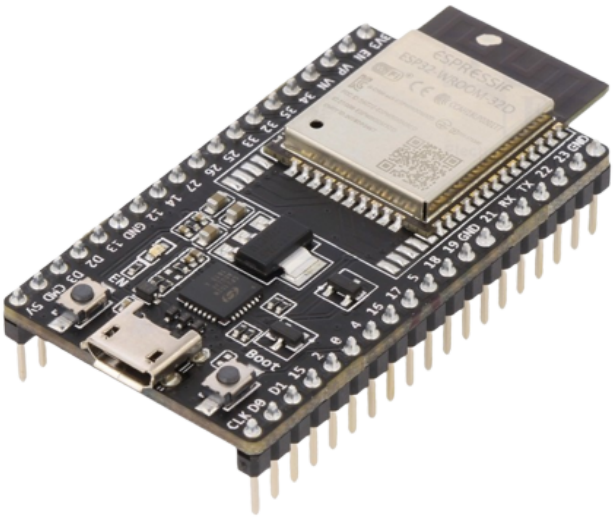
deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

ESPRESSIF ESP32-DEVKITC

Software aspects and ADC demonstration

Part 2



TP1 - Grupo 7
Lúcia Sousa 93086
Raquel Pinto 92948

28/03/2022

ASE - Arquitetura de Sistemas Embutidos

Índice

1. Linguagens e ferramentas de desenvolvimento

Para programar a placa ESP 32 existem inúmeras opções.

Pode-se usar várias linguagens como o C/C++, Python, Micropython, Lua e JavaScript (são as mais utilizadas) e vários IDEs.

Os mais conhecidos como o Visual Studio Code, o Eclipse ou outro similar.

O ESP-IDF (Espressif IoT Development Framework) é uma opção de baixo nível usado para desenvolvimento de software.

O Arduino (opção de nível intermédio) que com uma extensão pode-se adaptar para usar na placa ESP32 usando a linguagem C++.

E como alto nível o MicroPython.

1.1 ESP-IDF

ESP-IDF é o framework oficial de desenvolvimento para a ESP32.

Para começar a utilizar o ESP-IDF no ESP32, será necessário instalar o seguinte software:

- **Toolchain** para compilar código para ESP32;
- **Ferramentas de construção** - CMake e Ninja para construir uma aplicação completa para ESP32;
- **ESP-IDF** que contém essencialmente API (bibliotecas de software e código fonte) para ESP32 e scripts para operar a cadeia de ferramentas.

A instalação do ESP-IDF pode ser feita através de um IDE como o Eclipse ou Visual Studio Code.

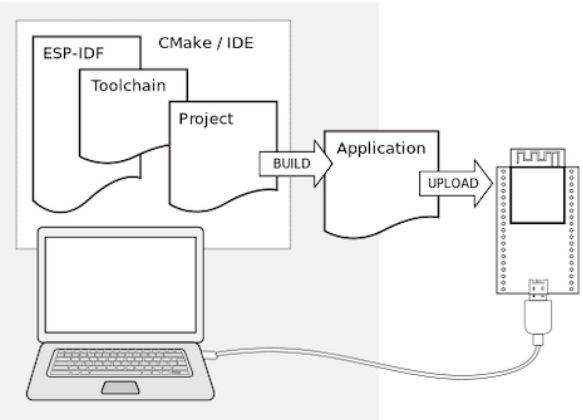


Figura 1 - Passos do ESP-IDF.

2. Demonstração do ADC

2.1 Descrição

Quanto à demonstração do periférico ADC como se pode ver na Figura 2, o ADC recebe uma entrada analógica (voltagem) varia entre 0 e 3.3 volts. O ADC converte essa entrada em números de 0 a 4095 formando um output digital binário de 12 bits.

Para o ADC converter o sinal de entrada tem 3 fases. Primeiro faz uma amostragem do sinal, depois qualifica-o para determinar a sua resolução e finalmente estabelece valores binários e envia esses valores para o sistema ler o sinal digital.

