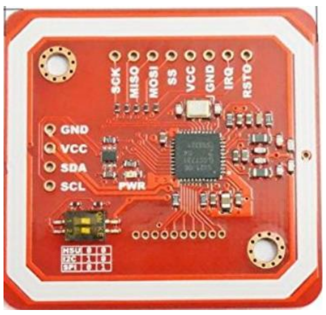
RFID(Radio Frequency Identification)

1. PN532 NFC RFID V3 近場通信模組



Ref. https://www.taiwansensor.com.tw/product/pn532-nfc-rfid-v3-%e8%bf%91%e5%a0%b4%e9%80%9a%e4%bf%a1%e6%a8%a1%e7%b5%84/

NFC(Near Field Communication)又稱近距離無線通信，是一種短距離的高頻無線通信技術，允許電子設備之間進行非接觸式點對點數據傳輸（在十厘米內）交換數據。這個技術由免接觸式射頻識別（RFID）演變而來，並向下兼容 RFID，最早由 Sony 和 Philips 各自開發成功，主要用於手機等手持設備中提供 M2M(Machine to Machine)的通信。由於近場通訊具有天然的安全性，因此，NFC 技術被認為在手機支付等領域具有很大的應用前景。

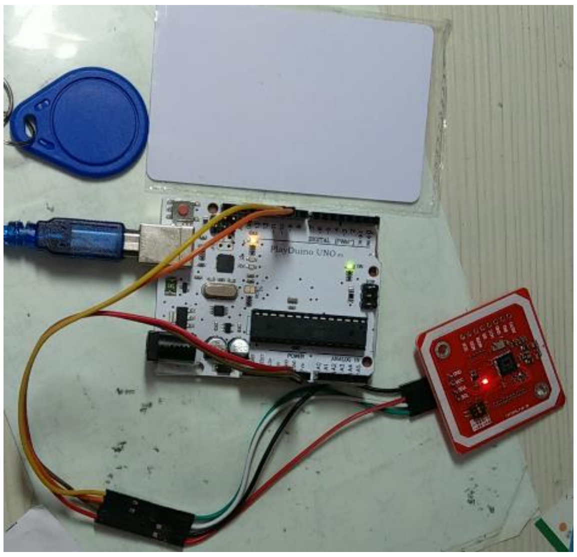
NFC 是近年來的一個熱門技術。我們經常聽到這個工作雖然智慧手機如 HTC 公司三星或介紹他們的最新高端手機。幾乎所有的高端手機市場支持 NFC。近距離通信（NFC）是一組用於智慧手機和類似的設備建立無線通訊與互相碰在一起或是帶他們到接近標準，通常不超過幾釐米。本模組圍繞 NXP PN532。恩智浦 PN532 是 NFC 受歡迎的模組。

- 支援 I2C、SPI、**HSU**(高速 UART)，可以很容易在這些通訊方式之間進行切換，通過板子左下角的撥動開關即可輕鬆實現通訊方式的切換
- Work in NFC Mode or RFID reader/writer Mode
- RFID reader/writer supports: （RDID 讀寫支援）
  Mifare 1k, 4k, Ultralight, 和 DesFire 卡
  ISO/IEC 14443-4 cards，如 CD97BX, CD light, Desfire, P5CN072 (SMX)
  Innovision Jewel cards，如 IRT5001 card
  FeliCa cards ，如 RCS_860 和 RCS_854
- 即插即用,相容 Arduino
- 板內天線，支援 5cm～7cm 通訊距離
- On-board level shifter, 標準 5V TTL for I2C and UART, 3.3V TTL for SPI
- 可工作在 RFID 讀/寫模式
- 可工作在 1443-A card 或者 虛擬 card 模式
- 可與其他 NFC 設備交換數據，如 Android 手機
- Interface 數據埠 I2C 是默認數據埠. 用戶也可根據自己需要利用引出的管腳改變數據傳輸方式，如串埠，SPI 等

Testing the hardware

Because the default communication mode/interface of the module is HSU, let’s continue without bearing on the selector switch right now. Just take a set of breadboard jumper wires and follow the hardware setup as shown in the below table:

