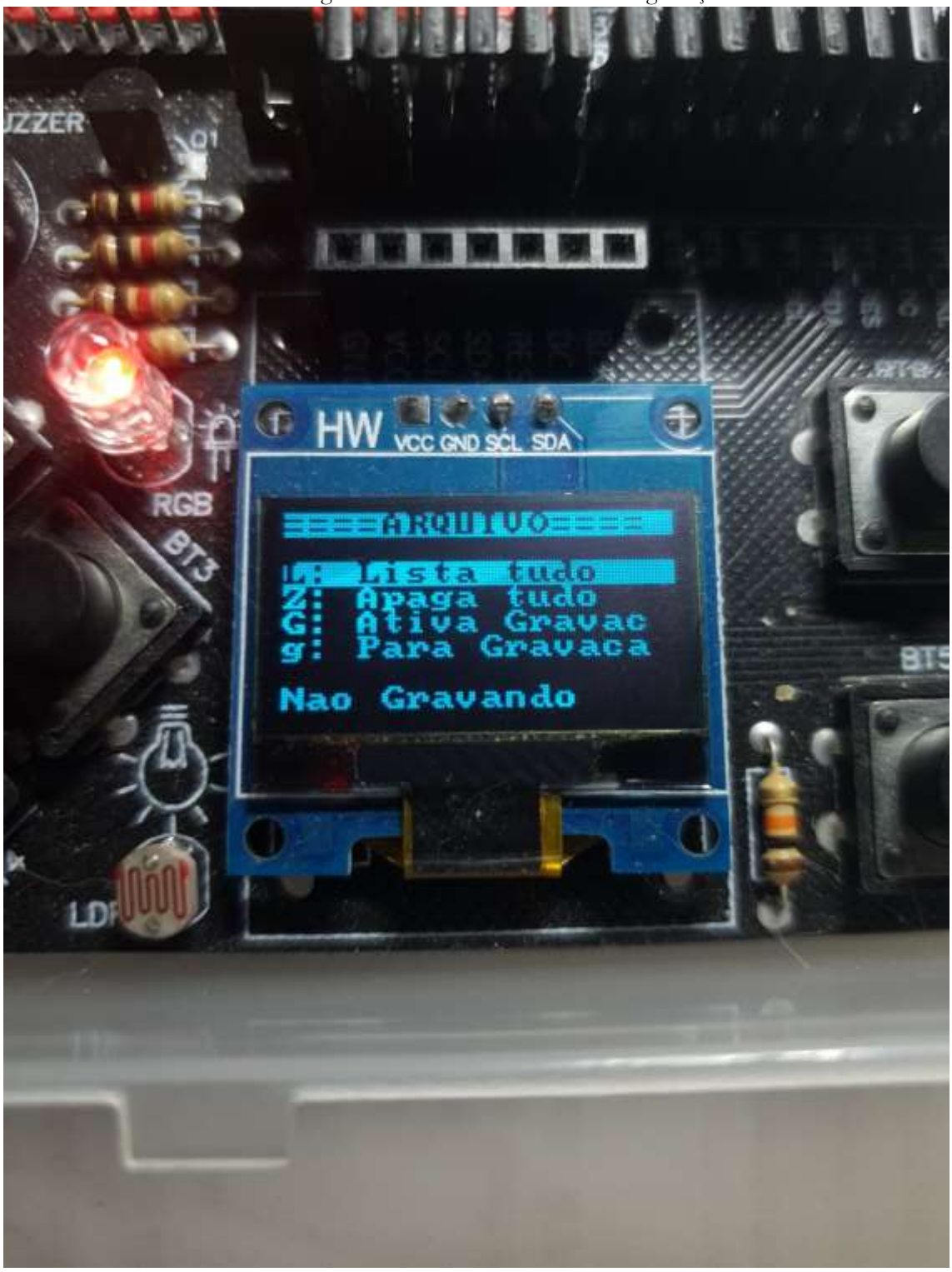


Figura 6. Tela de controle da gravação

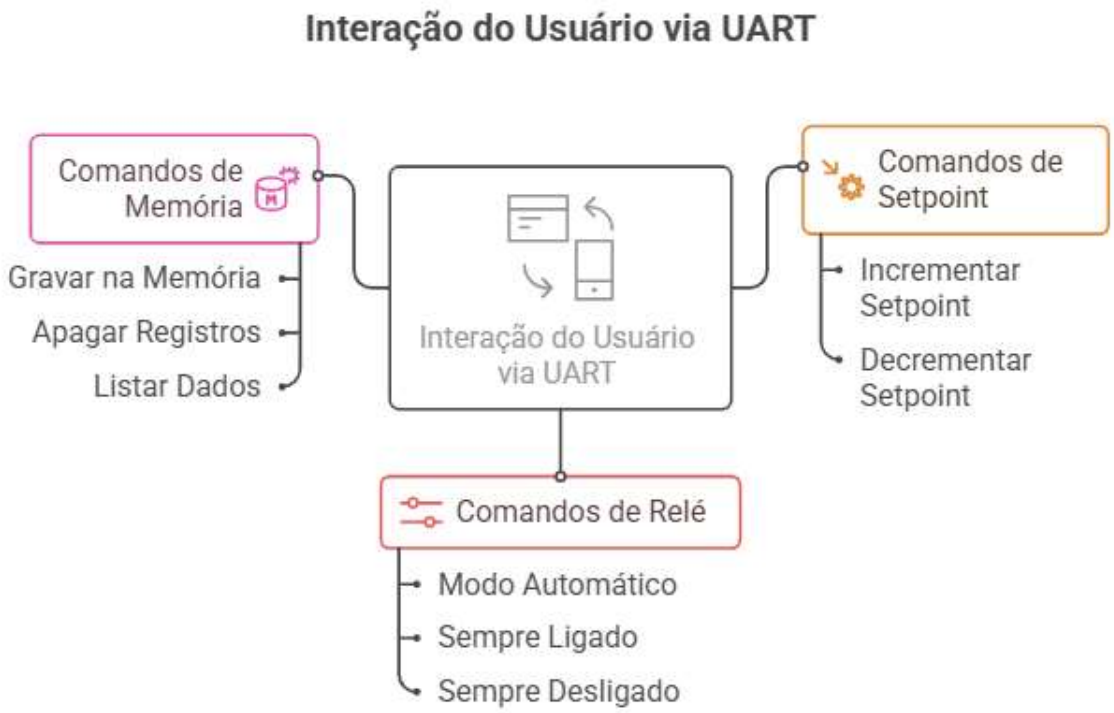


4 Comando pela UART

O usuário pode interagir pelo sistema através da comunicação serial UART a 115200 bps, os comandos são os mesmos acessados pela interface teclado e display, através dos caracteres:

```
'+' --> Incrementa o setpoint
'-' --> Decrementa o setpoint
'a' --> Relé no modo automático
'A' --> Relé no modo automático
'l' --> Relé sempre ligado
'L' --> Relé sempre ligado
'd' --> Relé sempre desligado
'D' --> Relé sempre desligado
'g' --> Para a gravação na memória
'G' --> Ativa a gravação na memória
'Z' --> Apaga todos os registros
'T' --> Lista todos os dados
```

Figura 7. Fluxo das telas



5 Partição de memória

A configuração da memória é:

```
# ESP-IDF Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,         data, nvs,      0x9000,  0x6000,
phy_init,    data, phy,      0xf000,  0x1000,
factory,     app,  factory,  0x10000, 1M,
littlefs,data,spiffs,,256K,,
```

