RFID

檢查 Raspberry Pi 5 上 SPI 功能的逐步指南

1. 檢查 `spidev0` 是否正常運作。
   - 打開終端機。
   - 執行以下指令： `ls -l /dev/spidev0*`。
   - 檢查檔案 `/dev/spidev0.0` 和 `/dev/spidev0.1` 是否已列出。
2. 檢查引腳配置。
   - 在終端機中執行以下命令： `pinctrl | head -n 14`。
   - 驗證 `GPIO9`、`GPIO10` 和 `GPIO11` 是否已列為 `SPI0_MISO`、`SPI0_MOSI` 和 `SPI0_SCLK`。
3. `SPI` 環回測試
   - 將 `SPI0_MISO`（接腳 `21`）實際連接到 `SPI0_MOSI`（接腳 `19`）。
   - 開啟 `Python`：`python3`
   - 運行以下 `Python` 程式碼：

     ```
     import spidev spi = spidev.SpiDev()
     spi.open(0,0)
     print(spi.xfer([8,4,2,1,0]))
     print(spi.xfer([8,4,2,1,0]))
     ```

     傳回第二次回傳。
4. `CE0` 測試
   - 將 `SPI0_MISO`（接腳 `21`）連接到 `SPI0_CE0`（接腳 `24`）。
   - 在 `Python` 中：

     ```
     spi1 = spidev.SpiDev()
     spi1.open(0,1)
     print(spi.xfer([8,4,2,1,0]))
     print(spi1.xfer([8,4,2,1,0]))
     ```

     `spi.xfer` 應返回 `[0, 0, 0, 0, 0]` 且 `spi1.xfer` 應返回 `[255, 255, 255, 255, 255]`。
     如果所有步驟都成功，則 `SPI` 在您的 `Raspberry Pi 5` 上正常運作。

