Relatório da Atividade: Interface SPI com o leitor de RFID RC522

Parente , Nuno Duarte, 202107940
Sousa, Sérgio Quelhas Ferreira de, 202106740
Licenciatura em Engenharia Física

Departamento de Engenharia Eletrónica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor: Mendonça, Hélio Sousa ; Souza, João
Unidade Curricular: Eletrónica Digital e Microprocessadores (EEC3001)
Bancada: 03
Data: 25 de setembro de 2024

Resumo

Neste trabalho foi possível estabelecer uma ligação entre o ESP32 e o sensor RC552, utilizando o protocolo SPI. Conseguimos detetar os cartões e comunicar ao ESP o seu UID, permitindo que virtualmente sejam guardadas e alteradas informações relevantes associadas aos cartões.

1 Tabela de Conteúdos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2 Introdução

Neste trabalho utilizamos o leitor RFID RC522 para realizar a leitura de cartões e acessórios de registo. Este sistema pode ser utilizado em diversos casos do dia a dia, como por exemplo: cartão de acesso nos laboratórios e áreas restritas da universidade, registo da entrada e saída de livros numa biblioteca, transportes públicos, entre outros. Neste trabalho procuramos imitar estes sistemas.

3 Abordagem Proposta

3.1 Funcionamento do RFID RC522

O sensor RC522 é constituído essencialmente por um microship responsável pelo processamento de dados (MFRC522) e uma antena para comunicação sem fio com as tags RFID. Este módulo consegue comunicar com um microcontrolador usando os protocolos UART, SPI ou I2C. De seguida podemos ver o esquemático do microchip.

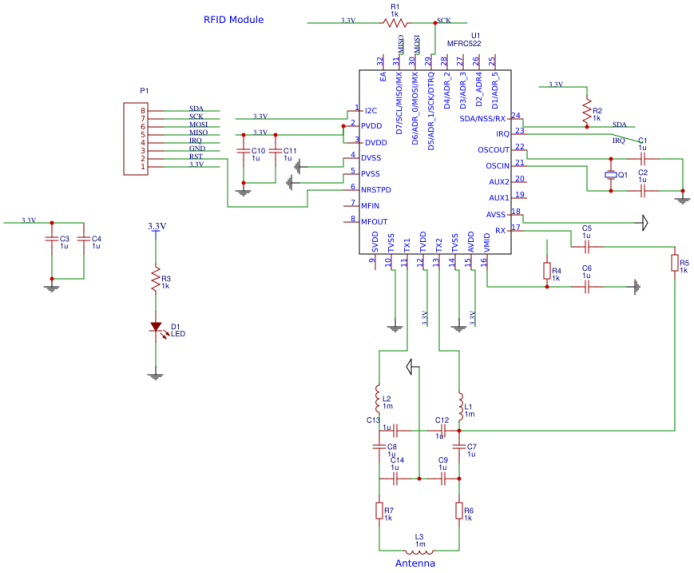

Figura 1: *Schematic* do microship MFRC522

Neste projeto realizamos a comunicação entre o ESP32 e o MFRC522 com SPI. Passamos agora a explicar a forma como são lidas as tags RFID:

O master envia ao sensor um sinal específico de 1 byte. Em resposta, a antena do módulo gera um campo eletromagnético na frequência de 13.56 MHz. Quando uma tag RFID entra nesse campo, ela usa a energia para alimentar o seu circuito interno. Desta forma, estes 2 componentes comunicam usando um protocolo UART sem contacto, que na prática consiste em o leitor variar a intensidade do campo eletromagnético. Conforme aquilo que o master envia ao MFRC522, este processo permite escrever ou ler dados do cartão ou simplesmente ler o seu UID (Identificador Único).

3.2 Apresentação da ideia do projeto

Neste projeto elaboramos um sensor que nos indica se um cartão possui ou não acesso a um determinado serviço, através de três leds: um vermelho, um verde e um azul (sem acesso, com acesso e registo inválido, respetivamente). Para além disso, o ESP32 envia os registos de passagem dos cartões e o tempo desses acontecimentos para uma página web via websocket. Assim, a informação dos cartões e respetivos utilizadores encontra-se sempre registados nesta página *web*, que na realidade representa um "servidor". Nesta página é possível observar e alterar os dados dos utilizadores.