6 Arquivo “Pinos_Franzininho_LAB01.h”

Arquivo com as definições dos pinos da placa LAB01 e Franzininho wifi

```
//Pinos do Franzinhoho Wifi LAB01
#define PINO_LDR                  1    //Pino do LDR, GPIO 34 é um pino
apenas de entrada
#define PINO_BOTAO_ESC   2   //LAB01 GPIO2
#define PINO_BOTAO_OK   3   //LAB01 GPIO3
#define PINO_BOTAO_BAIXO        4   //LAB01 GPIO4
#define PINO_BOTAO_DIREITA      5   //LAB01 GPIO5
#define PINO_BOTAO_ESQUERDA     6   //LAB01 GPIO6
#define PINO_BOTAO_CIMA         7   //LAB01 GPIO7

#define PINO_I2C_SDA            8   //LAB01 SDA
#define PINO_I2C_SCL            9   //LAB01 SCL
#define PINO_OLED_RESET         -1  // Nao implementado no LAB01, GPIO
-1

#define PINO_LED_B              12   //LAB01 LED Azul
#define PINO_LED_G              13   //LAB01 LED Verde
#define PINO_LED_R              14   //LAB01 LED Vermelho
#define PINO_DHT11              15
#define PINO_BUZZER             17   //LAB01 Buzzer
#define PINO_LED_F1             21   //LED Franzininho
#define PINO_LED_F2             33   //LED Franzininho
```

7 Funções do arquivo "Estufa_Rogerio1.c"

É o programa principal. O arquivo implementa o controle de uma estufa automatizada baseada em ESP32, integrando sensores de temperatura, umidade e luminosidade, além de display OLED, relé, botões e armazenamento NVS. Ele gerencia a inicialização dos periféricos, a leitura dos sensores, o processamento do controle do relé conforme o modo de operação (automático ou manual), a navegação por menus no display, o registro e recuperação de dados na memória não volátil, e o tratamento de comandos recebidos via botões ou porta serial. O sistema permite ajustar parâmetros como setpoint de temperatura, modo de controle e ativar/desativar gravação de dados, proporcionando uma interface interativa e flexível para o usuário.

7.1 1. void Display_OLED_Inicia(int TEspera)

```
void Display_OLED_Inicia(int TEspera)
```

Descrição:

- Inicializa o display OLED SSD1306, configura contraste, limpa a tela e exibe textos iniciais.

Parâmetros:

- TEspera: Tempo de espera (em segundos) antes de limpar a tela.

7.2 Termostato_Processa

```
void Termostato_Processa(void)
```

Descrição:

- Processa o controle do relé conforme o modo de operação (automático ou manual) e a temperatura lida.

Parâmetros:

- Nenhum.

7.3 IRAM_ATTR

```
static void IRAM_ATTR gpio_isr_handler(void* arg)
```

Descrição:

- Tratador de interrupção para eventos nos GPIOs dos botões, envia o número do pino para a fila de eventos.

Parâmetros:

- arg: Ponteiro para o número do pino (GPIO).

7.4 Tarefa_Botao

```
void Tarefa_Botao(void *pvpameters)
```

Descrição:

- Tarefa que aguarda eventos de botões, processa pressionamentos e atualiza a tela OLED.

Parâmetros:

- pvpameters: Parâmetro da tarefa (não utilizado).

7.5 void Inicia_os_GPIO(void)

```
void Inicia_os_GPIO(void)
```

Descrição:

- Configura os pinos dos botões, LEDs, buzzer e relé, além de instalar o serviço de interrupção dos GPIOs.

Parâmetros:

- Nenhum.

7.6 void Tela_OLED_Escreve(void)

```
void Tela_OLED_Escreve(void)
```

Descrição:

- Escreve as informações atuais (temperatura, umidade, brilho, data/hora, setpoint, estado do relé) na tela OLED, conforme o menu ativo.

Parâmetros:

- Nenhum.

7.7 void Processa_Botoes_Teclado(uint8_t Botao_Pressionado)

```
void Processa_Botoes_Teclado(uint8_t Botao_Pressionado)
```

Descrição:

- Processa a navegação e ações dos botões no menu, alterando telas, setpoint, modo de controle e opções de gravação.

Parâmetros:

- Botao_Pressionado: Número do pino do botão pressionado.

7.8 void app_main(void)

```
void app_main(void)
```

Descrição:

- Função principal do sistema: inicializa componentes, executa leituras dos sensores, atualiza display, processa controle do relé, grava dados e processa comandos pela porta serial.

