Jhonatan Cléto

Projeto de Smart Lock utilizando MicroPython e o Microcontrolador ESP32

Campinas, SP

2021

Smart Lock é uma fechadura eletromecânica projetada para realizar travamento e destravamento de uma porta quando recebe tais instruções através de um mecanismo autorizado. O projeto Smart Campus da Prefeitura Universitária da Unicamp, desenvolve uma versão de Smart Lock utilizando o microprocessador Raspberry Pi. Essa versão utiliza um sistema de tag RFID e um sistema de QR CODE para realizar as operações de travamento e destravamento de portas.

Na tentativa de implementar uma versão do Smart Lock, utilizando o microcontrolador ESP32, realizei um estudo sobre a viabilidade de tal implementação, no qual verifiquei quais recursos, presentes no Smart Lock com Raspberry PI, poderiam ser portados para o ESP32.

O software do Smart Lock no Raspberry PI foi desenvolvido em Python, de modo a manter a mesma linguagem de programação, facilitando a portabilidade do software, na elaboração do Smart Lock com o ESP32, utilizei o MicroPython, que é uma implementação do interpretador de Python para microcontroladores como o ESP32. O MicroPython possui a maioria das bibliotecas padrões do Python já incluídas, além de permitir a inclusão de bibliotecas e módulos de terceiros caso seja necessário.

O projeto do Smart Lock, utiliza o módulo RFID MFRC522, o qual possui uma biblioteca em python compatível com o Raspberry Pi. Assim, uma das primeiras pesquisas que realizei, foi verificar a existência de uma biblioteca para esse módulo, compatível com MicroPython. Como resultado, encontrei duas opções de bibliotecas/módulos de código aberto, vistas na Tabela 1, que atenderam o requisito de compatibilidade.

Biblioteca/Módulo	Link
MFRC522 ESP32	**https://github.com/Tasm-Devil/micropython-mfrc522-esp32**
MicroPython MFRC522	**https://github.com/wendlers/micropython-mfrc522**

Tabela 1 - Bibliotecas para o módulo RFID MFRC522 compatíveis com MicroPython

Após encontrar bibliotecas para utilizar o módulo RFID, desenvolvi um código para testar o funcionamento e eficiência na leitura do sensor com a ESP32. Ambas as opções da Tabela 1 funcionaram corretamente. Com isso, escolhi manter a biblioteca MFRC522 ESP32 no projeto do Smart Lock, pois ela ocupa menos espaço na memória da ESP32.

No código da Raspberry Pi, o software faz uma verificação em um servidor para confirmar se a tag RFID, lida pelo sensor do módulo MFRC522, está autorizada a liberar o acesso da porta, para tanto, ele faz algumas requisições HTTP ao servidor. Assim, para implementar essa funcionalidade na versão com ESP32, necessitei encontrar bibliotecas para MicroPython que permitissem a conexão do microcontrolador com a rede e que ele efetue requisições HTTP.

No caso da conexão com a rede, o MicroPython vem por padrão com a biblioteca network, que utiliza o módulo wifi da ESP32 para fazer a conexão com qualquer rede conhecida. Para efetuar as requisições HTTP, o MicroPython possui a biblioteca urequests, que é uma versão reduzida da biblioteca requests do Python. Com essas duas bibliotecas, realizei um teste para verificar a funcionalidade e tempo de resposta da ESP32 em requisições HTTP. O resultado foi satisfatório, o microcontrolador mostrou-se capaz de realizar tais requisições em um tempo viável para o projeto do Smart Lock. Contudo,

durante os testes, encontrei um problema com algumas firmwares do MicroPython para o ESP32, o qual consistia do microcontrolador não conseguir realizar as requisições HTTP, pois ele não foi capaz de traduzir o endereço web, para o qual a requisição era feita, em um endereço IP. Na Tabela 2, apresento as firmwares testadas, identificando quais delas o problema foi observado e quais a requisição ocorreu corretamente. Nas figuras 1 e 2, apresento casos de erro e sucesso, durante os testes com requisições HTTP na ESP32.

