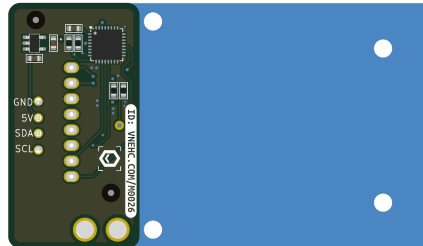


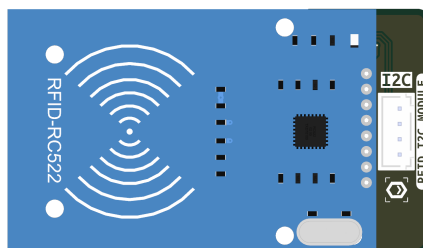
► PDF Export Style (Hidden on web)

Tài liệu hướng dẫn sử dụng MKE I2C RFID (RC522)



1. Giới thiệu chung

MKE_I2C_RFID là thư viện hỗ trợ giao tiếp với module RFID RC522 thông qua chuẩn I2C, được phát triển dành riêng cho hệ sinh thái MakerEdu. Việc sử dụng chuẩn I2C thay vì SPI giúp tiết kiệm tối đa số lượng chân GPIO của vi điều khiển, cho phép kết nối nhiều module khác nhau trên cùng một đường truyền.



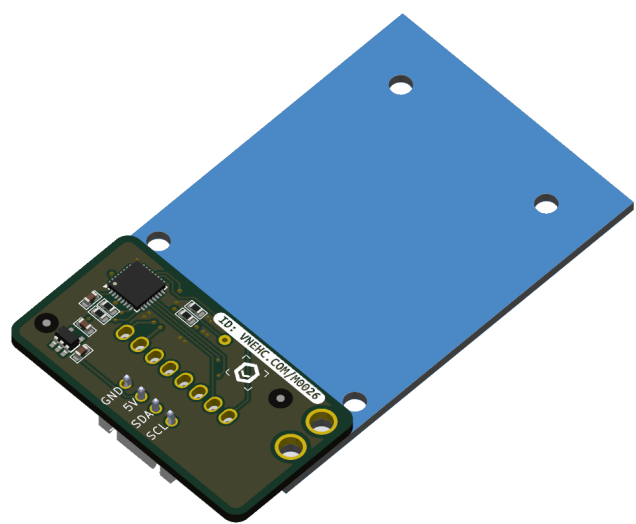
2. Cài đặt thư viện

QUAN TRỌNG: Để đảm bảo tính tương thích và tự động quản lý các dependencies, vui lòng cài đặt gói thư viện tổng hợp **MKE_ONE**.

- 1. Mở phần mềm Arduino IDE.
- 2. Điều hướng tới **Sketch** -> **Include Library** -> **Manage Libraries...**
- 3. Trong ô tìm kiếm, nhập **MKE_ONE**.
- 4. Tìm đến thư viện **MKE_ONE** và nhấn **Install**.

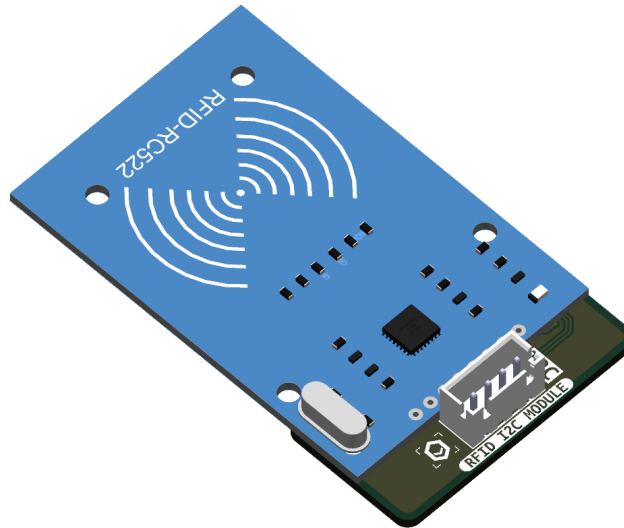
Sau khi cài đặt xong, thư viện **MKE_I2C_RFID** sẽ tự động có sẵn cùng với các thư viện khác trong hệ sinh thái MakerEdu.

