TC74 Temperature Sensor with a fan

Project presentation – ASE

João Torrinhas – 98435

Diogo Torrinhas - 98440

6 June 2023

Components

- ESP32.
- Sensor de temperatura TC74.
- Ventoinha/LED.
- Transistor VN2222(before).
- Díodo 1N4007(before).

TC74 Sensor

- O TC74 é um sensor que comunica com outros dispositivos por meio do protocolo de comunicação I2C.
- O sensor TC74 pode medir temperaturas na faixa dos -40ºC a +125ºC, tornando-o adequado para uma variadade de aplicações.
- Este sensor é capaz de medir a temperatura com alta precisão, geralmente com uma resolução de 0,5ºC.

Project Objectives

- Ler a temperatura: utilizar o sensor de temperatura TC74 para medir com precisão a temperatura ambiente.
- Controlo do brilho do LED baseado na temperatura: baseado na temperatura lida no sensor atualizar duty cycle do LED controlando assim o brilho.
- Dashboard: criar uma dashboard que mostre os valores da temperatura e os valores do duty cycle a cada segundo.
- Controlo remoto por terminal e sistema de ficheiros.

Connecting the sensor

- Para a leitura da temperatura conectamos o pino VCC do sensor com o pino 3.3V do ESP32.
- Em seguida, conectamos o pino SDA e o pino SCLK do sensor ao GPIO19 e ao GPIO 18 do ESP32, respetivamente.
- Por fim, foi feita a ligação do pino de ground do sensor ao ground do ESP32.

Project Overview

- Leitura dos valores de temperatura e enviá-los para a dashboard.
- Alteração da velocidade da ventoinha (luminosidade do LED) de acordo com os valores recebidos de temperatura (dinamicamente) e enviar os valores do duty-cycle para a dashboard.
- Alteração da velocidade da ventoinha (luminosidade do LED) manualmente, ou seja, turn on/off e enviar os valores do duty-cycle para a dashboard.
- Guardar/eliminar/ver logs, dos valores de temperatura recebidos.
- O modo manual, dinamico e o sistema de logs é ativado usando o terminal. Pressionar a tecla 'm' para o modo manual, 'n' para o modo dinâmico (normal), 'r' para ver os logs, 'e' para eliminar logs existentes. Assumindo que o modo manual está ativado, pressionar 'l' para ligar a ventoinha (LED) no máximo e 'd' para desligar a ventoinha (LED).

Reading Temperatures

- Foram desenvolvidas um conjunto de funções para lermos os valores de temperatura do sensor.
- Em primeiro lugar, é inicializado o sensor e, em seguida, o mesmo é acordado.
- Após o sensor ter sido acordado, ficamos à espera que as temperaturas estivessem prontas para serem lidas do sensor. Quando estivessem prontas, foi feita a leitura das mesmas.
- Após a aquisição dos valores de temperatura, o sensor é colocado em standby e só é acordado quando for necessário adquirir novos valores.

```c
#pragma once
#include "driver/i2c.h"

esp_err_t tc74_init(i2c_port_t i2cPort, int sdaPin, int sclPin, uint32_t clkSpeedHz);

esp_err_t tc_74_free(i2c_port_t i2cPort);

esp_err_t tc74_standby(i2c_port_t i2cPort, uint8_t sensAddr, TickType_t timeOut);

esp_err_t tc74_wakeup(i2c_port_t i2cPort, uint8_t sensAddr, TickType_t timeOut);

bool tc74_is_temperature_ready(i2c_port_t i2cPort, uint8_t sensAddr, TickType_t timeOut);

esp_err_t tc74_wakeup_and_read_temp(i2c_port_t i2cPort, uint8_t sensAddr,
                                    TickType_t timeOut, uint8_t* pData);

esp_err_t tc74_read_temp_after_cfg(i2c_port_t i2cPort, uint8_t sensAddr,
                                   TickType_t timeOut, uint8_t* pData);

esp_err_t tc74_read_temp_after_temp(i2c_port_t i2cPort, uint8_t sensAddr,
                                    TickType_t timeOut, uint8_t* pData);
```

File System

- Para guardar os valores da temperatura num ficheiro, foi usado o ESP File System(SPIFFS), que é um sistema de arquivos baseado em flash, onde os arquivos são armazenados em uma memória flash SPI, armazenada no chip.
- Através do SPIFFS, nós conseguimos guardar, eliminar e ler os logs dos valores de temperatura.

