rfid.c CMakeLists.txt

Projeto final ‘RFID’ {

[Arquiteturas para Sistemas Embutidos]

}

Diogo Correia, 90327
João Simões, 88930
TP2

rfid.c

CMakeLists.txt

Componentes {

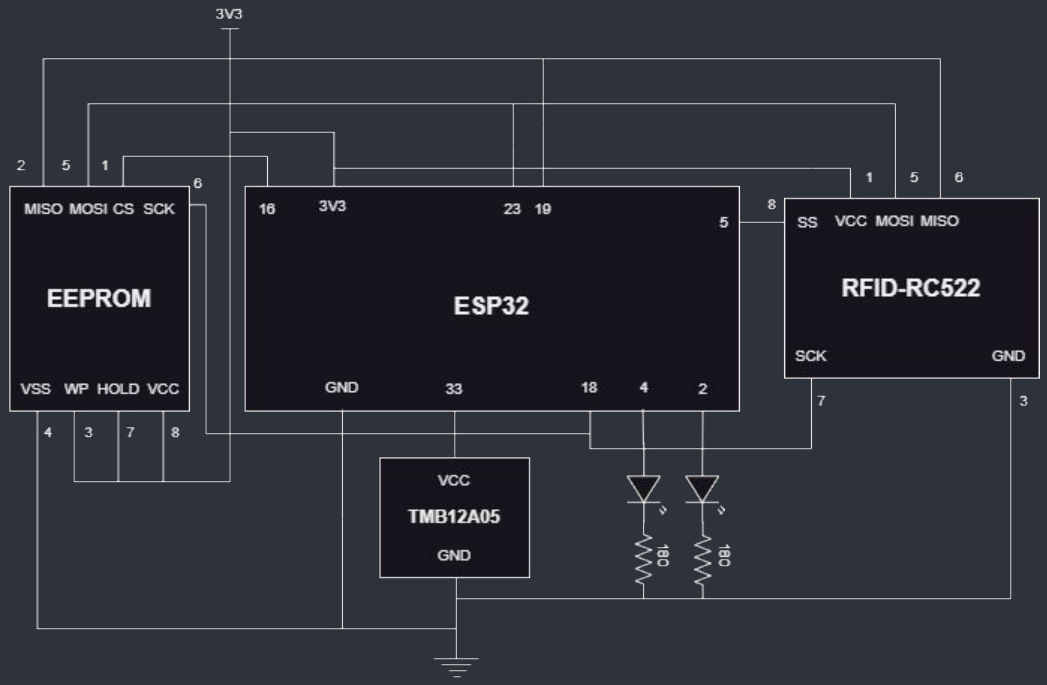
- **ESP32** — Base do *embedded system*
- **RFID-RC522** — Sensor de leitor de cartões RFID
- **Buzzer TMB12A05** — Indicador sonoro de acessos
- **LEDs** — Indicador visual de acessos
- **EEPROM** — Memória para guardar o último acesso

