

# PN512 NFC Reader based on Arduino v1.0



## Table of contents

<b>About</b>	<b>2</b>
<b>Library</b>	<b>3</b>
<b>Hardware</b>	<b>8</b>
<b>Revision history</b>	<b>10</b>

## About

The PN512 NFC Reader is based on the Arduino Nano and shares the same pinouts and capabilities, but it can also read cards and tags with the help of the PN512 chip. It is designed to replace the RFID-RC522 module, so all the functionality of the RFID-MFRC522 and PN512 are the same.

## Library

To use the device you will need a RFID library. We have modified an, already known, library for the MFRC-522 RFID Reader to work with the PN512 chip. The library that we used was miguelbalboa's rfid, <https://github.com/miguelbalboa/rfid>. You can find the PN512 library in the official Arduino Library Manager in the Arduino IDE application, or you can find it on github, [https://github.com/nfc-rfid-reader-sdk/MFRC522\\_PN512/](https://github.com/nfc-rfid-reader-sdk/MFRC522_PN512/).

If you want to do your own changes in the document below are the steps that you need to initialize the PN512's registers to work with the MFRC522 library.

First, in mfrc522.h file you need to add the PN512 self-test firmware reference

```
//PN512 (0x82)
const byte PN512_firmware_reference[] PROGMEM = {
    0x00, 0xEB, 0x66, 0xBA, 0x57, 0xBF, 0x23, 0x95,
    0xD0, 0xE3, 0x0D, 0x3D, 0x27, 0x89, 0x5C, 0xDE,
    0x9D, 0x3B, 0xA7, 0x00, 0x21, 0x5B, 0x89, 0x82,
    0x51, 0x3A, 0xEB, 0x02, 0x0C, 0xA5, 0x00, 0x49,
    0x7C, 0x84, 0x4D, 0xB3, 0xCC, 0xD2, 0x1B, 0x81,
    0x5D, 0x48, 0x76, 0xD5, 0x71, 0x61, 0x21, 0xA9,
    0x86, 0x96, 0x83, 0x38, 0xCF, 0x9D, 0x5B, 0x6D,
    0xDC, 0x15, 0xBA, 0x3E, 0x7D, 0x95, 0x3B, 0x2F
};
```

In the mfrc522.cpp file on line 326. Function PCD\_PerformSelfTest() add

```
case 0x82:    // PN512
reference = PN512_firmware_reference;
break;
```

In the end of the switch case function.

```

64     };
65     // Clone
66     // Fudan Semiconductor FM17522 (0x88)
67     const byte FM17522_firmware_reference[] PROGMEM = {
68         0x00, 0xD6, 0x78, 0x8C, 0xE2, 0xAA, 0x0C, 0x18,
69         0x2A, 0xB8, 0x7A, 0x7F, 0xD3, 0x6A, 0xCF, 0x0B,
70         0xB1, 0x37, 0x63, 0x4B, 0x69, 0xAE, 0x91, 0xC7,
71         0xC3, 0x97, 0xAE, 0x77, 0xF4, 0x37, 0xD7, 0x9B,
72         0x7C, 0xF5, 0x3C, 0x11, 0x8F, 0x15, 0xC3, 0xD7,
73         0xC1, 0x5B, 0x00, 0x2A, 0xD0, 0x75, 0xDE, 0x9E,
74         0x51, 0x64, 0xAB, 0x3E, 0xE9, 0x15, 0xB5, 0xAB,
75         0x56, 0x9A, 0x98, 0x82, 0x26, 0xEA, 0x2A, 0x62
76     };
77
78
79     //PN512 (0x82)
80     const byte PN512_firmware_reference[] PROGMEM = {
81         0x00, 0xEB, 0x66, 0xBA, 0x57, 0xBF, 0x23, 0x95,
82         0xD0, 0xE3, 0x0D, 0x3D, 0x27, 0x89, 0x5C, 0xDE,
83         0x9D, 0x3B, 0xA7, 0x00, 0x21, 0x5B, 0x89, 0x82,
84         0x51, 0x3A, 0xEB, 0x02, 0x0C, 0xA5, 0x00, 0x49,
85         0x7C, 0x84, 0x4D, 0xB3, 0xCC, 0xD2, 0x1B, 0x81,
86         0x5D, 0x48, 0x76, 0xD5, 0x71, 0x61, 0x21, 0xA9,
87         0x86, 0x96, 0x83, 0x38, 0xCF, 0x9D, 0x5B, 0x6D,
88         0xDC, 0x15, 0xBA, 0x3E, 0x7D, 0x95, 0x3B, 0x2F
89     };
90
91     class MFRC522 {
92     public:
93         // Size of the MFRC522 FIFO
94         static constexpr byte FIFO_SIZE = 64;           // The FIFO is 64 bytes.
95         // Default value for unused pin
96         static constexpr uint8_t UNUSED_PIN = UINT8_MAX;