您： `pinctrl | head -n 14`

檢查 `GPIO9`、`GPIO10` 和 `GPIO11` 是否列為 `SPI0_MISO`、`SPI0_MOSI` 和 `SPI0_SCLK`。

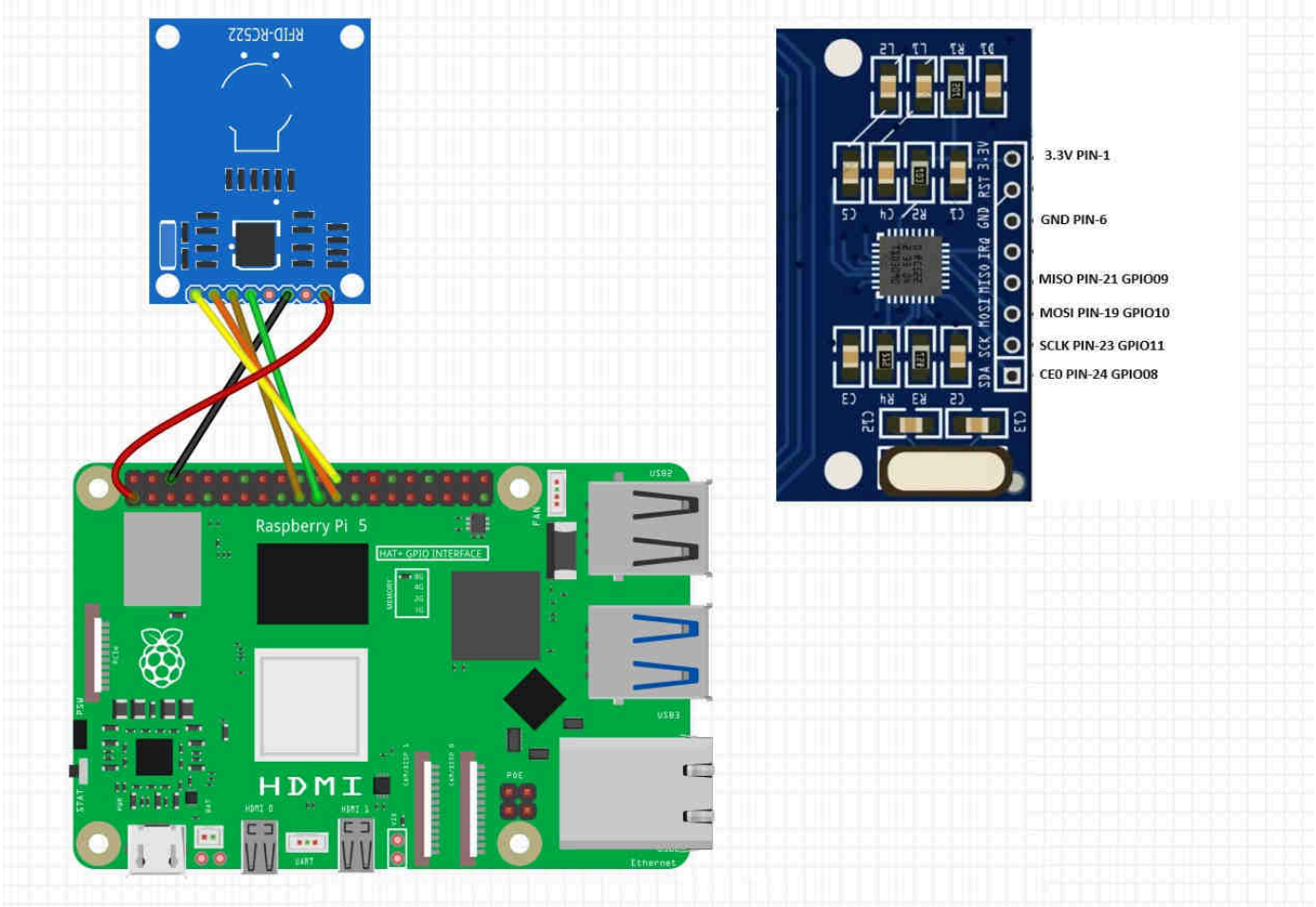
9：ip pn | lo // GPIO9 = 輸入

10：ip pn | lo // GPIO10 = 輸入

11：ip pn | lo // GPIO11 = 輸入

`Raspberry Pi 5` 上安裝 `RC522 RFID` 讀卡機所需的硬體連接、驅動與軟體安裝詳細步驟說明

1. 硬體連接（`RC522` 與 `Raspberry Pi 5`）



RC522 RFID 模組使用 SPI 介面與樹莓派連接，請依下表將 RC522 的腳位對應接到樹莓派 5 的 GPIO：

RC522 Pin	樹莓派 Pin	樹莓派 GPIO 功能說明	顏色
SDA (NSS)	24	GPIO8 (SPI0_CE0_N)	白

RC522 Pin	樹莓派 Pin	樹莓派 GPIO 功能說明	顏色
SCK	23	GPIO11 (SPI0_SCLK)	灰
MOSI	19	GPIO10 (SPI0_MOSI)	紫
MISO	21	GPIO9 (SPI0_MISO)	藍
IRQ	None	None	綠
GND	Any Ground	Any GND	黑
RST	22	GPIO25	黃
3.3V	1	3.3V	紅

> 注意： 請勿將 RC522 的 VCC 接到 5V，否則可能損壞模組！

2. 啟用 SPI 介面

   GPIO 引腳不僅僅是輸入/輸出，透過切換一些設置，您還可以使用 SPI、I2C、UART、PWM、單線等周邊。

   對於使用 MFRC522 讀卡機的項目，您需要能夠存取 SPI 設備。使用指令 raspi-config，您可以啟用 SPI0 設備，即 GPIO10 (MOSI)、GPIO9 (MISO)、GPIO11 (CLK)、GPIO8 (CE0) 和 GPIO7 (CE1)。您也可以透過修改 /boot/firmware/config.txt 來啟用其他 SPI。

   `sudo nano /boot/firmware/config.txt`

   ```
   :
   # 最後加上以下內容
   # RFID
   dtparam=spi=on
   dtoverlay=spi-bcm2708
   dtoverlay=spi0-hw-cs
   ```

   - 開啟終端機，執行：

     `sudo raspi-config`
   - 選擇 Interface Options → SPI → 啟用（Enable）。
   - 重新啟動樹莓派：

     `sudo reboot`
   - 開機後，檢查 SPI 是否啟用：

```
lsmod | grep spi
```

若有 `spi_bcm2835` 代表啟用成功。
啟用 `SPI` 並重新啟動後，您會注意到存在一個新的裝置 `spidev`。
`ls -l /dev/spi*`

```
crw------- 1 root root 153, 0 2023-09-11 10:00 /dev/spidev0.0
crw------- 1 root root 153, 1 2023-09-11 10:00 /dev/spidev0.1
```

`spidev0.0` 代表片選 `0`，`spidev0.1` 代表片選 `1`。
然後只需將 `MFRC522` 連接到正確的 `SPI` 引腳即可。剩下的就是使用 `spidev` 介面來存取它！
無需處理任何 `GPIO` 輸入/輸出。這樣就完全不需要 `gpiozero` 了。