}

rfid.c

CMakeLists.txt

Hardware {

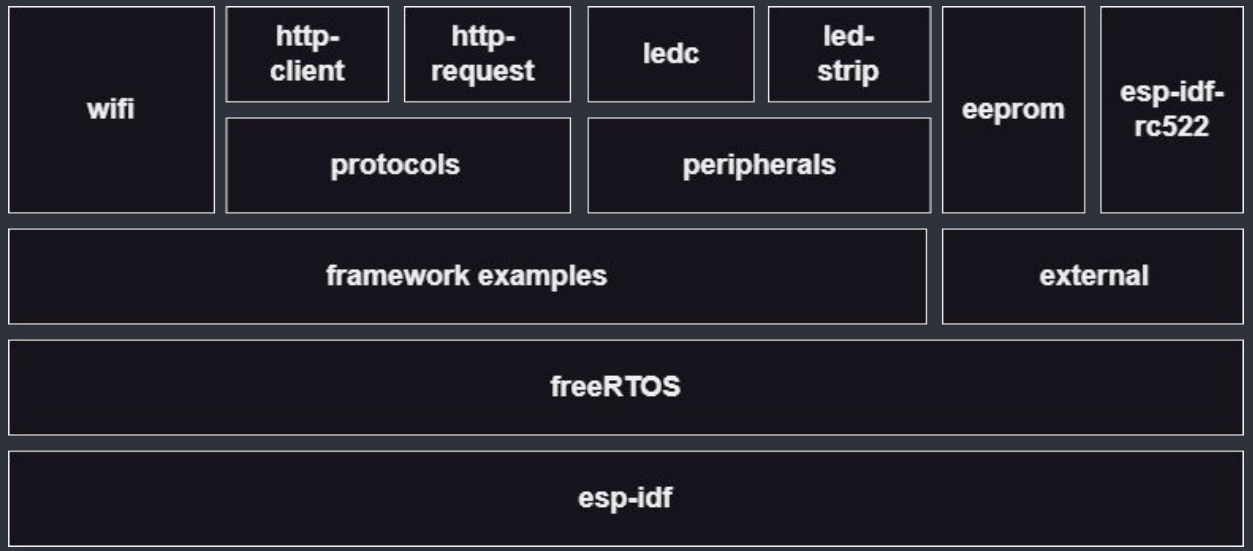


}

rfid.c

CMakeLists.txt

Software {

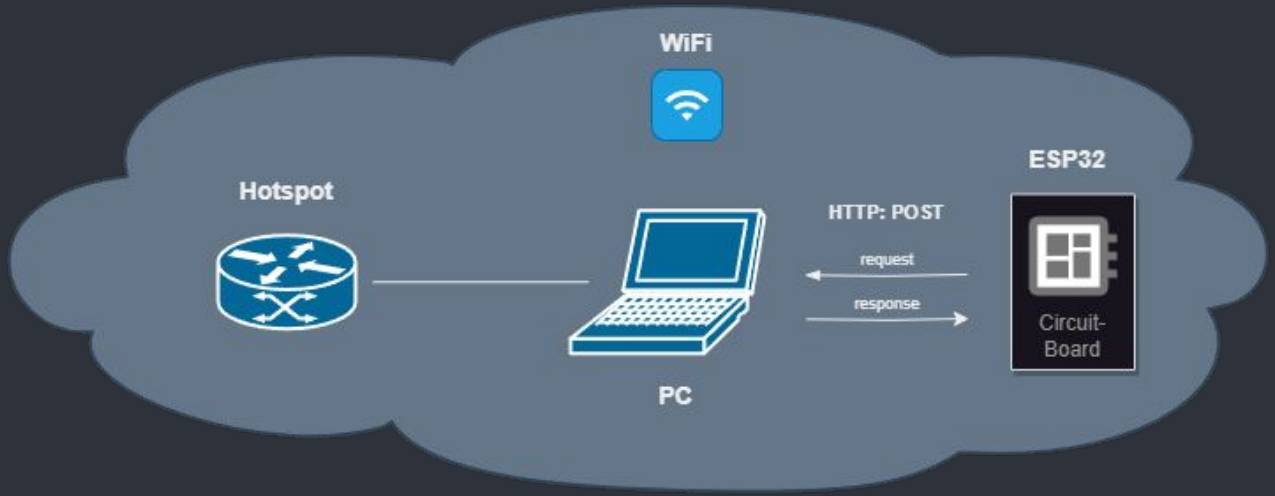


}

rfid.c

CMakeLists.txt

Comunicações {



}

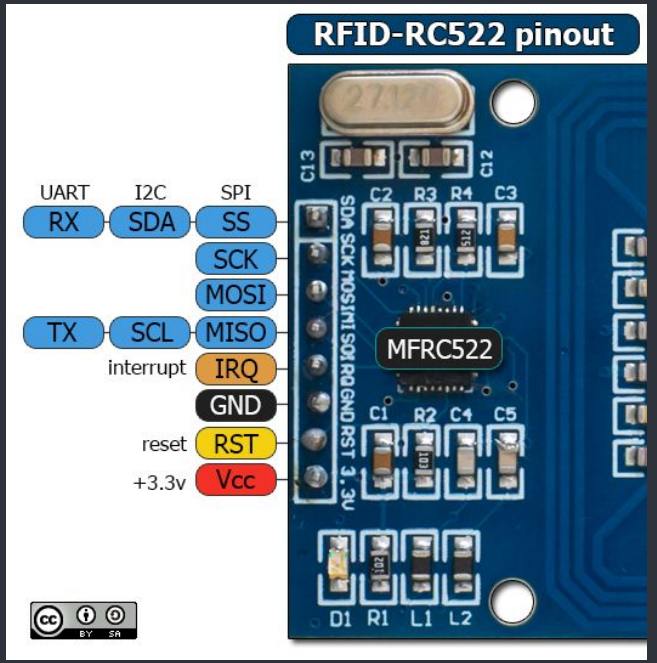
rfid.c

CMakeLists.txt

Leitura de cartões {

* O *serial number* dos cartões é lido através de uma biblioteca externa, que faz uso da antena do sensor RFID-RC522.
* O leitor comunica com o microcontrolador através do protocolo SPI, mas também oferece suporte para comunicação através dos protocolos I2C e UART.

}



Conectividade de rede {

* A ESP32 funciona como uma *station*, ligada a uma rede *hotspot* portátil gerada por um *smartphone*.
* Atua como um cliente que envia um HTTP request para um servidor a correr num PC também ligado à mesma rede e aguarda pela resposta.
* O servidor recebe o pedido POST da ESP32 com o número do cartão, verifica se o mesmo tem acesso ao constar ou não numa lista de autorizações (um ficheiro de dados local) e responde com 1 ou 0, respetivamente.

}

Escrita e leitura na memória {

* Utiliza-se a EEPROM para manter o registo do último cartão lido pelo sensor.
* No contexto do projeto, serve como uma caixa negra para armazenar a última tentativa de acesso.
* É utilizado o protocolo de comunicação SPI com dois *slaves* (EEPROM em comunhão com o leitor de RFID).

}

Dashboard remoto {

* Servidor web desenvolvido com FastAPI (em Python).
* Responsável por lidar com o pedido POST da ESP32 e de lhe responder com sucesso ou insucesso.
* Permite visualizar em tempo-real o histórico de acessos, que ficam armazenados localmente num ficheiro de dados.
