Aplicação Datalogger Multifuncional

Nome: **José Adriano Filho**
Matrícula: **2025101109806**
Unidade_5 - Capítulo_2

Seu desafio é criar um datalogger multifuncional utilizando a plataforma Labrador 32, que seja capaz de monitorar uma ou mais variáveis de interesse definidas por você e seus colegas, com base nos sensores disponíveis em laboratório. O sistema deve registrar periodicamente os dados coletados em um arquivo de texto no cartão microSD, que será usado como armazenamento.

O sistema deverá ser capaz de:

• Coletar dados dos sensores selecionados.

• Registrar as leituras em intervalos de tempo definidos.

• Armazenar as leituras em arquivos .txt no cartão microSD, incluindo o timestamp de cada leitura.

• Garantir a escrita de forma eficiente, sem perda de dados.

Resolução:

Link: https://github.com/EngAdriano/Residencia/tree/main/Exercicios/U5C1Tarefas/pico_datalog

Conforme solicitação foi desenvolvido o projeto de um datalogger baseado para armazenar o histórico do sensor AHT10, sensor de temperatura e umidade de alta precisão I2C, armazenando os dados em um arquivo denominado “LOG_ENV.TXT” em um cartão microSD.

Abaixo temos o detalhamento do código:

pico_datalog.c > WIFI_SSID

```c
/* ------------------------------------------------------------------------------------------------------------
/ Projeto: Datalog AHT10 (Sensor de Temperatura e Umidade)
/ Descrição: Este código lê a temperatura e umidade do sensor AHT10 e exibe os dados em um display OLED SSD1306,
/ Salvando um histórico em um cartão microSD.
/ Bibliotecas: aht10, ssd1306, FatFs_SPI
/ Autor: José Adriano
/ Data de Criação: 16/10/2025
/------------------------------------------------------------------------------------------------------------
*/
// main.c - Datalogger AHT10 -> SD card (log a cada 60s), OLED em tempo real
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include "hardware/rtc.h"
#include "pico/cyw43_arch.h"

#include "aht10.h"
#include "ssd1306.h"
#include "f_util.h"
#include "ff.h"
#include "sd_card.h"
#include "hw_config.h"
```

Fig. 1 – Identificação e includes necessários

Como vamos trabalhar com gravação em FAT32, estamos utilizando a biblioteca FatFs_SPI para gerenciar o acesso ao cartão SD, bem como a gravação, leitura dos dados no arquivo. As bibliotecas aht10 e ssd1306 são utilizadas para manipular o sensor e o OLED respectivamente.

```
28  #define WIFI_SSID "xxxxxxxxxx"
29  #define WIFI_PASSWORD "xxxxxxxxxx"
30
31  // I2C: AHT10 -> i2c0 (GPIO0/GPIO1), OLED -> i2c1 (GPIO14/GPIO15)
32  #define I2C_PORT0 i2c0
33  #define I2C_SDA0 0
34  #define I2C_SCL0 1
35
36  #define I2C_PORT1 i2c1
37  #define I2C_SDA1 14
38  #define I2C_SCL1 15
39
40  #define LOG_FILENAME "LOG_ENV.TXT"
41
42  // Protótipos
43  int i2c_write(uint8_t addr, const uint8_t *data, uint16_t len);
44  int i2c_read(uint8_t addr, uint8_t *data, uint16_t len);
45  void delay_ms(uint32_t ms);
46  void log_data(float temp, float hum);
47  void display_values(float temp, float hum, bool show_recording);
48  void display_alert(const char *msg);
49  bool init_wifi_and_print_ip(void);
50  bool get_timestamp_string(char *buf, size_t len);
51  void set_initial_time(void);
```