Parâmetros:

- Nenhum.

8 Funções do arquivo “RMemoria_NVS.c”

O arquivo "RMemoria_NVS.c" implementa a lógica de armazenamento não volátil (NVS) para a estufa automatizada, permitindo salvar, recuperar, listar e apagar dados importantes do sistema, como setpoint de temperatura, modo de controle, SSID, senha e registros de datalogger. Ele garante que as configurações e informações do usuário sejam preservadas entre reinicializações, facilitando a persistência e a confiabilidade do funcionamento do equipamento. O código também realiza a inicialização da memória NVS.

8.1 void ESTUFA_NVS_Inicializar(void)

```
void ESTUFA_NVS_Inicializar(void)
```

Parâmetros:

- Nenhum

Descrição:

- Inicializa a partição padrão da NVS. Se a partição estiver truncada ou com versão incompatível, apaga e reinicializa.

8.2 void ESTUFA_NVS_Setpoint_Le(int8_t *SetPoint_Memoria)

```
void ESTUFA_NVS_Setpoint_Le(int8_t *SetPoint_Memoria)
```

Parâmetros:

- SetPoint_Memoria: ponteiro para armazenar o valor lido do setpoint.

Descrição:

- Lê o valor do setpoint armazenado na NVS. Se não existir, inicializa com valor padrão e grava na NVS.

8.3 void ESTUFA_NVS_Setpoint_Grava(int8_t SetPoint_Memoria)

```
void ESTUFA_NVS_Setpoint_Grava(int8_t SetPoint_Memoria)
```

Parâmetros:

- SetPoint_Memoria: valor do setpoint a ser gravado.

Descrição:

- Grava o valor do setpoint na NVS.

8.4 void NVS_Le_SSID(void)

```
void NVS_Le_SSID(void)
```

Parâmetros:

- Nenhum

Descrição:

- Lê o SSID armazenado na NVS. Se não existir, inicializa com valor padrão e grava na NVS.

8.5 void NVS_Le_Senha(void)

```
void NVS_Le_Senha(void)
```

Parâmetros:

- Nenhum

Descrição:

- Lê a senha armazenada na NVS. Se não existir, inicializa com valor padrão e grava na NVS.

8.6 void ESTUFA_NVS_Controle_Le(int8_t *Modo)

```
void ESTUFA_NVS_Controle_Le(int8_t *Modo)
```

Parâmetros:

- Modo: ponteiro para armazenar o valor lido do modo de controle.

Descrição:

- Lê o modo de controle armazenado na NVS. Se não existir, inicializa com valor padrão e grava na NVS.

8.7 void ESTUFA_NVS_Controle_Grava(int8_t Modo)

```
void ESTUFA_NVS_Controle_Grava(int8_t Modo)
```

Parâmetros:

- Modo: valor do modo de controle a ser gravado.

Descrição:

- Grava o modo de controle na NVS.

8.8 void NVS_Lista_Entradas(void)

```
void NVS_Lista_Entradas(void)
```

Parâmetros:

- Nenhum

Descrição:

- Lista todas as entradas armazenadas na NVS, mostrando chave e tipo de cada entrada.

8.9 void NVS_Apaga_Tudo(void)

```
void NVS_Apaga_Tudo(void)
```

Parâmetros:

- Nenhum

Descrição:

- Apaga todos os dados da partição NVS.

8.10 void NVS_Datalogger_Grava_Linha(const char* Linha)

```
void NVS_Datalogger_Grava_Linha(const char* Linha)
```

Parâmetros:

- Linha: string com os dados a serem gravados.

Descrição:

- Grava uma linha de dados (registro) na NVS sob a chave definida.

8.11 void NVS_Datalogger_Lista_Tudo(void)

```
void NVS_Datalogger_Lista_Tudo(void)
```

Parâmetros: Nenhum

Descrição:

- Lê e exibe todos os dados registrados na NVS sob a chave de registro.