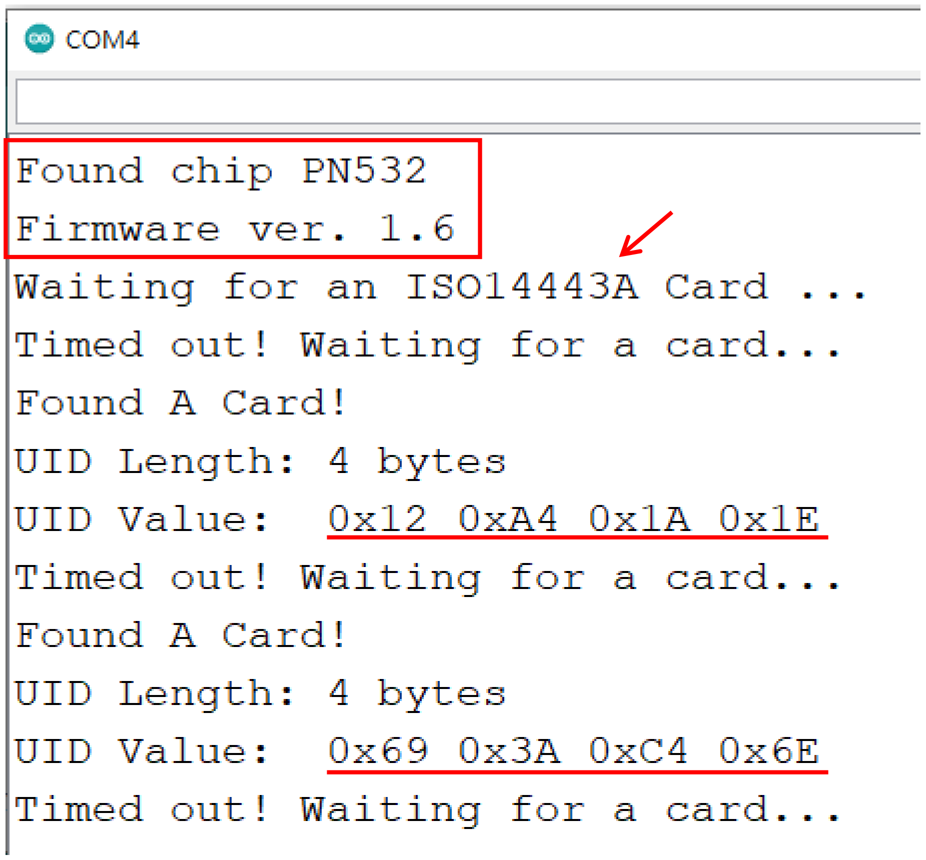
PN532 NFC RFID MODULE	**ARDUINO UNO**
VCC	5V
GND	GND
RX(SCL)	D11
TX(SDA)	D10



Ex.1 Get UID

```
#include <SoftwareSerial.h>
#include <PN532_SWHSU.h>
#include <PN532.h>
SoftwareSerial SWSerial( 10, 11 ); // RX, TX
PN532_SWHSU pn532swhsu( SWSerial );
PN532 nfc( pn532swhsu );

void setup(void) {
  Serial.begin(9600);
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Didn't Find PN53x Module");
    while (1); // Halt
  }
  // Got valid data, print it out!
  Serial.print("Found chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print("Firmware ver. "); Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);
  // Configure board to read RFID tags
  nfc.SAMConfig();
  Serial.println("Waiting for an ISO14443A Card ...");
}
boolean success;
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };// Buffer to store the returned UID
uint8_t i,k,uidLength;    // Length of the UID (4 or 7 bytes depending on ISO14443A card type)
void loop(void) {
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);
  if (success) {
    Serial.println("Found A Card!");
    Serial.print("UID Length: ");
    Serial.print(uidLength, DEC);
    Serial.println(" bytes");
    Serial.print("UID Value: ");
```

```
        for (i=0; i < uidLength; i++)
        {
           Serial.print(" 0x");
           Serial.print(uid[i], HEX);
        }
        Serial.println();
        k=0;
        // 2 second halt
        delay(2000);
     }
     else if (k==0)
     { // PN532 probably timed out waiting for a card
        Serial.println("Timed out! Waiting for a card...");
        k++;
     }
}
```