3. 安裝 `Python` 相關驅動與函式庫
   i. 安裝 `Python` 開發工具與 `pip`

```
sudo apt-get update
sudo apt-get install -y python3-dev python3-pip
```

   ii. 安裝 `SPI-Py`（新版 `Raspberry Pi OS` 通常已內建 `SPI` 支援，可略過）
   iii. 安裝 `MFRC522 Python` 函式庫
   目前推薦使用 `pimylifeup/MFRC522-python` 或 `mfrc522`（`pip` 版本，較新且支援 `Python 3`）。
   使用 `pip` 安裝（建議）
   `pip3 install mfrc522`
   或手動安裝 `GitHub` 版本

```
cd ~/python
git clone https://github.com/danjperron/MFRC522-python
cd MFRC522-python
```

> 若能正確顯示卡號，表示安裝成功。

4. 測試 `RFID` 讀卡功能
   `python read.py`

```
#!/usr/bin/env python3
# 指定此腳本使用 Python 3 解譯器執行
# -*- coding: utf8 -*-
# 指定此檔案的編碼為 UTF-8，支援中文等多國語言
# 匯入 MFRC522 模組（用於操作 RFID 讀卡機）與 signal 模組（用於處理中斷訊號）。
import MFRC522
import signal

#定義一個全域變數 continue_reading，用來控制主循環是否繼續執行。
continue_reading = True



# function to read uid an conver it to a string
# 定義 uidToString 函式，將 UID（卡片唯一識別碼，通常為一組整數）轉換為字串。這裡是將每個數
def uidToString(uid):
    mystring = ""
    for i in uid:
        mystring = format(i, '02X') + mystring
    return mystring



# Capture SIGINT for cleanup when the script is aborted
# 定義 end_read 函式，當收到中斷訊號（如 Ctrl+C）時執行。會印出提示訊息並將 continue_reading 設
def end_read(signal, frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False

# Hook the SIGINT
% 註冊 SIGINT（中斷訊號，通常是 Ctrl+C）時要執行的處理函式為 end_read。
signal.signal(signal.SIGINT, end_read)

# 建立一個 MFRC522 類別的物件 MIFAREReader，用來與 RFID 讀卡機溝通。
# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# Welcome message
# 印出歡迎訊息與操作提示。
```

```
print("Welcome to the MFRC522 data read example")
print("Press Ctrl-C to stop.")

# This loop keeps checking for chips.
# If one is near it will get the UID and authenticate
# 主循環，只要 continue_reading 為 True 就會持續執行。用來不斷檢查是否有卡片靠近。
while continue_reading:

    # Scan for cards
    (status, TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    # 呼叫 MFRC522_Request 方法掃描是否有卡片靠近，回傳狀態與卡片類型。
    if status == MIFAREReader.MI_OK:
        print ("Card detected")

        # Get the UID of the card
        # 如果有偵測到卡片（狀態為 MI_OK），印出「Card detected」。
        (status, uid) = MIFAREReader.MFRC522_SelectTagSN()
        # If we have the UID, continue
        # 如果成功取得 UID（狀態為 MI_OK），則將 UID 轉為字串並印出。否則印出「Authentication err
        # 呼叫 MFRC522_SelectTagSN 方法取得卡片的 UID。
        if status == MIFAREReader.MI_OK:
            print("Card read UID: %s" % uidToString(uid))
        else:
            print("Authentication error")
```