Tasks

Foram criadas tasks para cada ação no sistema embutido:

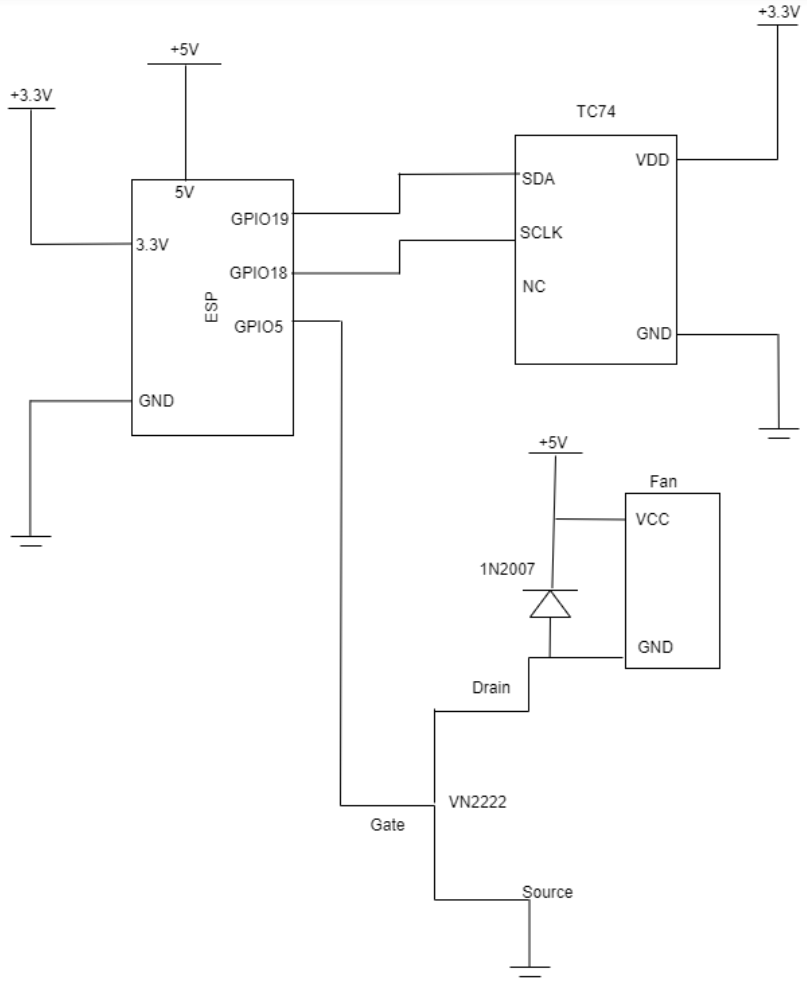
- Task para executar ações através da consola do terminal (ver logs, modo manual...)
- Task para efetuar a leitura de valores de temperatura do sensor.
- Task para atualizar o duty-cycle da ventoinha (LED) dinamicamente.
- Tasks para ver/eliminar logs.

```
xTaskCreate(console_task, "console_task", 2048, NULL, 5, NULL);
xTaskCreate(read_temperature_task, "Read Temperature", 2048, NULL, 5, NULL);
xTaskCreate(ledc_update_task, "LEDC Update", 2048, NULL, 5, NULL);
xTaskCreate(show_logs_task, "Show Logs", 2048, NULL, 5, NULL);
xTaskCreate(delete_logs, "Delete Logs", 2048, NULL, 5, NULL);
```