Fig. 2 – Protótipo das funções utilizadas

Acima temos os protótipos das funções para facilitar o uso das bibliotecas, assim não necessitamos ficar reescrevendo códigos várias vezes.

```
55  // ======================== main ============================
56  int main(void) {
57      stdio_init_all();
58      sleep_ms(200);
59
60      // Inicializa hora inicial
61      set_initial_time();
62
63      // I2C sensor (I2C0)
64      i2c_init(I2C_PORT0, 100 * 1000); // 100 kHz
65      gpio_set_function(I2C_SDA0, GPIO_FUNC_I2C);
66      gpio_set_function(I2C_SCL0, GPIO_FUNC_I2C);
67      gpio_pull_up(I2C_SDA0);
68      gpio_pull_up(I2C_SCL0);
69
70      // I2C OLED (I2C1)
71      i2c_init(I2C_PORT1, 400 * 1000); // 400 kHz
72      gpio_set_function(I2C_SDA1, GPIO_FUNC_I2C);
73      gpio_set_function(I2C_SCL1, GPIO_FUNC_I2C);
74      gpio_pull_up(I2C_SDA1);
75      gpio_pull_up(I2C_SCL1);
76
77      // Opcional: inicializa Wi-Fi (se desejar)
78      init_wifi_and_print_ip();
```

Fig. 3 – Inicialização das portas I2C do sensor e Oled

Na figura 3 temos a inicialização das portas seriais que serão utilzadas para comuniocação com o sensor aht10 e o display Oled, bem como uma conexão com o WiFi para implementação de envio dos dados para nuvem em implementações futuras.

```
80      // Inicializa OLED
81      ssd1306_init(I2C_PORT1);
82      ssd1306_clear();
83      ssd1306_draw_string(12, 10, "Inicializando...");
84      ssd1306_show();
85      sleep_ms(400);
86
87      // Configura handle AHT10
88      AHT10_Handle aht10 = {
89          .iface = {
90              .i2c_write = i2c_write,
91              .i2c_read  = i2c_read,
92              .delay_ms  = delay_ms
93          }
94      };
95
96      printf("Inicializando AHT10...\n");
97      if (!AHT10_Init(&aht10)) {
98          printf("Falha na inicialização do AHT10\n");
99          ssd1306_clear();
100         ssd1306_draw_string(10, 20, "Falha AHT10");
101         ssd1306_show();
102         while (1) sleep_ms(1000);
103     }
104     sleep_ms(50); // estabiliza sensor
```

Fig. 4 – Exibição de cabeçalho inicial no Oled e conexão das funções para operação com o AHT10

Na figura 4 exibimos as mensagens no Oled, bem como registramos as funções para manipulação da comunicação via I2C com o sensor de temperatura e umidade. Este registro se faz necessário pois ao criarmos a biblioteca utilizamos apenas comandos padrões do C, assim ela pode ser levada pára outra plataforma sem alteração alguma, bastando apenas reescrever o que chamamos de driver composto das funções no headle do AHT10.

```
106     // Cria cabeçalho do arquivo se necessário
107     sd_card_t *pSD = sd_get_by_num(0);
108     if (pSD) {
109         FRESULT fr = f_mount(&pSD->fatfs, pSD->pcName, 1);
110         if (fr == FR_OK) {
111             FIL fil;
112             fr = f_open(&fil, LOG_FILENAME, FA_OPEN_APPEND | FA_WRITE);
113             if (fr == FR_OK) {
114                 FSIZE_t size = f_size(&fil);
115                 if (size == 0) {
116                     f_printf(&fil, "DataHora;Temperatura(C);Umidade(%%)\n");
117                 }
118                 f_close(&fil);
119             }
120             f_unmount(pSD->pcName);
121         }
122     }
```

Fig. 5 – Cria o cabeçalho do arquivo, se necessário

Na figura 5, testamos se o arquivo “LOG_ENV.TXT” existe, caso não criamos e colocamos um cabeçalho para identificar os dados presentes nele.

```
124     // Mostra mensagem inicial
125     ssd1306_clear();
126     ssd1306_draw_string(8, 10, "Datalogger pronto");
127     ssd1306_draw_string(6, 30, "OLED: em tempo real");
128     ssd1306_show();
129     sleep_ms(800);
130
131     // --- Agendamento ---
132     const uint32_t display_interval_ms = 2000; // atualiza display a cada 2s
133     const uint32_t log_interval_ms = 60 * 1000; // grava no SD a cada 60s
134
135     absolute_time_t next_display_time = make_timeout_time_ms(500);
136     absolute_time_t next_log_time = make_timeout_time_ms(1000);
137
138     float last_temp = 0.0f;
139     float last_hum  = 0.0f;
```