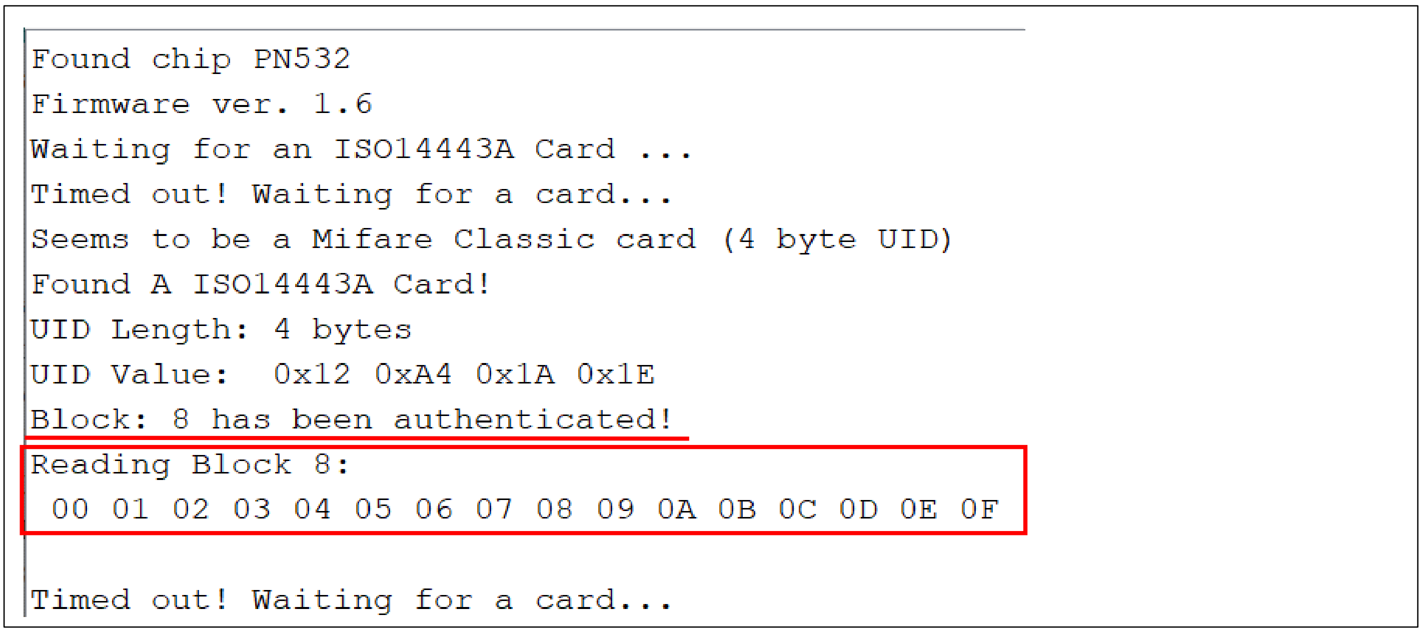


COM4

```
Found chip PN532
Firmware ver. 1.6
Waiting for an ISO14443A Card ...
Timed out! Waiting for a card...
Found A Card!
UID Length: 4 bytes
UID Value:  0x12 0xA4 0x1A 0x1E
Timed out! Waiting for a card...
Found A Card!
UID Length: 4 bytes
UID Value:  0x69 0x3A 0xC4 0x6E
Timed out! Waiting for a card...
```