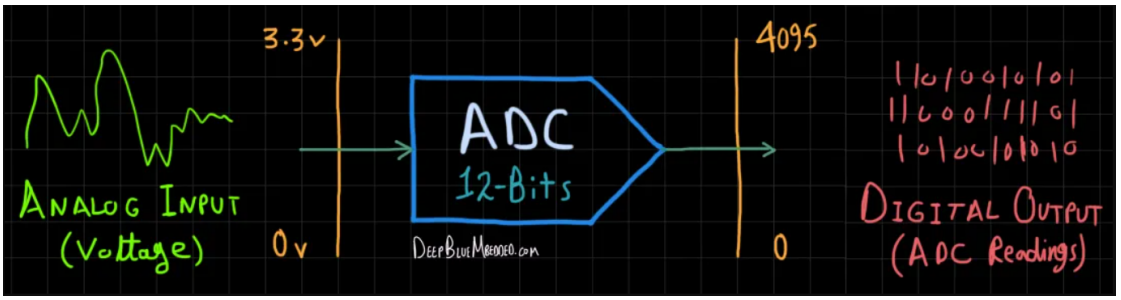


Figura 2 - Funcionamento do periférico ADC.

2.2 Implementação

Foi definido o pino de entrada, VP (GPIO36), para ler o valor analógico, de um potenciómetro.

```
Raw: 1707       Voltage: 472mV
Raw: 1699       Voltage: 470mV
Raw: 1702       Voltage: 470mV
Raw: 1702       Voltage: 470mV
Raw: 3714       Voltage: 938mV
Raw: 4095       Voltage: 1026mV
Raw: 4095       Voltage: 1026mV
```

Figura 3 - Resultados do programa implementado.

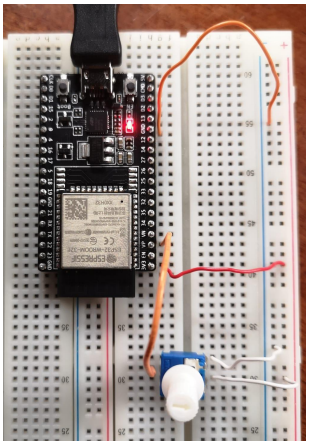


Figura 4 - Circuito.

3. Modos de Energia

Existem vários modos de energia:

- Modo ativo onde todas as componentes do chip estão ativas, gastando-se normalmente entre 160 a 260 mA podendo ficar a 790 mA se se usar o Wi-Fi e o Bluetooth juntos ao mesmo tempo (Figura 5).

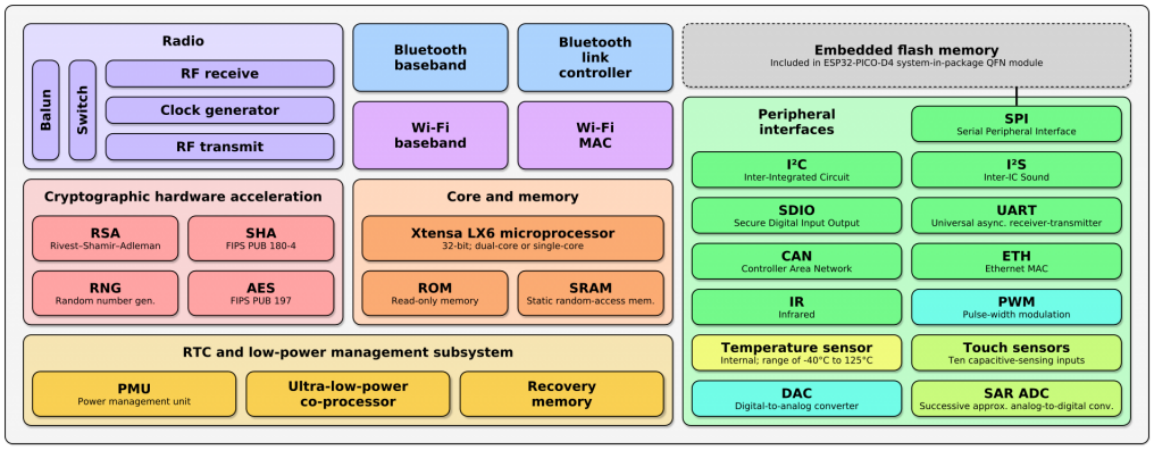


Figura 5 - Diagrama de blocos ativos no modo ativo.

- Modem sleep mode onde está tudo ativo menos os módulos de Wi-Fi, de Bluetooth e de Rádio que estão desativados (Figura 6), gastando cerca de 3 a 20 mA. Para manter as conexões WiFi / Bluetooth ativas, a CPU, o Wi-Fi, o Bluetooth e o rádio são ativados em intervalos predefinidos. É conhecido como padrão de sono de associação. Durante esse padrão de sono, o modo de energia alterna entre o modo ativo e o Modem sleep mode. O ESP32 permanece conectado ao router através do mecanismo de sinalização DTIM.