Fig. 6 – Criação das variáveis de controle para intervalo de gravação e de refresh do Oled

Na figura 6 informamos via Oled que o sistema está pronto para iniciar os trabalhos bem como criamos as variáveis para controle do tempo de exibição das informações no lcd Oled e o intervalo de gravação das informações no arquivo de log.

```
141     while (true) {
142         absolute_time_t now = get_absolute_time();
143
144         // --- Atualização de display ---
145         if (absolute_time_diff_us(now, next_display_time) >= 0) {
146             if (AHT10_ReadTemperatureHumidity(&aht10, &last_temp, &last_hum)) {
147                 display_values(last_temp, last_hum, false);
148             } else {
149                 display_alert("Erro leitura AHT10");
150             }
151             next_display_time = delayed_by_ms(next_display_time, display_interval_ms);
152         }
153
154         // --- Gravação no SD ---
155         if (absolute_time_diff_us(now, next_log_time) >= 0) {
156             log_data(last_temp, last_hum);
157             display_values(last_temp, last_hum, true);
158             sleep_ms(1000);
159             display_values(last_temp, last_hum, false);
160             next_log_time = delayed_by_ms(next_log_time, log_interval_ms);
161         }
162
163         sleep_ms(50);
164     }
165
166     return 0;
167 }
```

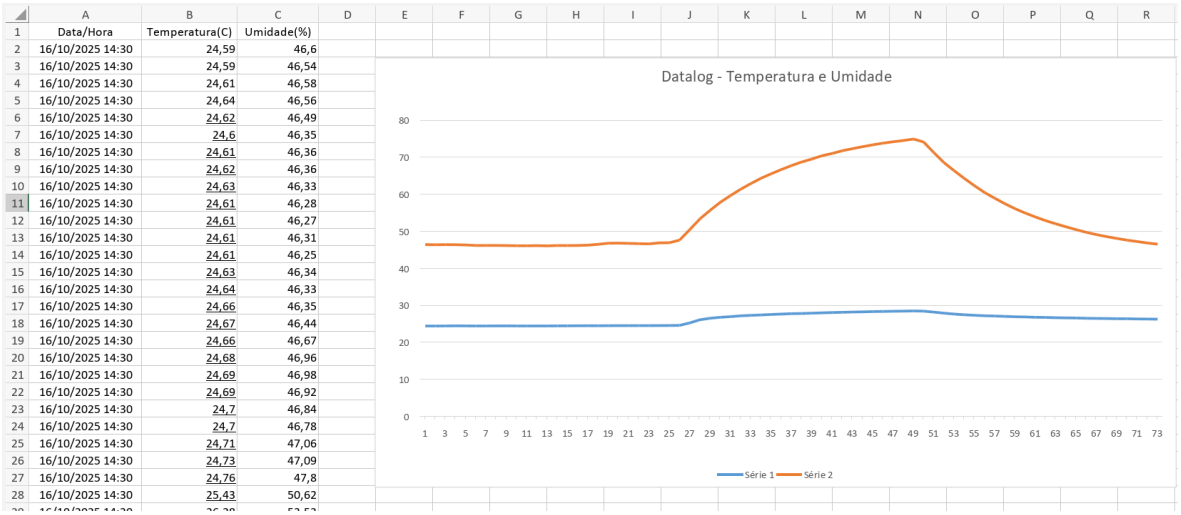
Fig. 7 – Loop infinito

Na figura 7 temos nosso loop infinito, afinal de contas é uma característica padrão em sistemas embarcados, nele temos as tarefas que são executadas periodicamente, que são atualizar o display com a leitura dos dados do sensor AHT10 e a gravação dos mesmos no arquivo de log.

Conclusão

Durante o desenvolvimento do projeto, tivemos a oportunidade de engrandecer nosso conhecimento como por exemplo, a utilização de uma biblioteca para manipulação de arquivos, informação importante para mim já que vou utiliazar um sistema datalogger em meu projeto final.