3. Đấu nối phần cứng



Chân Module RFID	Chân Arduino / ESP32	Chức năng
GND	GND	Nối đất (Ground)
VCC	5V / 3.3V	Nguồn cấp
SDA	SDA (A4 trên Uno)	Data I2C
SCL	SCL (A5 trên Uno)	Clock I2C

(Lưu ý: Bạn có thể cắm qua các header tương ứng trên mạch mở rộng của MakerEdu như hình minh họa dưới đây)



4. API Cơ bản (Basic APIs)

Dưới đây là một số hàm thông dụng nhất của lớp `MKE_I2C_RFID` để bạn đọc và ghi thẻ:

- `bool begin(uint8_t address = 0x29)`: Khởi tạo module với địa chỉ I2C. Trả về `true` nếu thành công.
- `bool isCardPresent()`: Kiểm tra xem có thẻ từ nào đang nằm trong vùng quét hay không.
- `uint32_t getUID()`: Đọc 4 byte mã UID cơ bản của thẻ.
- `void haltCard()`: Cho thẻ vào trạng thái nghỉ (Rất quan trọng: giúp tránh việc đọc liên tục gây nghẽn bus I2C).
- `void setAuthKey(const uint8_t key[6]) / uint8_t authenticateKeyA(uint8_t blockAddr)`: Xác thực Key A (thường là mảng 6 byte `0xFF`) để lấy quyền đọc/ghi vào Block.
- `uint8_t readBlock(uint8_t blockAddr, uint8_t *buffer)`: Đọc 16 bytes dữ liệu từ Block (trả về 0 nếu thành công).
- `uint8_t writeBlock(uint8_t blockAddr, const uint8_t *buffer)`: Ghi 16 bytes dữ liệu vào Block.

5. API Quản trị (Advanced APIs)

Dành riêng cho lớp `MKE_I2C_RFID_Advanced`. Các lệnh này sẽ ghi thẳng vào bộ nhớ EEPROM của vi điều khiển trên mạch.

⚠ LƯU Ý QUAN TRỌNG: Chỉ kết nối 1 module duy nhất trên bus I2C khi sử dụng các lệnh này để tránh việc thay đổi nhầm cài đặt của các thiết bị khác.

- `void unlockAdminMode(uint32_t password = 0xA5A5A5A5)`: Mở khóa quyền Quản trị viên để ghi thông số hệ thống.
- `void setI2CAddress(uint8_t newAddress)`: Đổi địa chỉ I2C phần cứng (Yêu cầu gọi `unlockAdminMode` trước đó).
- `void factoryReset()`: Khôi phục cài đặt gốc của module (Đưa địa chỉ I2C về mặc định là `0x29`).

6. Các mã mẫu (Examples) & Kiến thức nhận được

Thư viện đi kèm với một hệ thống các ví dụ từ cơ bản đến nâng cao. Qua quá trình thực hành, bạn sẽ nắm bắt được các kiến thức sau:

1. **01_Read_UID**: Hướng dẫn phát hiện thẻ và đọc mã định danh UID.
 - *Kiến thức*: Hiểu cách thẻ từ hoạt động, cách đọc mã UID để phân biệt các thẻ.
2. **02_DumpInfo**: Đọc và hiển thị toàn bộ thông tin nội dung các Sector/Block trên thẻ.
 - *Kiến thức*: Nắm được cấu trúc bộ nhớ thẻ MIFARE toàn diện, cơ chế xác thực bảo mật bằng Key A/B.
3. **03_Read_Write_Personal_Data**: Đọc và ghi dữ liệu cá nhân vào một Block cụ thể.
 - *Kiến thức*: Biết cách ghi dữ liệu ứng dụng (ví dụ: tên nhân viên) vào thẻ an toàn.
4. **04_Digital_Wallet**: Ví dụ nâng cao mô phỏng một ví điện tử.
 - *Kiến thức*: Cách thao tác toán học trực tiếp trên Block (Nạp tiền, trừ tiền thẻ xe, lưu trữ giá trị).
5. **05_Change_I2C_Address**: Đổi địa chỉ I2C bằng phần mềm.
 - *Kiến thức*: Hiểu cơ chế phân quyền (Admin Mode) và thao tác ghi đè cấu hình phần cứng.
6. **06_Reset_Factory**: Khởi phục cài đặt gốc.
 - *Kiến thức*: Cách trigger lệnh reset hệ thống cho module Slave từ xa.
7. **07_I2C_Multiple_Device**: Giao tiếp với nhiều module RFID cùng lúc để đọc UID.
 - *Kiến thức*: Khai thác sức mạnh của bus I2C, cách điều khiển độc lập nhiều thiết bị không bị xung đột.
8. **08_Compare_UID**: Đọc và so sánh UID thẻ với UID cho trước để cấp quyền truy cập.
 - *Kiến thức*: Logic cơ bản nhất của mọi hệ thống kiểm soát cửa từ, thẻ xe, máy quét thẻ nhân viên.