Para economizar energia, o ESP32 desativa o módulo Wi-Fi entre dois intervalos de Beacon DTIM e acorda automaticamente antes da próxima chegada do Beacon. O tempo de sono é decidido pelo tempo de intervalo DTIM Beacon do router, que normalmente é de 100ms a 1000ms.

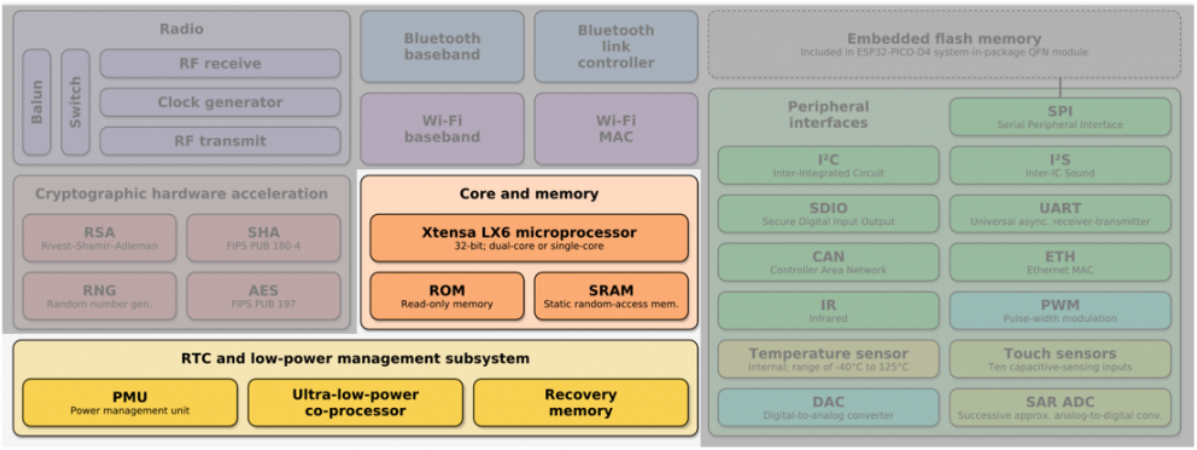


Figura 6 - Diagrama de blocos ativos no Modem sleep mode.

- Light sleep mode que é parecido com o Modem sleep mode. O CPU é pausado desligando os pulsos de clock enquanto que o RTC e o co-processador ULP estão ativos (Figura 7). Isto resulta em menos energia gasta usando cerca de 0,8 mA. Antes de entrar neste modo, o ESP32 guarda o seu estado interno e retoma ao estado guardado ao sair deste modo.

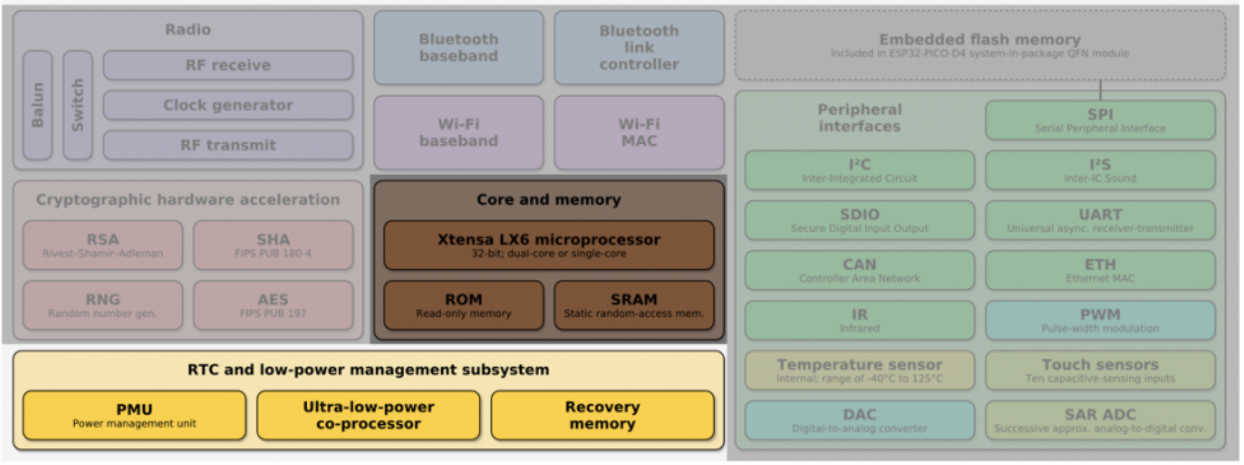


Figura 7 - Diagrama de blocos ativos no Light sleep mode.