Abaixo temos uma figura que mostra a utilização dos dados para a geração de um gráfico



	Data/Hora	Temperatura(C)	Umidade(%)
2	16/10/2025 14:30	24,59	46,6
3	16/10/2025 14:30	24,59	46,54
4	16/10/2025 14:30	24,61	46,58
5	16/10/2025 14:30	24,64	46,56
6	16/10/2025 14:30	24,62	46,49
7	16/10/2025 14:30	24,6	46,35
8	16/10/2025 14:30	24,61	46,36
9	16/10/2025 14:30	24,62	46,36
10	16/10/2025 14:30	24,63	46,33
11	16/10/2025 14:30	24,61	46,28
12	16/10/2025 14:30	24,61	46,27
13	16/10/2025 14:30	24,61	46,31
14	16/10/2025 14:30	24,61	46,25
15	16/10/2025 14:30	24,63	46,34
16	16/10/2025 14:30	24,64	46,33
17	16/10/2025 14:30	24,66	46,35
18	16/10/2025 14:30	24,67	46,44
19	16/10/2025 14:30	24,66	46,67
20	16/10/2025 14:30	24,68	46,96
21	16/10/2025 14:30	24,69	46,98
22	16/10/2025 14:30	24,69	46,92
23	16/10/2025 14:30	24,7	46,84
24	16/10/2025 14:30	24,7	46,78
25	16/10/2025 14:30	24,71	47,06
26	16/10/2025 14:30	24,73	47,09
27	16/10/2025 14:30	24,76	47,8
28	16/10/2025 14:30	25,43	50,62

Fig. 8 – Processamento dos dados, azul: temperatura, laranja: umidade

Na figura 8 temos um processamento dos dados a partir do arquivo de log, a linha azul temos a temperatura e a linha laranja temos a umidade, como podemos perceber temos uma variação bem acentuada da umidade, este fenômeno se deu pela imposição do dedo no sensor provando de forma intencional para mostrar a importância do acompanhamento do fenômeno físico em forma de gráfico. Projeto funcionando.

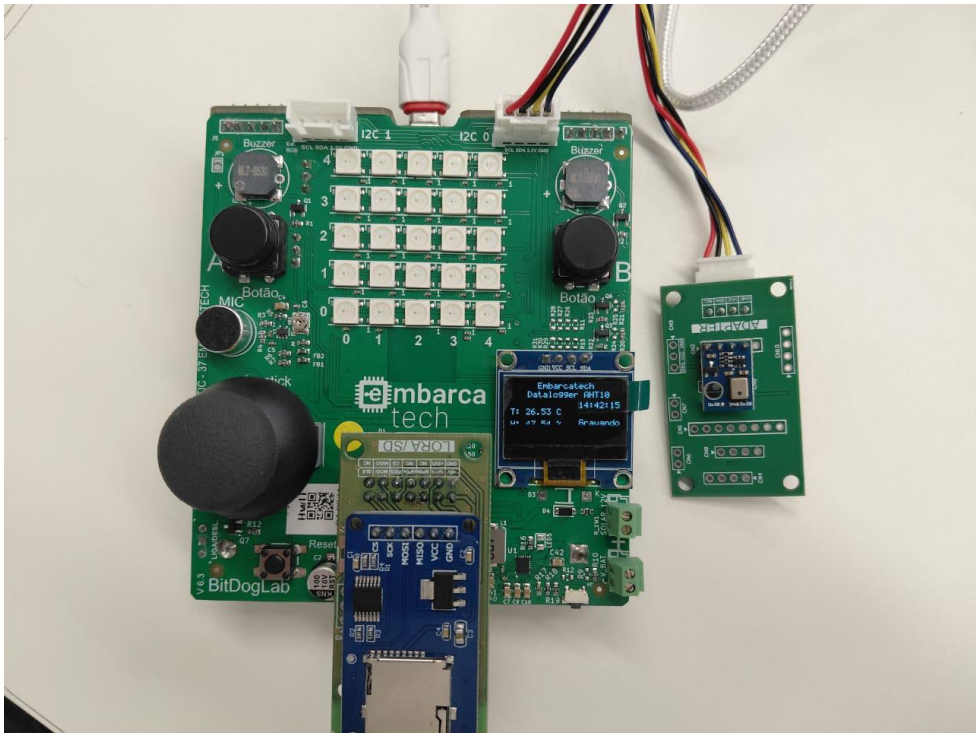


Fig. 9 – Hardware utilizado