3.3 Explicação do circuito realizado

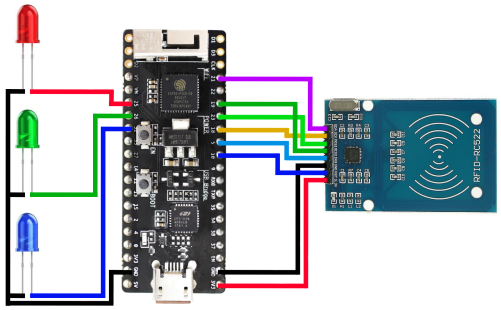

Figura 2: Esquema representativo das ligações feitas neste projeto.

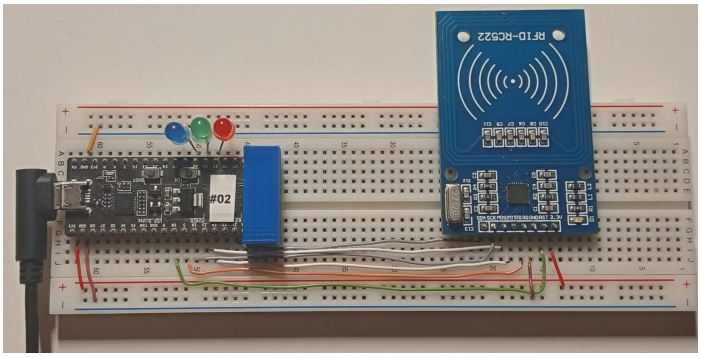

Figura 3: Imagem do circuito montado.

Conforme visível na figura acima, simplesmente ligamos os terminais MOSI,MISO,SCK da comunicação SPI aos pinos 23,19,18 que têm estas funcionalidades. Ligamos também os terminais SDA,IRQ e RST, apesar de não terem sido utilizados.

Ligamos ainda os LEDs vermelho, verde e azul previamente referidos aos pinos 25,26,32. Como podemos ver, não colocamos resistências, tendo-se utilizado as resistências Pull Up inbutidas nestes pinos do ESP32, de forma a não danificar os LEDs.

3.4 Explicação do Funcionamento do Sistema

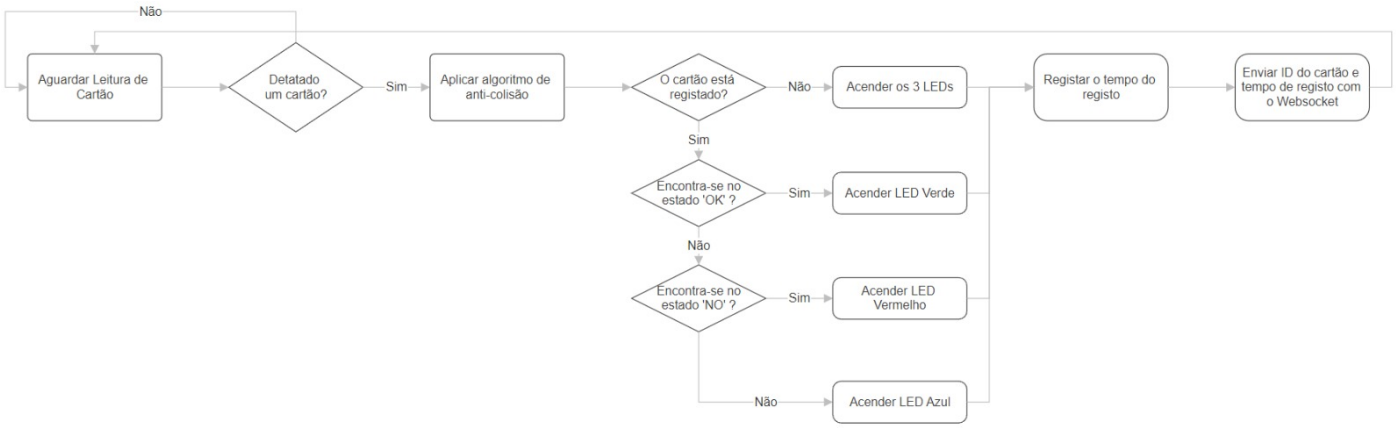

Figura 4: Fluxograma do processo de Leitura dos IDs

Para um determinado cartão temos 4 hipóteses: ele pode estar registado no sistema e com estado OK ou NO (com ou sem acesso, respetivamente). Ele pode ainda estar registado, mas ter o estado 'Nao registado' que pode representar um cartão suspenso, caducado ou cancelado. Por fim temos o caso em que o cartão ainda não foi registado no sistema.

Todos estes casos resultam numa resposta diferente dos LEDs, como podemos observar acima. Todas as passagens de cartões, independentemente do estado, são registadas no website.