```

```

373
374     // Pick the appropriate reference values
375     const byte *reference;
376     switch (version) {
377         case 0x88: // Fudan Semiconductor FM17522 clone
378             reference = FM17522_firmware_reference;
379             break;
380         case 0x90: // Version 0.0
381             reference = MFRC522_firmware_referenceV0_0;
382             break;
383         case 0x91: // Version 1.0
384             reference = MFRC522_firmware_referenceV1_0;
385             break;
386         case 0x92: // Version 2.0
387             reference = MFRC522_firmware_referenceV2_0;
388             break;
389         case 0x82: // PN512
390             reference = PN512_firmware_reference;
391             break;
392         default: // Unknown version
393             return false; // abort test
394     }
395

```

And in the PCD\_DumpVersionToSerial() in the mfrc522.h file, add this switch case line.

```
case 0x82: Serial.println(F(" = PN512")); break;
```

```
1391  ~/  
1392  void MFRC522::PCD_DumpVersionToSerial() {  
1393      // Get the MFRC522 firmware version  
1394      byte v = PCD_ReadRegister(VersionReg);  
1395      Serial.print(F("Firmware Version: 0x"));  
1396      Serial.print(v, HEX);  
1397      // Lookup which version  
1398      switch(v) {  
1399          case 0x88: Serial.println(F(" = (clone)")); break;  
1400          case 0x90: Serial.println(F(" = v0.0")); break;  
1401          case 0x91: Serial.println(F(" = v1.0")); break;  
1402          case 0x92: Serial.println(F(" = v2.0")); break;  
1403          case 0x12: Serial.println(F(" = counterfeit chip")); break;  
1404          case 0x82: Serial.println(F(" = PN512")); break;  
1405          default: Serial.println(F(" = (unknown)"));  
1406      }  
1407      // When 0x00 or 0xFF is returned, communication probably failed  
1408      if ((v == 0x00) || (v == 0xFF))
```

Next, it would be necessary to add the following 3 registers in the PCD\_Register list.

```
TypeBReg = 0x1E << 1, // Configure the ISO/IEC 14443 type B
```

```
GsNOffReg = 0x23 << 1, // Selects the conductance of the  
antenna driver pin TX1 and TX2 for modulation, when the driver is  
switched off
```

```
TxBitPhaseReg = 0x25 << 1, // Adjust the TX bit phase at 106 kbit
```

These Registers are "Reserved for future use" in the MFRC522 chip, but are used in PN512 chips.

The final step is to initialize the new registers and more in the PCD\_Init() function in the MFRC522.cpp file. Initializing these registers enables the chip to work properly

```
{
    PCD_WriteRegister(TxModeReg, 0x00);
    PCD_WriteRegister(RxModeReg, 0x00);
    PCD_WriteRegister(ModWidthReg, 0x26);

    PCD_WriteRegister(TxASKReg, 0x40);
    PCD_WriteRegister(RxThresholdReg, 0xFF);
    PCD_WriteRegister(ControlReg, 0x10);
    PCD_WriteRegister(DemodReg, 0x4D);
    PCD_WriteRegister(MfTxReg, 0x62);
    PCD_WriteRegister(TxBitPhaseReg, 0x87);
    PCD_WriteRegister(RxSelReg, 0x84);
    PCD_WriteRegister(RFCfgReg, 0x48);
    PCD_WriteRegister(GsNReg, 0x88);
    PCD_WriteRegister(CWGsPReg, 0x20);
    PCD_WriteRegister(GsNOffReg, 0x88);
    PCD_WriteRegister(ModGsPReg, 0x20);

    PCD_WriteRegister(ModeReg, 0x3D);
    PCD_WriteRegister(BitFramingReg, 0x00);
    PCD_WriteRegister(WaterLevelReg, 64);
    PCD_WriteRegister(TypeBReg, 0);
    PCD_WriteRegister(MfTxReg, 0x8A);
    PCD_AntennaOn();
} // End PCD_Init()
```

After this the device will work properly. For further assistance, we made some examples for how you would use the device on our github [https://github.com/nfc-rfid-reader-sdk/MFRC522\\_PN512](https://github.com/nfc-rfid-reader-sdk/MFRC522_PN512)