* Permite ainda adicionar e remover acesso de um determinado cartão através de uma interface gráfica.

}

Requisitos {

- ✅ **ESP32DevKitC** como base do embedded system
- ✅ **Sensor** ligado ao kit ESP32DevKitC
- ✅ Aplicação desenvolvida em **C** com **freeRTOS**
- ✅ Exploração dos **periféricos** do ESP32
- ✅ Dados do sensor numa **Dashboard** por **WiFi**
- ✅ Ligação por **Terminal**

rfid.c

CMakeLists.txt

- [x] Atualizações remotas (**Over-the-Air**) do sistema
- [ ] **Atuador** complementar ao sensor
- [x] Suporte de sistema de ficheiros para **armazenamento local**
- [x] Exploração de **funcionalidades de desenvolvimento**
- [ ] Modos de **baixo consumo energético** do ESP32

}

Referências {

Biblioteca externa ‘esp-idf-rc522’

C library for interfacing ESP32 with MFRC522 RFID card reader.
github.com/abobija/esp-idf-rc522

Exemplos fornecidos no diretório do ESP-IDF

- examples\wifi\getting_started\station
- examples\protocols\esp_http_client
- examples\protocols\http_request

}

rfid.c

CMakeLists.txt

Contribuição dos autores {

* Diogo Correia, 90327

* João Simões, 88930

}

rfid.c

CMakeLists.txt

Demonstração {

```
    run();
}
```

Conclusão {

* Praticamente todos os objetivos deste projeto foram alcançados, o que demonstra que seria uma implementação viável para o nosso problema inicial de criar um sistema para abertura de uma porta com recurso a cartões RFID.
* Este trabalho permitiu-nos aumentar o nosso conhecimento a nível de hardware (ESP32) e de programação de baixo nível (ESP-IDF) e ainda ganhar experiência com sistemas operativos real-time (FreeRTOS).
* Motivou-nos a futuramente continuar a desenvolver mini-projetos em microcontroladores para resolver desafios do dia-a-dia.

}