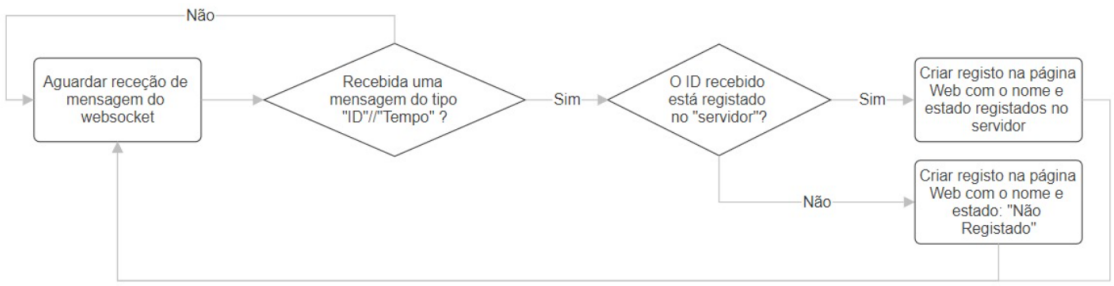

Figura 5: Fluxograma do funcionamento do website

Depois de o UID e tempo de passagem do cartão serem enviados por websocket para a página web seguimos o fluxograma acima. Simplesmente, se o cartão está registado no sistema, o registo de passagem do cartão é feito com essa informação, senão é feito um registo em que os dados em falta são preenchidos com "Não Registado".

Figura 6: Fluxograma do processo de registo de um cartão

Por fim, (como será possível ver na Secção 4) após ter sido feito um registo de passagem de cartão podemos selecionar o cartão passado e alterar a informação a ele associada. Após premir o botão responsável de submeter estes dados, eles são enviados ao ESP32, que depois reenvia aos websocket clients.

Isto pode parecer redundante, mas é feito para garantir que uma alteração feita num dos websocket clients ficará guardada em todos eles. De realçar, claro, que todo este processo teve que ser feito porque não usamos um servidor neste projeto.

3.5 Explicação de código relevante

Vamos agora demonstrar brevemente como implementamos algumas das funcionalidades acima descritas. Neste projeto, para realizar a comunicação com o leitor RFID usamos o módulo "micropython-mfrc522-esp32"[1].

Comecemos por ver o código do main.py:

```
while True:
    stat, tag_type = rdr.request(rdr.REQIDL)
    if stat == rdr.OK:
        stat, raw_uid = rdr.anticoll()
        if stat == rdr.OK:
            uid = "0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3])
            if uid != previous and previous2 == previous:
                if uid in ids:
                    i = ids.index(uid)
                    if states[i] == 'OK':
                        green.value(True)
                    elif states[i] == 'NO':
                        red.value(True)
                    elif states[i] == 'Nao Registado':
                        blue.value(True)
```

Figura 7: Função responsável por ler cartões do main.py

Na figura acima podemos observar um loop que permite detetar a presença de cartões (rdr.request) e após detetar, determina o seu uid (rdr.anticoll), fazendo com que apenas seja detetado 1 cartão quando vários se encontram no leitor e envia sinal para os leds correspondente ao seu estado. Podemos ver que este loop se repete infinitamente, ou seja, estamos sempre a verificar se foi colocado um cartão no leitor.

No main.py usamos o módulo asyncio, de modo que temos 2 tarefas: a função "do_read()"mostrada acima e a tarefa "app", que tem a função de criar a página web em cada websocket client e de receber mensagens dos mesmos.

```
def request(self, mode):

        self._wreg(0x0D, 0x07)
        (stat, recv, bits) = self._tocard(0x0C, [mode])

        if (stat != self.OK) | (bits != 0x10):
                stat = self.ERR

        return stat, bits
```

Figura 8: Função request do módulo MFRC522

```
def anticoll(self):

        ser_chk = 0
        ser = [0x93, 0x20]

        self._wreg(0x0D, 0x00)
        (stat, recv, bits) = self._tocard(0x0C, ser)

        if stat == self.OK:
                if len(recv) == 5:
                        for i in range(4):
                                ser_chk = ser_chk ^ recv[i]
                        if ser_chk != recv[4]:
                                stat = self.ERR
                else:
                        stat = self.ERR

        return stat, recv
```

Figura 9: Função "anticoll"do módulo MFRC522.