Ex.2 Read the assigned block

```
#include <SoftwareSerial.h>
#include <PN532_SWHSU.h>
#include <PN532.h>
SoftwareSerial SWSerial( 10, 11 ); // RX, TX
PN532_SWHSU pn532swhsu( SWSerial );
PN532 nfc( pn532swhsu );

void setup(void) {
  Serial.begin(9600);
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Didn't Find PN53x Module");
    while (1); // Halt
  }
  // Got valid data, print it out!
  Serial.print("Found chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print("Firmware ver. "); Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);
  // Configure board to read RFID tags
  nfc.SAMConfig();
  Serial.println("Waiting for an ISO14443A Card ...");
}
boolean success;
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };// Buffer to store the returned UID
uint8_t i,k,uidLength, iBlockNumber, keyNumber=0;//Length of the UID(4 or 7
bytes depending on ISO14443A card type)
uint8_t keyA[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
//keyNumber: Which key type to use during authentication (0 =
MIFARE_CMD_AUTH_A, 1 = MIFARE_CMD_AUTH_B)
uint8_t data[16];
void loop() {
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);
  if (success) {// Display some basic information about the card
    if (uidLength == 4)
      Serial.println("Seems to be a Mifare Classic card (4 byte UID)");
    Serial.println("Found A ISO14443A Card!");
    Serial.print("UID Length: ");
    Serial.print(uidLength, DEC);
    Serial.println(" bytes");
    Serial.print("UID Value: ");
    for (i=0; i < uidLength; i++)
    {
      Serial.print(" 0x");
      Serial.print(uid[i], HEX);
    }
    Serial.println();
    // Now we need to try to authenticate it for read/write access
    // Try with the factory default KeyA: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
       // to leave it alone unless you know what you're doing
    delay(1500);
    iBlockNumber=8;
    //uint8_t keyA[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
    success = nfc.mifareclassic_AuthenticateBlock(uid, uidLength, iBlockNumber,
keyNumber, keyA);
    if (success)
    {
        Serial.print(F("Block: "));
        Serial.print(iBlockNumber);
        Serial.println(F(" has been authenticated!"));
        delay(1500);
        k=0;
        do{// Try to read the contents of block
          success = nfc.mifareclassic_ReadDataBlock(iBlockNumber, data);
          if (success){// Data seems to have been read ... spit it out
            Serial.print(F("Reading Block "));
            Serial.print(iBlockNumber);
            Serial.println(":");
            nfc.PrintHexChar(data, 16);
            Serial.println();
            k=0;
            break;
          }
          else
          {
            delay(1500);
            k++;
          }
        }while( k< 10);
        if (!success)
          Serial.println("Ooops ... unable to read the requested block.  Try
another key?\n\n");
    }
    else
      Serial.println("Ooops ... authentication failed: Try another key?\n\n");
    delay(2000);
  }
  else if (k==0)
  { // PN532 probably timed out waiting for a card
    Serial.println("Timed out! Waiting for a card...");
    k++;
  }
}
```



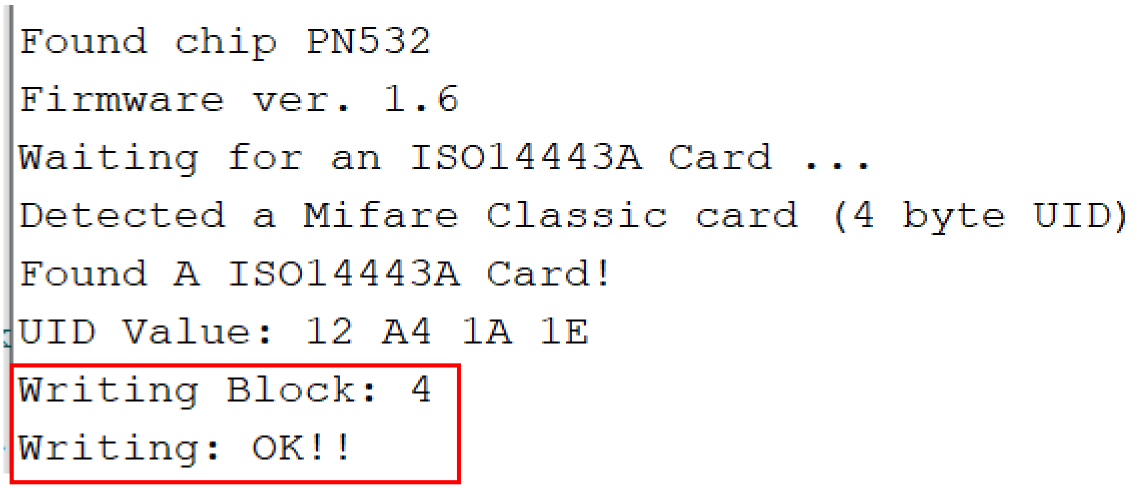
```
Found chip PN532
Firmware ver. 1.6
Waiting for an ISO14443A Card ...
Timed out! Waiting for a card...
Seems to be a Mifare Classic card (4 byte UID)
Found A ISO14443A Card!
UID Length: 4 bytes
UID Value:  0x12 0xA4 0x1A 0x1E
Block: 8 has been authenticated!
Reading Block 8:
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Timed out! Waiting for a card...
```