Firmware	Requisição HTTP
1.12	**Funcional**
1.13	**Não Funcional**
1.14	**Não Funcional**
1.16	**Funcional**

Tabela 2 - Firmwares avaliadas durante o teste de Requisições HTTP

```
  MicroPython v1.14 on 2021-02-02; ESP32 module with ESP32
  Type "help()" for more information.
MicroPython v1.14 on 2021-02-02; ESP32 module with ESP32
Type "help()" for more information.
>>> ets Jun  8 2016 00:22:57

  rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
  configsip: 0, SPIWP:0xee
  clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
  mode:DIO, clock div:2
  load:0x3fff0018,len:4
  load:0x3fff001c,len:5008
  ho 0 tail 12 room 4
  load:0x40078000,len:10600
  ho 0 tail 12 room 4
  load:0x40080400,len:5684
  entry 0x400806bc
  connecting to network homewifi_net ...
  Successful Connection
  Network configuration: ('192.168.0.24', '255.255.255.0', '192.168.0.1', '181.213.132.2')
  MicroPython v1.14 on 2021-02-02; ESP32 module with ESP32
  Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
  Traceback (most recent call last):
    File "<stdin>", line 2, in <module>
    File "requests.py", line 112, in get
    File "requests.py", line 53, in request
  OSError: -202

>>>
```

Figura 1 - Tentativa Falha de Efetuar uma requisição HTTP usando a ESP32

```
>>> ets Jun  8 2016 00:22:57

  rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
  configsip: 0, SPIWP:0xee
  clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
  mode:DIO, clock div:2
  load:0x3fff0030,len:5640
  load:0x40078000,len:12696
  load:0x40080400,len:4292
  entry 0x400806b0
  connecting to network homewifi_net ...
  Successful Connection
  Network configuration: ('192.168.0.24', '255.255.255.0', '192.168.0.1', '181.213.132.2')
  MicroPython v1.16 on 2021-06-23; ESP32 module with ESP32
  Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  <class 'Response'>
  {
    "userId": 1,
    "id": 1,
    "title": "quidem molestiae enim"
  }
  <class 'str'>
>>> response.status_code
200
```

Figura 2 - Tentativa bem sucedida de Efetuar uma requisição HTTP na ESP32

Outra funcionalidade presente no Smart Lock com Raspberry Pi, é o recurso de validação do acesso à porta por meio da leitura de QR CODE. Para essa funcionalidade há a necessidade de que o microcontrolador acesse uma câmera, a qual captura uma foto do QR CODE. Então, o software do Smart Lock, com o auxílio de bibliotecas de tratamento de imagem e leitura de QR CODE, faz uma requisição ao servidor, perguntando se o código identificado é autorizado a liberar a porta.

O modelo de ESP32 que utilizei nos testes anteriores não possui uma câmara. Com isso, para verificar a viabilidade de implementação do recurso de leitura de QR-CODE, utilizei o modelo de ESP32-CAM, o qual como o nome sugere possui uma câmera integrada à sua placa. O ESP32-CAM, é comercializado na mesma faixa de preços da ESP32 convencional. No entanto, esse modelo possui menos portas IO que o modelo convencional, limitando o número de sensores que podem ser usados concomitantemente.

Infelizmente não existe uma firmware oficial de MicroPython para o ESP32-CAM, contudo, existem versões não oficiais e tutoriais ensinando a como compilar firmwares de MicroPython para esse modelo. No caso dos testes que eu realizei, optei por utilizar uma firmware já compilada para o ESP32-CAM. No entanto, não consegui testar a câmera do dispositivo, pois ela estava com defeito.

Em relação às bibliotecas para decodificação de QR CODE, não encontrei nenhuma versão compatível com MicroPython. Assim, por enquanto, não é possível implementar, com auxílio de bibliotecas de terceiros, a funcionalidade de leitura de QR CODE no Smart Lock com ESP32 utilizando MicroPython.