Temos agora as 2 funções do módulo "micropython-mfrc522-esp32"que usamos no nosso projeto. De forma resumida, ambos as funções se resumem no envio de uma mensagem na forma 0x.. para o sensor. Estas mensagens são definidas pelo fabricante do sensor e executam um determinado "comando"no sensor de modo a que ele realize uma certa ação. Por exemplo o comando "0x93"permite que o sensor entre no modo anticolisão e apenas leia um cartão caso dois sejam colocados nas proximidades ao mesmo tempo.

Na função de anticolisão devemos realçar que o parâmetro "recv"retornado pela função consiste numa lista de 5 números inteiros e é aquilo que consideramos como sendo o UID do cartão. Notemos que no loop em que fazemos "ser_chk=ser_chk ^ rev[i]"estamos a realizar a operação XOR repetidamente, como uma forma de verificar os dados recebidos (o último elemento da lista serve como número de controlo).

Neste código torna-se evidente que não usamos as funcionalidades de leitura nem escrita do leitor de cartões. Isto foi feito propositadamente, pois assim todo o sistema funciona no "servidor"e os cartões apenas servem como uma forma de saber quando um certo utilizador fez registo, sem necessitarem de mais informação. Outra razão para esta escolha foi que para ler e escrever nos cartões é necessário eles estarem em repouso sobre o leitor por uns instantes para a informação ser transmitida. Na forma como executamos o sistema, os dados e estado dos cartões pode ser alterado na página web a qualquer instante, de forma quase instantânea.

4 Resultados

4.1 Resultados finais

Ao testar o sistema, foi possível realizar a leitura de cartões Andante da Metro do Porto, bem como do cartão estudante da Universidade do Porto. Por fim, conseguimos obter o seguinte resultado:

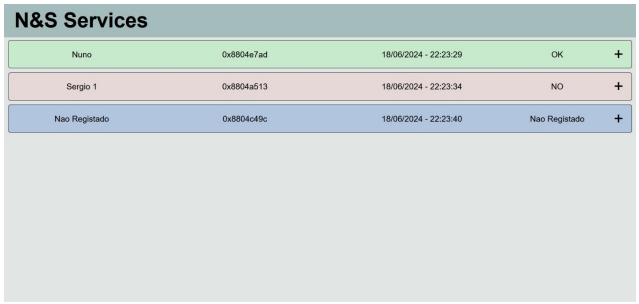

Figura 10: Página Web - PC

Figura 11: Página Web - PC com aba aberta

Tal como podemos ver, obtivemos uma página web onde é registado a passagem dos cartões e é identificado o seu estado, nome, ultima data de acesso e o UID do cartão. Clicando no botão "+"disponível ao ler um cartão é possível alterar diversas opções, tais como: utilizador, estado, email, telemóvel e morada. Estas opções

Estes dados estão disponíveis para todos os dispositivos que acessarem o endereço, inclusivé um dispositivo móvel, tal como podemos ver de seguida.

Figura 12: Página Web - Telemóvel

Figura 13: Página Web - Telemóvel com aba aberta

Verificamos que o sistema funciona como esperado, as respostas dos LEDs e cores dos registos sempre concordantes. A alteração de dados funcionou sempre e nunca houve muito delay. Vimos ainda que o computador e telemóvel ligados a este sistema se mantiveram sempre em concordância.

4.2 Recomendações futuras

Vejamos possíveis melhorias futuras a este projeto. Do ponto de vista de implementação do software, este projeto poderia ser adaptado para receber os valores de uma base de dados (de um servidor externo), de modo a que fosse acessível em vários dispositivos e que o ESP32 apenas tivesse que notificar o servidor quando um certo cartão fosse passado. Poderia ainda ser útil implementar um sistema de pesquisa de utilizadores, porque na versão atual só podemos alterar as definições de um cartão se ele for lido.

5 Conclusão

Neste projeto comunicamos com o sensor RFID RC522 usando o protocolo SPI. Com ele, fazendo-se leitura de UIDs de cartões e imitando o que acontece nos dias de hoje nos transportes públicos, bibliotecas entre outros.

Conseguimos fazer um interface Wi-Fi com protoclo HTML em que se observou e controlou o sistema de leitura de cartões.

Foi ainda observada a aplicabilidade desta tecnologia na vida real, observando-se compatibilidade entre este sensor e cartões usados no dia a dia.

6 Bibliografia

[1] Micropython-mfrc522-esp32. `https://github.com/Tasm-Devil/micropython-mfrc522-esp32/tree/master`, Consultado a 27/05/2024.