Ex.3 Writing data into the assigned block

```
#include <SoftwareSerial.h>
#include <PN532_SWHSU.h>
#include <PN532.h>
SoftwareSerial SWSerial( 10, 11 ); // RX, TX
PN532_SWHSU pn532swhsu( SWSerial );
PN532 nfc( pn532swhsu );

boolean success;
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };// Buffer to store the returned UID
uint8_t i,k,uidLength;//Length of the UID(4 or 7 bytes depending on ISO14443A
card type)
uint8_t keyA[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
uint8_t iBlockNumber, keyType=0, assignBlockNum;
//keyType: Which key type to use during authentication (0 = MIFARE_CMD_AUTH_A,
1 = MIFARE_CMD_AUTH_B)
uint8_t data[16]={15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0};
int iRes;
void setup(void) {
  Serial.begin(9600);
  //Serial.println("Hello Maker!");
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Didn't Find PN53x Module");
    while (1); // Halt
  }
  // Got valid data, print it out!
  Serial.print(F("Found chip PN5")); Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print(F("Firmware ver. ")); Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);
  // Configure board to read RFID tags
  nfc.SAMConfig();
  Serial.println(F("Waiting for an ISO14443A Card ..."));
  k=255;
}

boolean getUID()
{
  boolean bRes = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength);
  return bRes;
}

boolean readUID()
{
  boolean bRes = getUID();
  if (bRes)
  {
    if (uidLength == 4)
      Serial.println("Detected a Mifare Classic card (4 byte UID)");
    if (uidLength == 7)
      Serial.println("Detected a Mifare Ultralight tag (7 byte UID)");
    Serial.println("Found A ISO14443A Card!");
    Serial.print("UID Value: ");
    printHex(uid, uidLength);
  }
  delay(1000);
  return bRes;
}

int getBlock(uint8_t iBlckNum, uint8_t *buf)
{
  boolean bOK;
  bOK = nfc.mifareclassic_AuthenticateBlock(uid, uidLength, iBlckNum, keyType, keyA);
  if (!bOK)
    return -1;
  //read block
  delay(1500);
  k=0;
  do{// Try to read the contents of block
    bOK = nfc.mifareclassic_ReadDataBlock(iBlckNum, buf);
    if (bOK)
      break;
    else
    {
      delay(1000);
      k++;
    }
  }while( k< 10);
  if (bOK)
    return 0;
  else
    return -1;
}
int readBlock(uint8_t blckNum)
{
  int iRes=getBlock(blckNum, data);
  if (iRes != -1)
  {
    Serial.print(F("Block: "));
    Serial.print(iBlockNumber);
    Serial.println(F(" has been authenticated!"));
    if (iRes == 0)
    {
      Serial.print(F("Reading Block "));
      Serial.print(blckNum);
      Serial.println(":");
      printHex(data, 16);
    }
    else
      Serial.println("Unable to read the requested block.  Try another
key?\n\n");
  }
  else
    Serial.println("Authentication failed: Try another key?\n\n");
  return iRes;
}
```