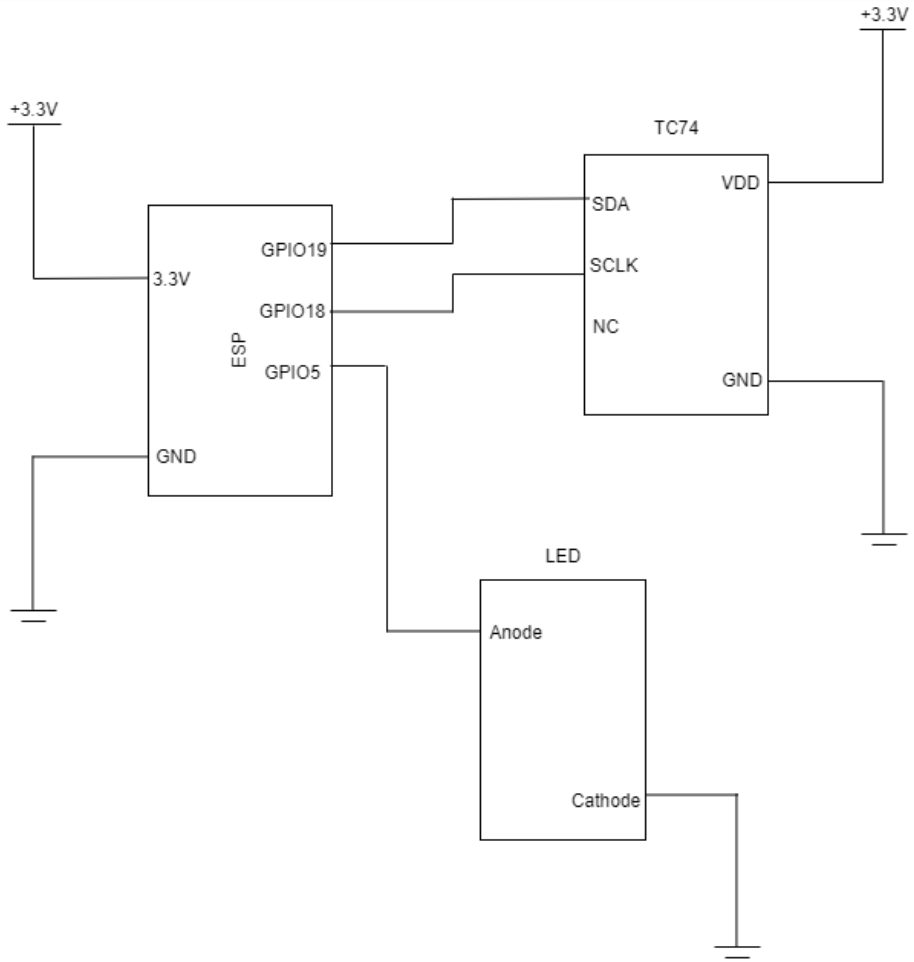
Changing the fan duty cycle

- Para calcular o duty cycle da ventoinha, inicialmente foram definidas a temperatura máxima, onde o duty cycle é 100%, e a temperatura mínima onde o duty cycle é 0%.
- Em seguida, de maneira ao duty cycle ir variando tendo em conta a temperatura, foi usada a expressão (*duty = (tempC – tempMinima)* 255/3*)