7. Dành cho Developer (Lập trình I2C cấp thấp)

Nếu bạn lập trình trên các nền tảng khác (MicroPython, ESP-IDF, Raspberry Pi) mà không có sẵn thư viện, đây là chi tiết giao thức I2C của mạch:

A. Giao thức cơ bản (Gửi 7 bytes, Nhận 5 bytes): Hầu hết các lệnh cơ bản đều tuân theo chuẩn gửi 7 bytes và nhận 5 bytes.

- **Gửi**: `[I2C_Addr, Mode_ID, Val0, Val1, Val2, Val3, Checksum]` ($Checksum = I2C_Addr \wedge Mode_ID \wedge Val0 \wedge Val1 \wedge Val2 \wedge Val3$)
- **Nhận (Sau 5-10ms)**: `[Mode_Echo, Data0, Data1, Data2, Data3]`
- **Ví dụ**:
 - **Mode 55 (0x37)**: Kiểm tra có thẻ hay không (Data3 trả về 1 là có thẻ).
 - **Mode 50 (0x32)**: Lấy 4 byte UID của thẻ (nằm trong Data0..3).

B. Quy trình làm việc với thẻ MIFARE (Nâng cao): Do bộ nhớ thẻ MIFARE chia thành các Block 16-byte và mã bảo mật (Key) là 6-byte, giao thức I2C được thiết kế đặc biệt để truyền vớ mảnh:

1. Xác thực Key A (6 bytes): Gửi 3 lệnh 7-byte liên tiếp, cách nhau 50ms:

- **Mode 72 (0x48)**: Set 4 byte đầu của Key vào hệ thống.
- **Mode 73 (0x49)**: Set 2 byte cuối của Key vào hệ thống.
- **Mode 70 (0x46)**: Gửi lệnh Auth Key A với **Val0** = Block Address. Lệnh này sẽ thực thi việc xác thực dựa trên 6 byte Key đã lưu tạm ở trên. Trả về **0** (STATUS_OK) nếu đúng Key.

2. Đọc Block (Read 16 bytes):

- **Bước 1:** Gửi lệnh **Mode 75 (0x4B)** (chuẩn 7-byte) với **Val0** = Block Address.
- **Bước 2:** Đợi 50ms, sau đó thực hiện lệnh I2C Read **18 bytes** liên tục.
- **Dữ liệu nhận:** [Mode_Echo (75), Data0, Data1 ... Data15, Checksum].

3. Ghi Block (Write 16 bytes): Gửi 5 lệnh 7-byte liên tiếp, cách nhau 50ms:

- **Mode 76 (0x4C):** Gửi 4 byte Data cao nhất (Byte 0-3).
- **Mode 77 (0x4D):** Gửi 4 byte tiếp theo (Byte 4-7).
- **Mode 78 (0x4E):** Gửi 4 byte tiếp theo (Byte 8-11).
- **Mode 79 (0x4F):** Gửi 4 byte thấp nhất (Byte 12-15).
- **Mode 80 (0x50):** Lệnh thực thi Ghi với **Val0** = Block Address. Trả về 0 nếu ghi thành công.

*Lưu ý: Luôn nhớ sử dụng **Mode 100 (Halt Card)** sau khi thao tác xong để tránh việc module đọc liên tục làm nghẽn đường truyền.*