案例

1. RFID + GPIO 控制 LED ON/OFF

1. 功能說明
   讀取 RFID 卡號，卡號的 UID 必須在授權清單內，且區塊授權成功，才亮 LED0(綠燈)；否則亮 LED1(紅燈)。
2. 硬體連接
3. 程式碼

```python
#!/usr/bin/env python3
# -*- coding: utf8 -*-
"""
   使用 Python 的 GPIO Zero 模組控制 LED，並讀取 RFID 卡片。
   適用於 Raspberry Pi 5 (Bookworm OS)
"""
import time  # 匯入時間模組，用於延遲與計時
from gpiozero import LED  # 匯入 gpiozero 的 LED 類別，用於控制樹莓派的 GPIO 腳位
import MFRC522  # 匯入 MFRC522 RFID 讀卡機模組
import signal  # 匯入 signal 模組，用於處理 Ctrl+C 中斷

# 設定 LED 腳位（BCM 編號，請依實際接線調整）
LED0 = LED(4)  # 建立 LED0 物件，連接到 BCM 4 腳位（請依實際接線調整）
LED1 = LED(17)  # 建立 LED1 物件，連接到 BCM 17 腳位（請依實際接線調整）

# 授權 UID 清單，只有在這個清單內的卡片才算授權
# AUTHORIZED_UIDS = ["038C6069"]
AUTHORIZED_UIDS = ["038C6069","86038C6069"]  # 可自行新增多個授權卡號

continue_reading = True  # 控制主迴圈是否繼續執行的旗標

# 將 UID 陣列轉換為字串，方便比對與顯示
# 例如 [0x03, 0x8C, 0x60, 0x69] -> '038C6069'
def uidToString(uid):
    mystring = ""  # 初始化空字串
    for i in uid:
        mystring = format(i, '02X') + mystring  # 將每個 byte 轉為兩位大寫 16 進位字串，並前置串接
    return mystring  # 回傳組合後的 UID 字串

# 當按下 Ctrl+C 時執行，負責結束主程式並關閉 LED
# signal: 訊號編號, frame: 當前堆疊幀
def end_read(signal, frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")  # 顯示結束訊息
    continue_reading = False  # 結束主迴圈
    LED0.off()  # 關閉 LED0
    LED1.off()  # 關閉 LED1
```

```
# 註冊 SIGINT (Ctrl+C) 的處理函式
signal.signal(signal.SIGINT, end_read)

# 建立 MFRC522 物件，初始化 RFID 讀卡機
MIFAREReader = MFRC522.MFRC522()

print("歡迎使用 MFRC522 讀卡範例 (GPIO Zero)")  # 顯示歡迎訊息
print("按 Ctrl-C 結束程式.")  # 提示如何結束

# 預設金鑰，MIFARE 卡片預設為 6 個 0xFF
key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

# 主迴圈，持續偵測是否有卡片靠近
while continue_reading:
    # 掃描是否有卡片靠近讀卡機
    (status, TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    if status == MIFAREReader.MI_OK:  # 若偵測到卡片
        print("偵測到卡片")  # 顯示偵測到卡片
        # 取得卡片的 UID
        (status, uid) = MIFAREReader.MFRC522_Anticoll1()
        if status == MIFAREReader.MI_OK:  # 若成功取得 UID
            uid_str = uidToString(uid)  # 將 UID 轉為字串
            print("卡片 UID: %s" % uid_str)  # 顯示卡片 UID
            if uid_str in AUTHORIZED_UIDS:  # 若 UID 在授權清單內
                # 選取卡片，準備進行授權
                MIFAREReader.MFRC522_PcdSelect1(uid)
                # 嘗試對區塊 8 進行授權
                status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
                if status == MIFAREReader.MI_OK:  # 若授權成功
                    MIFAREReader.MFRC522_Read(8)  # 讀取區塊 8 的資料
                    MIFAREReader.MFRC522_StopCrypto1()  # 停止加密通訊
                    print("授權通過！")  # 顯示授權通過
                    LED0.on()  # 亮起 LED0 表示授權成功
                    time.sleep(0.5)  # 保持 0.5 秒
                    LED0.off()  # 關閉 LED0
                else:
                    print("區塊授權失敗")  # 顯示授權失敗
```

```
                LED1.on()  # 亮起 LED1 表示授權失敗
                time.sleep(0.5)  # 保持 0.5 秒
                LED1.off()  # 關閉 LED1
            else:
                print("UID 不在授權清單")  # 顯示未授權
                LED1.on()  # 亮起 LED1 表示未授權
                time.sleep(0.5)  # 保持 0.5 秒
                LED1.off()  # 關閉 LED1
        else:
            print("讀取 UID 失敗")  # 顯示讀取 UID 失敗
    time.sleep(0.1)  # 每次循環間隔 0.1 秒，避免 CPU 過度佔用
```