- Deep sleep mode a CPU, a maior parte da RAM e todos os periféricos digitais são desligados. Apenas o controlador RTC, os periféricos RTC (incluindo o co-processador ULP) e memórias RTC (lentas e rápidas) é se permanecem ligadas (Figura 8). Neste modo consome-se cerca de 0.15mA. Se o coprocessador não estiver a ser usado usa-se apenas 10μA (0,01 mA).

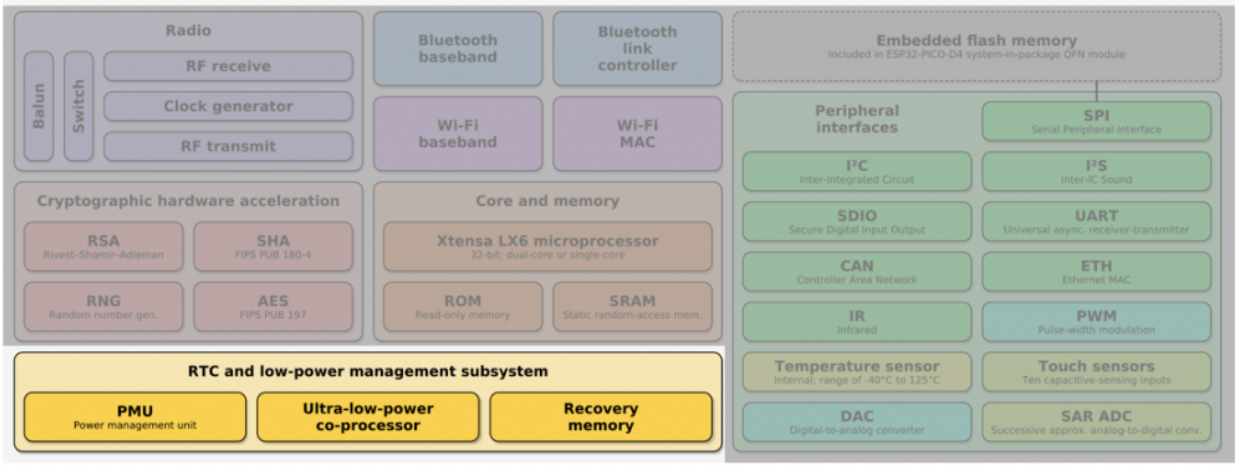


Figura 8 - Diagrama de blocos ativos no Deep sleep mode.

- Hibernation mode que ao contrário do modo de suspensão profunda, o chip também desabilita o oscilador interno de 8MHz, o co-processador ULP e a memória de recuperação RTC também é desligada, o que significa que não é possível preservar nenhum dado durante este modo. Apenas o temporizador RTC com clock lento e alguns GPIOs RTC é que continuam ligados (Figura 9). Estes componentes são responsáveis por despertar o chip. Este modo gasta cerca de 2,5µA (0,0025 mA).