Outros recursos importantes para o projeto do Smart Lock são, a possibilidade de executar múltiplas tarefas em simultâneo e um sistema de log, para o monitoramento das atividades do sistema. O ESP32 é um microcontrolador equipado com um processador com 3 núcleos, assim, é possível executar mais de uma tarefa simultaneamente. O MicroPython possui uma API responsável por gerenciar, em baixo nível, a execução de tarefas em diferentes threads. Portanto, a execução de tarefas em simultâneo é viável em uma versão do Smart Lock com ESP32. Quanto ao recurso de logs do sistema, o MicroPython possui a

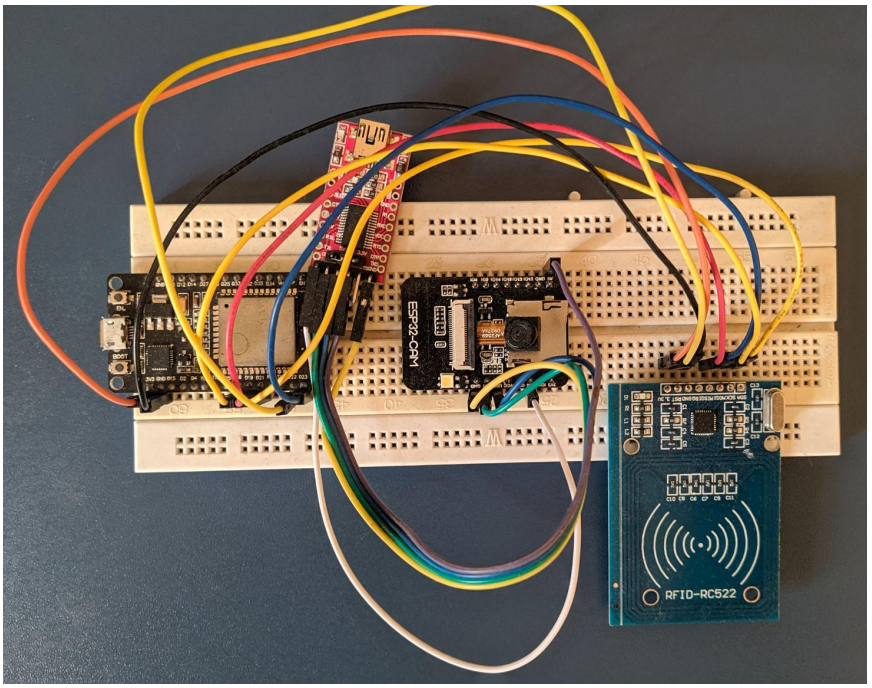
biblioteca ulogging, que é uma implementação da biblioteca logging do Python, responsável por prover uma interface para elaboração de logs customizáveis. Com isso, esse recurso também é viável na ESP32.

Após pesquisar, testar e analisar a portabilidade do Smart Lock, do Raspberry Pi, para o ESP32, identificando os recursos que podem e os que não podem ser implementados no microcontrolador, no qual o número de portas IO, capacidade de armazenamento e poder de processamento são inferiores. Concluo que, não é possível implementar uma versão do Smart Lock, com todas as funcionalidades presentes na atual versão do Raspberry Pi, no ESP32. No atual cenário, ainda não existem certas bibliotecas necessárias para algumas das funcionalidades do Smart Lock, bem como o número de portas do microcontrolador não conseguirá suprir a quantidade de portas usadas pelos sensores e atuadores do projeto, caso o modelo ESP32-CAM seja utilizado.

Pelo que percebi, durante o estudo, o MicroPython ainda é um projeto em desenvolvimento. Embora muitos recursos do Python já tenham sido portados, vários outros ainda não foram, como por exemplo as bibliotecas para decodificação de QR CODE e alguns recursos precisam ser refinados, como a biblioteca de sockets, na qual em algumas versões de firmware, algum problema nela impossibilitou a execução de requisições HTTP. Assim, conforme o MicroPython for evoluindo, é bem provável que mais recursos do Smart Lock possam ser portados para o ESP32.

Por fim, acredito que é possível implementar uma versão de Smart Lock usando o ESP32, com menos funcionalidades que a versão com Raspberry Pi. O Microcontrolador mostrou-se eficiente na realização de tarefas em simultâneo, na execução de requisições HTTP, e na identificação de Tags utilizando o sensor RFID. Portanto, é viável a elaboração de um Smart Lock, que utiliza tais recursos bem como alguns atuadores, como relés e leds.

Anexos



Anexo 1 - Placas ESP32 utilizadas nos testes

Anexo 2 - SL - ESP32 - Pasta Com os arquivos desenvolvidos para testes.