General Diagram with a fan



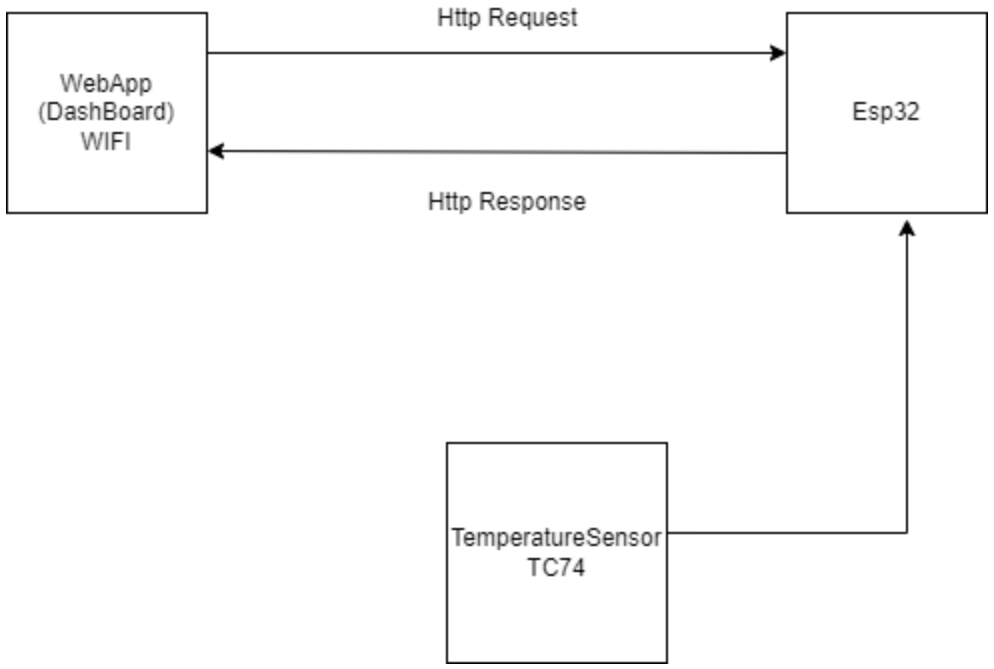
General Diagram with a LED

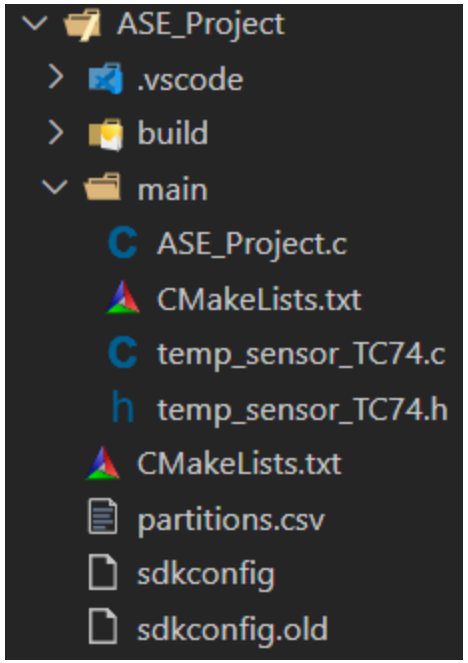


DashBoard

- Para a criação da dashboard usámos a Biblioteca (HTTP Web Server) disponível no esp-idf e o frontend foi desenvolvido utilizando html e CSS. A dashboard mostra os valores da temperatura do sensor e do duty cycle da ventoinha e são atualizados a cada segundo.

Software Architecture





Project Improvements

O que tinhamos antes e que mantivémos:

- Variação da velocidade da ventoinha de acordo com a temperatura lida do sensor, modo dinâmico (Está a funcionar bem).
- Conexão do ESP32 à Dashbord através do wifi.

Melhorias feitas:

- Adicionámos controlo remoto por terminal para escolha dos modos de variação do duty-cycle da ventoinha/LED e para gestão de logs.
- Desenvolvemos um novo código para leitura dos valores de temperatura (sem usar uma biblioteca externa), uma vez que usámos um sensor diferente.
- Utilização de um sistema de ficheiros (spiffs) para armazenar dados localmente (logs).
- Utilização de tasks na estrutura do projeto.

Check list

- Utilizar o ESP32DevKitC como base do embedded system;
- Cada grupo deve selecionar um sensor para ligar ao kit ESP32DevKitC e propor a sua aquisição até 14/mar;
- A aplicação a executar no kit ESP32DevKitC deve ser desenvolvida em C/C++ e tirar partido do FreeRTOS;
- Devem ser explorados os periféricos do ESP32 que fizerem sentido no contexto do projeto, incluindo aspetos de interrupções e DMA;
- Os dados recolhidos do sensor e processados no ESP32 devem ser apresentados num dashboard remoto, sendo para tal necessária conectividade de rede (WiFi / BT);
- Deve ser disponibilizada uma ligação por Terminal; independente do dashboard remoto;
- Devem ser exploradas as várias funcionalidades das ferramentas de desenvolvimento, incluindo debug.
- Podem ser explorados os modos de baixo consumo energético do ESP32;
- Podem ser suportadas atualizações remotas (Over-the-Air) do sistema;
- Pode ser incluído algum tipo de atuador cuja utilização faça sentido com o sensor usado (de forma a criar um loop de controlo; ou que seja controlado através do dashboard);
- Pode ser suportado um sistema de ficheiros para armazenar dados localmente.

Baseline

- A dashboard foi baseada na biblioteca (HTTP Web Server) do esp32, e através do respetivos exemplos do express.
- Código de leitura do sensor desenvolvido por nós baseado nos exemplos da aula e no datasheet do sensor TC74.
- Controlo da ventoinha por pwm foi baseado no que aprendemos na aula acerca de pwm.
- Montagem de todo o circuito na placa branca baseado em tudo o que aprendemos na aula.

Conclusion

- Achamos que este projeto foi muito importante no nosso percurso académico uma vez que, ficámos a entender melhor como ler valores de sensores, trabalhar com PWM, montar circuitos e mexer branca, etc.
- Em suma, nós acreditamos que atingimos com sucesso os objetivos do projeto e, no geral, atingimos com sucesso os objetivos da cadeira.

Contribuição dos autores

- O trabalho desenvolvido ao longo deste projeto foi distribuído de igual forma pelos dois elementos do grupo.
- João Torrinhas – 50%
- Diogo Torrinhas – 50%

Bibliography

- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2c.html