9 Funções do arquivo “Componente_Rele.c”

O arquivo "Componente_Rele.c" implementa a lógica de controle do relé para automação da estufa, utilizando o ESP-IDF. Ele realiza a configuração do pino do relé, define o estado inicial e permite o acionamento ou desligamento do relé conforme comandos do programa principal.

9.1 Rele_Inicia

Protótipo:

```
void Rele_Inicia(Tipo_Rele *Rele, int Pino)
```

Descrição:

- Inicializa o componente relé, configurando o pino GPIO e definindo o estado inicial como desligado (0).

Parâmetros:

- Rele: ponteiro para a estrutura do relé.
- Pino: número do pino GPIO a ser utilizado.

Ações:

- Define o pino e o estado na estrutura.
- Configura o pino como saída.
- Define o nível do pino conforme o estado inicial.

9.2 Rele_Liga

Protótipo:

```
void Rele_Liga(Tipo_Rele *Rele)
```

Descrição:

- Liga o relé, alterando o estado para 1.

Parâmetros:

- Rele: ponteiro para a estrutura do relé.

Ações:

- Atualiza o estado para ligado.
- Define o nível do pino como alto.

9.3 Rele_Desliga

Protótipo:

```
void Rele_Desliga(Tipo_Rele *Rele)
```

Descrição:

- Desliga o relé, alterando o estado para 0.

Parâmetros:

- Rele: ponteiro para a estrutura do relé.

Ações:

- Atualiza o estado para desligado.
- Define o nível do pino como baixo.

9.4 Rele_Estado

Protótipo:

```
int Rele_Estado(Tipo_Rele *Rele)
```

Descrição:

- Retorna o estado atual do relé (1 para ligado, 0 para desligado).

Parâmetros:

- Rele: ponteiro para a estrutura do relé.

Retorno:

- Estado atual do relé.

9.5 Rele_Define_Estado

Protótipo:

```
void Rele_Define_Estado(Tipo_Rele *Rele, int Estado)
```

Descrição:

- Define o estado do relé (1 para ligado, 0 para desligado) de forma explícita.

Parâmetros:

- Rele: ponteiro para a estrutura do relé.

Estado: valor desejado para o estado (qualquer valor diferente de zero será considerado ligado).

Ações:

- Atualiza o estado conforme o parâmetro.
- Define o nível do pino conforme o novo estado.

10 Funções do arquivo “Componente_LAB01_Sensores.c”

O arquivo "Componente_LAB01_Sensores.c" implementa a integração dos sensores de temperatura, umidade (DHT11) e luminosidade (LDR) para a placa LAB01, utilizando o ESP-IDF. Ele realiza a configuração e inicialização dos periféricos necessários, como ADC para o LDR

10.1 LAB01_DHT11_Leitura

Protótipo:

```
uint8_t  LAB01_DHT11_Leitura(int  Pino_DHT11,  int16_t  *Temperatura,
int16_t *Umidade)
```

Descrição:

- Realiza a leitura do sensor DHT11 conectado ao pino especificado. Retorna 1 se a leitura foi bem-sucedida e 0 em caso de erro. Os valores de temperatura e umidade são armazenados nos ponteiros fornecidos.

Parâmetros:

- Pino_DHT11: Pino onde o DHT11 está conectado.
- Temperatura: Ponteiro para armazenar o valor lido da temperatura.
- Umidade: Ponteiro para armazenar o valor lido da umidade.

10.2 LAB01_LDR_Inicializa

Protótipo:

```
void LAB01_LDR_Inicializa(void)
```

Descrição:

- Inicializa o ADC para leitura do sensor LDR da placa LAB01, configurando unidade, canal, resolução e atenuação.

Parâmetros:

- Nenhum.

10.3 LAB01_LDR_Ler

Protótipo:

```
void LAB01_LDR_Ler(int16_t *Tensao_Lida)
```

Descrição:

- Realiza a leitura do valor analógico do LDR via ADC, converte para tensão em mV e armazena no ponteiro fornecido.

Parâmetros:

- Tensao_Lida: Ponteiro para armazenar o valor da tensão lida (em mV).