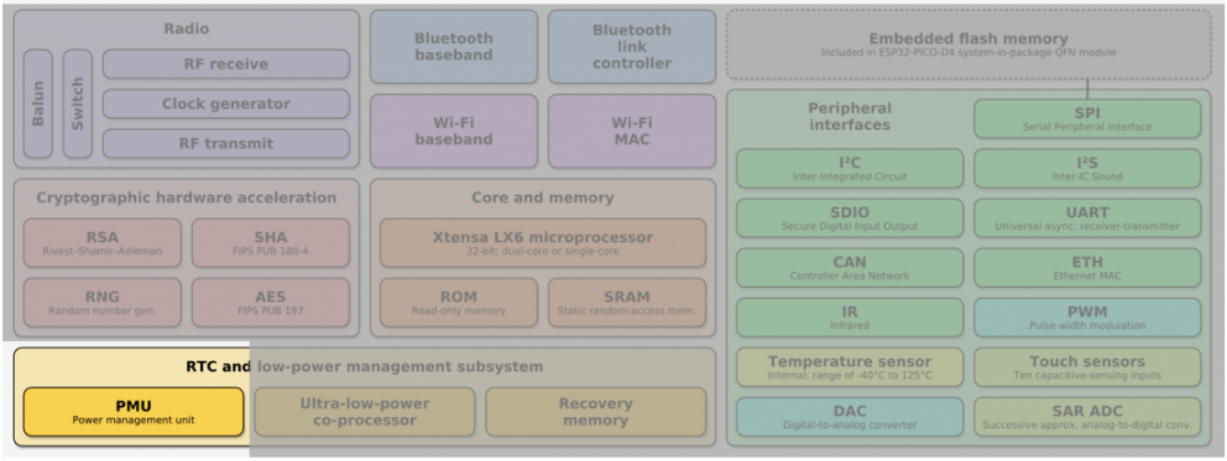


Figura 9 - Diagrama de blocos ativos no Hibernation mode.

4. Características de gestão de energia

O algoritmo de gerir a energia incluído no ESP-IDF pode ajustar a frequência nos periféricos APBs (advanced peripheral bus), no CPU e colocar o chip no modo de light sleep, dependendo do que a placa está a fazer.

Os componentes da placa podem criar e adquirir bloqueios para gerir a energia.

Como por exemplo:

- Um driver de um periférico APB com clock pode pedir que a sua frequência seja 80 MHz quando está a ser utilizado.
- O sistema operativo de tempo real (RTOS) pode solicitar que a CPU seja executada com máxima frequência quando houver tarefas prontas para execução.
- O periférico pode solicitar a desativação do modo light sleep para se ativar.

A utilização de frequências mais altas pelo periférico APB ou pela CPU e a desativação dos modos de sleep causa maior consumo de energia, ou seja é importante uma boa gestão destes modos para que se possa poupar energia colocando inativos os componentes que não estão a ser usados.

4.1 Configurações

Para ativar a gestão de energia no tempo de compilação usa-se o sinal CONFIG_PM_ENABLE. Esta configuração aumenta a latência de 0.2 para 40 us dependendo da frequência do CPU e do modo do core single/dual.

Com a função esp_pm_configure() ativa-se a escala dinâmica de frequência (DFS) e o modo light sleep automaticamente. Esta função tem como argumentos a frequência máxima do CPU, a frequência mínima do CPU e um sinal que indica se o sistema ativa ou não automaticamente o modo light sleep.

Alternativamente pode-se ativar a opção CONFIG_PM_DFS_INIT_AUTO. Assim que a frequência máxima da CPU é dada pela configuração CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ onde a frequência da cpu é bloqueada pela frequência XTAL normalmente de 40 MHz por defeito.

4.2 Bloqueios de gestão de energia

As aplicações têm a capacidade de adquirir e liberar bloqueios a fim de controlar o algoritmo de gestão de energia.

Quando uma aplicação adquire um bloqueio, o funcionamento do algoritmo de gestão de energia é restrito.

A placa ESP32-DEVKITC suporta três tipos de bloqueios (Figura 10):

1. Bloqueio **ESP_PM_CPU_FREQ_MAX** que solicita que a frequência do CPU tenha o seu valor máximo definido através da função esp_pm_configure() podendo ser de 80, 160 ou 240 MHz.
2. Bloqueio **ESP_PM_APB_FREQ_MAX** solicita que a frequência APB seja o valor máximo suportado pela placa, ou seja, 80 MHz.
3. Bloqueio **ESP_PM_NO_LIGHT_SLEEP** desativa o modo light sleep automático.

Quando o bloqueio é retirado, estas restrições impostas são removidas.

Lock	Description
`ESP_PM_CPU_FREQ_MAX`	Requests CPU frequency to be at the maximum value set with `esp_pm_configure()`. For ESP32, this value can be set to 80 MHz, 160 MHz, or 240 MHz.
`ESP_PM_APB_FREQ_MAX`	Requests the APB frequency to be at the maximum supported value. For ESP32, this is 80 MHz.
`ESP_PM_NO_LIGHT_SLEEP`	Disables automatic switching to light sleep.

Figura 10 - Tipos de bloqueios suportados pela placa ESP 32.