2. RFID + GPIO 控制電磁鎖

1. 功能說明

   當卡號 `UID` 在授權清單內且區塊授權成功時，會：

   - 亮起 `LED0`
   - 開啟繼電器（腳位 `18`，持續 `10` 秒，控制電磁鎖）
   - 0.5 秒後關閉 LED0
   - 程式結束時會自動關閉繼電器並清理 GPIO 資源
2. 硬體連接
3. 程式碼

```python
#!/usr/bin/env python3
# -*- coding: utf8 -*-
"""
   使用 Python 的 GPIO Zero 模組控制 LED，並讀取 RFID 卡片。
   適用於 Raspberry Pi 5 (Bookworm OS)
"""
import time  # 匯入時間模組，用於延遲與計時
from gpiozero import LED  # 匯入 gpiozero 的 LED 類別，用於控制樹莓派的 GPIO 腳位
import MFRC522  # 匯入 MFRC522 RFID 讀卡機模組
import signal  # 匯入 signal 模組，用於處理 Ctrl+C 中斷
from relay_control import RelayControl  # 匯入繼電器控制類別

# 設定 LED 腳位（BCM 編號，請依實際接線調整）
LED0 = LED(4)  # 建立 LED0 物件，連接到 BCM 4 腳位（請依實際接線調整）
LED1 = LED(17)  # 建立 LED1 物件，連接到 BCM 17 腳位（請依實際接線調整）

# 設定繼電器腳位（預設 18，請依實際接線調整）
RELAY_PIN = 18
relay = RelayControl(relay_pin=RELAY_PIN)

# 授權 UID 清單，只有在這個清單內的卡片才算授權
# AUTHORIZED_UIDS = ["038C6069"]
AUTHORIZED_UIDS = ["038C6069","86038C6069"]  # 可自行新增多個授權卡號

continue_reading = True  # 控制主迴圈是否繼續執行的旗標

# 將 UID 陣列轉換為字串，方便比對與顯示
# 例如 [0x03, 0x8C, 0x60, 0x69] -> '038C6069'
def uidToString(uid):
    mystring = ""  # 初始化空字串
    for i in uid:
        mystring = format(i, '02X') + mystring  # 將每個 byte 轉為兩位大寫 16 進位字串，並前置串接
    return mystring  # 回傳組合後的 UID 字串

# 當按下 Ctrl+C 時執行，負責結束主程式並關閉 LED
# signal: 訊號編號, frame: 當前堆疊幀
def end_read(signal, frame):
    global continue_reading
```

```
    print("Ctrl+C captured, ending read.")  # 顯示結束訊息
    continue_reading = False  # 結束主迴圈
    LED0.off()  # 關閉 LED0
    LED1.off()  # 關閉 LED1
    relay.relay_off()  # 關閉繼電器
    relay.cleanup()    # 清理 GPIO 資源

# 註冊 SIGINT (Ctrl+C) 的處理函式
signal.signal(signal.SIGINT, end_read)

# 建立 MFRC522 物件，初始化 RFID 讀卡機
MIFAREReader = MFRC522.MFRC522()

print("歡迎使用 MFRC522 讀卡範例 (GPIO Zero)")  # 顯示歡迎訊息
print("按 Ctrl-C 結束程式.")  # 提示如何結束

# 預設金鑰，MIFARE 卡片預設為 6 個 0xFF
key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

# 主迴圈，持續偵測是否有卡片靠近
while continue_reading:
    # 掃描是否有卡片靠近讀卡機
    (status, TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    if status == MIFAREReader.MI_OK:  # 若偵測到卡片
        print("偵測到卡片")  # 顯示偵測到卡片
        # 取得卡片的 UID
        (status, uid) = MIFAREReader.MFRC522_Anticoll1()
        if status == MIFAREReader.MI_OK:  # 若成功取得 UID
            uid_str = uidToString(uid)  # 將 UID 轉為字串
            print("卡片 UID: %s" % uid_str)  # 顯示卡片 UID
            if uid_str in AUTHORIZED_UIDS:  # 若 UID 在授權清單內
                # 選取卡片，準備進行授權
                MIFAREReader.MFRC522_PcdSelect1(uid)
                # 嘗試對區塊 8 進行授權
                status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
                if status == MIFAREReader.MI_OK:  # 若授權成功
                    MIFAREReader.MFRC522_Read(8)  # 讀取區塊 8 的資料
```

```
                MIFAREReader.MFRC522_StopCrypto1()  # 停止加密通訊
                print("授權通過！")  # 顯示授權通過
                LED0.on()  # 亮起 LED0 表示授權成功
                relay.relay_on(duration=10)  # 開啟繼電器 10 秒，開門
                time.sleep(0.5)  # 保持 0.5 秒
                LED0.off()  # 關閉 LED0
            else:
                print("區塊授權失敗")  # 顯示授權失敗
                LED1.on()  # 亮起 LED1 表示授權失敗
                time.sleep(0.5)  # 保持 0.5 秒
                LED1.off()  # 關閉 LED1
        else:
            print("UID 不在授權清單")  # 顯示未授權
            LED1.on()  # 亮起 LED1 表示未授權
            time.sleep(0.5)  # 保持 0.5 秒
            LED1.off()  # 關閉 LED1
    else:
        print("讀取 UID 失敗")  # 顯示讀取 UID 失敗
time.sleep(0.1)  # 每次循環間隔 0.1 秒，避免 CPU 過度佔用
```

4. 測試

參考資源

1. MFRC522 文檔
2. SPI 文檔