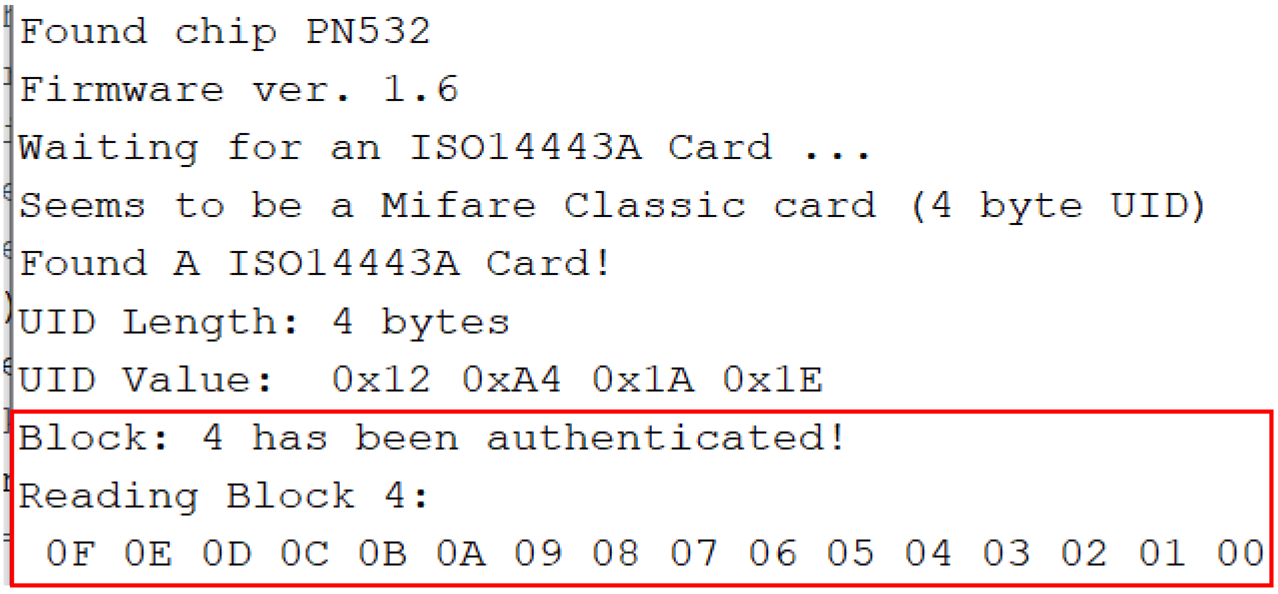
```
int writeBlock(uint8_t iBlckNum, uint8_t *buf)
{
  boolean bOK;
  bOK = nfc.mifareclassic_AuthenticateBlock(uid, uidLength, iBlckNum, keyType, keyA);
  if (!bOK)
    return -1;
  int iRes=nfc.mifareclassic_WriteDataBlock(iBlckNum,buf);
  return iRes;
}

void loop() {
  success =readUID();
  if (success) {
    if (Serial.available() > 0)
    {
      iBlockNumber=Serial.parseInt();
      iRes=readBlock(iBlockNumber);
      if (iRes == 0)
        k=0;
    }
    if (k == 255)
    {
      Serial.print(F("Writing Block: "));
      iBlockNumber=4;
      Serial.println(iBlockNumber);
      iRes=writeBlock(iBlockNumber,data);
      if (iRes == 1)
      {
        Serial.println(F("Writing: OK!!"));
        k=254;
      }
      else if (iRes == -1)
        Serial.println(F("Authentication: Fail!!"));
      else
        Serial.println(F("Writing: Fail!!"));
    }
  }
  else if (k==0)
  { // PN532 probably timed out waiting for a card
    Serial.println(F("Timed out! Waiting for a card..."));
    k++;
  }
  delay(1000);
}

//Helper routine to dump a byte array as hex values to Serial.
void printHex(const uint8_t *buf,const byte bufferSize) {
  char res[bufferSize*3+1];
  for (byte i = 0; i < bufferSize; i++) {
    sprintf(res+i*3,"%02X ", *(buf+i));
  }
  Serial.println(res);
}
//Helper routine to dump a byte array as dec values to Serial.
```

```
void printDec(const uint8_t *buffer, const byte bufferSize) {
  byte i;
  for (i = 0; i < bufferSize-1; i++) {
    Serial.print(buffer[i]);
    Serial.print(", ");
  }
  Serial.println(buffer[i]);
}
void printASCII(const uint8_t *buffer, const byte bufferSize) {
  byte i;
  for (i = 0; i < bufferSize; i++) {
    Serial.write(buffer[i]);
  }
  Serial.println();
}
```



```
Found chip PN532
Firmware ver. 1.6
Waiting for an ISO14443A Card ...
Detected a Mifare Classic card (4 byte UID)
Found A ISO14443A Card!
UID Value: 12 A4 1A 1E
Writing Block: 4
Writing: OK!!
```

驗證 by Ex.2!



```
Found chip PN532
Firmware ver. 1.6
Waiting for an ISO14443A Card ...
Seems to be a Mifare Classic card (4 byte UID)
Found A ISO14443A Card!
UID Length: 4 bytes
UID Value:  0x12 0xA4 0x1A 0x1E
Block: 4 has been authenticated!
Reading Block 4:
 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
```