5. Aplicações

A placa ESP 32 tem várias aplicações como:

- Networking: A Antena Wi-Fi e o dual-core do módulo permitem a ligação de dispositivos incorporados a routers e a transmissão de dados, como se pode observar na Figura 2.
- Processamento de dados: Inclui o processamento de entradas básicas de sensores analógicos e digitais para cálculos muito mais complexos com um RTOS ou SDK não-OS.
- Conectividade P2P: Cria comunicação directa entre diferentes ESPs e outros dispositivos usando a conectividade IoT P2P.
- Servidor Web: Acesso a páginas escritas em HTML ou linguagens de desenvolvimento.
- Dispositivos industriais inteligentes, incluindo Controladores Lógicos Programáveis (PLCs).
- Dispositivos médicos inteligentes, incluindo monitores de saúde viáveis, como se pode observar na Figura 1.
- Dispositivos inteligentes de energia, incluindo AVAC e termostatos.
- Dispositivos de segurança inteligentes, incluindo câmaras de vigilância e fechaduras inteligentes.

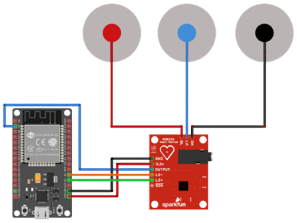


Figura 11 - Monitor de Saúde.

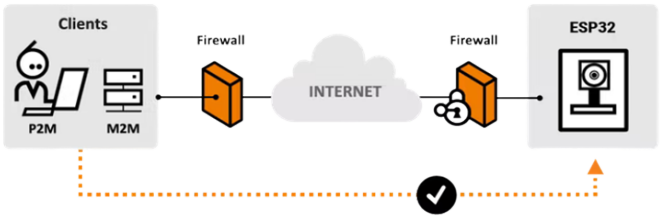


Figura 12 - Aplicação de Rede.

6. Acesso Remoto e Atualizações

Acesso remoto pode ser feito configurando um servidor http na placa, ligar o servidor http à rede, aceder à aplicação desse servidor e controlar a placa, só é preciso configurar o servidor para ligar à rede e ter métodos no servidor para mexer na placa.

A programação OTA (Over-The-Air) é útil para atualizar o código para placas ESP32 que não são facilmente acessíveis. Um único nó central pode enviar uma actualização para múltiplos ESPs na mesma rede.

Há dois métodos de implementação de OTA:

- **OTA básico:** No método OTA básico, o programa é atualizado em ESP32 por via aérea utilizando o Arduino IDE.
- **Actualizador web OTA:** No actualizador web OTA, o programa é actualizado por via aérea utilizando um navegador web.

O mecanismo de actualização OTA permite que um dispositivo se actualize a

si próprio com base nos dados recebidos enquanto o firmware normal está a funcionar (através de Wi-Fi ou Bluetooth).

A OTA requer a configuração da Tabela de Partição do dispositivo com pelo menos duas partições de aplicação "Ota_0 e Ota_1" e uma "Partição de Dados".
As funções de operação OTA escrevem uma nova imagem de firmware de aplicação para qualquer slot de aplicação que não esteja atualmente selecionada para arrancar. Uma vez verificada a imagem, a partição de dados é atualizada para especificar que esta imagem deve ser utilizada para o próximo arranque.

Uma partição de dados deve ser incluída na Tabela de Partição de qualquer projeto que utilize as funções OTA.

Para configurações de arranque de fábrica, a partição de dados não deve conter dados. O ESP-IDF software bootloader iniciará a aplicação de fábrica se esta estiver presente na tabela de partição. Se nenhuma aplicação de fábrica estiver incluída na tabela de partição, a primeira partição OTA disponível (normalmente ota_0) é inicializada.

Após a primeira actualização, a partição de dados é actualizada para especificar qual a partição de aplicação que deve ser inicializada a seguir.
A partição de dados tem dois sectores flash, para evitar problemas se houver uma falha de energia enquanto está a ser escrita. Os sectores são apagados independentemente e escritos com dados correspondentes, e se não concordarem, é utilizado um campo de contador para determinar qual o sector que foi escrito mais recentemente.

O principal objectivo do rollback da aplicação é manter o dispositivo a funcionar após a actualização. Esta característica permite voltar à aplicação que estava a trabalhar anteriormente no caso de uma nova aplicação ter erros críticos.

7. Bibliografia

- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power_management.html
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html
- https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/
- https://www.makers.pt/2019/01/18/modos-de-sleep-do-esp32-e-seu-consumo-de-energia/
- https://deepbluembedded.com/esp32-adc-tutorial-read-analog-voltage-arduino/
- https://newscrewdriver.com/2021/02/22/evaluating-my-options-for-esp32-development/