# Hardware

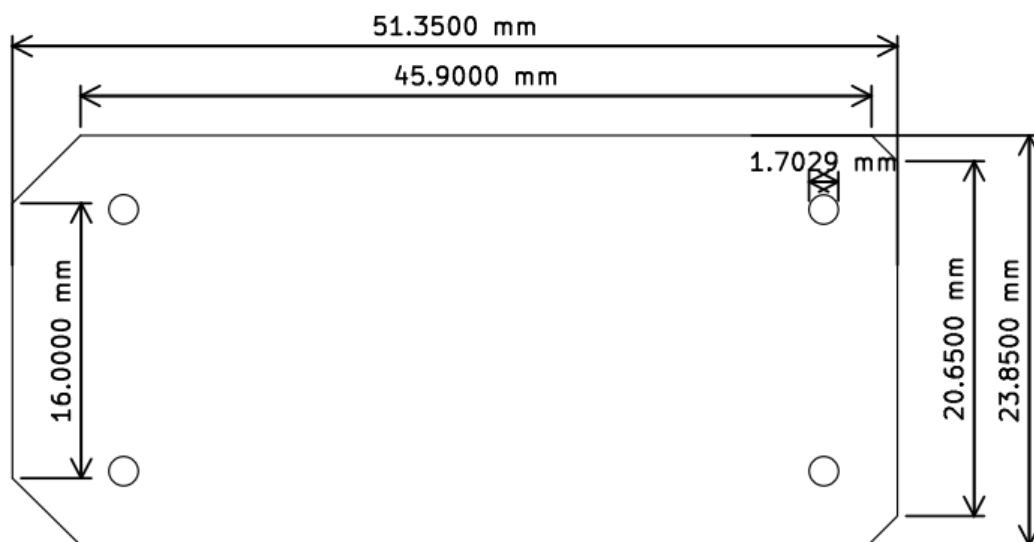
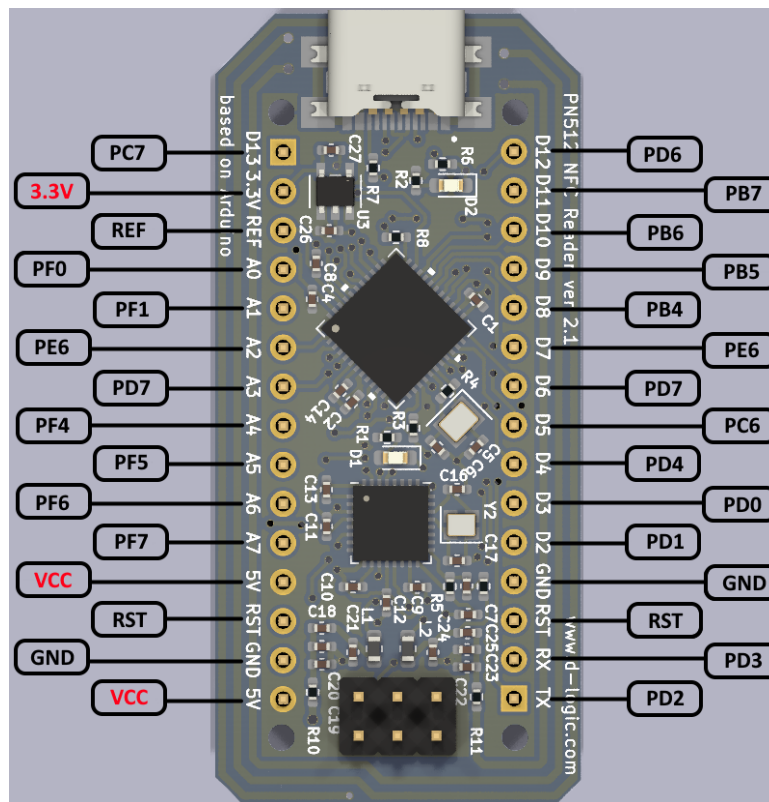
## Pinout

Pin Number	Pin Name	Mapped Pin Name	Pin Number	Pin Name	Mapped Pin Name
1	PC7	Digital pin 13	16	PD6	Digital pin 12
2	3.3V	3.3V	17	PB7	Digital pin 11
3	AREF	REF	18	PB6	Digital pin 10
4	PF0	Analog pin 0	19	PB5	Digital pin 9
5	PF1	Analog pin 1	20	PB4	Digital pin 8
6	PE6	Analog pin 2	21	PE6	Digital pin 7
7	PD7	Analog pin 3	22	PD7	Digital pin 6
8	PF4	Analog pin 4	23	PC6	Digital pin 5
9	PF5	Analog pin 5	24	PD4	Digital pin 4
10	PF6	Analog pin 6	25	PD0	Digital pin 3
11	PF7	Analog pin 7	26	PD1	Digital pin 2
12	VCC	5V	27	GND	GND
13	Reset	RST	28	Reset	RST
14	GND	GND	29	PD3	RX
15	VCC	5V	30	PD2	TX





Digital Logic



## Revision history

Date	Version	Comment
2023-8-1	1.0	Base document