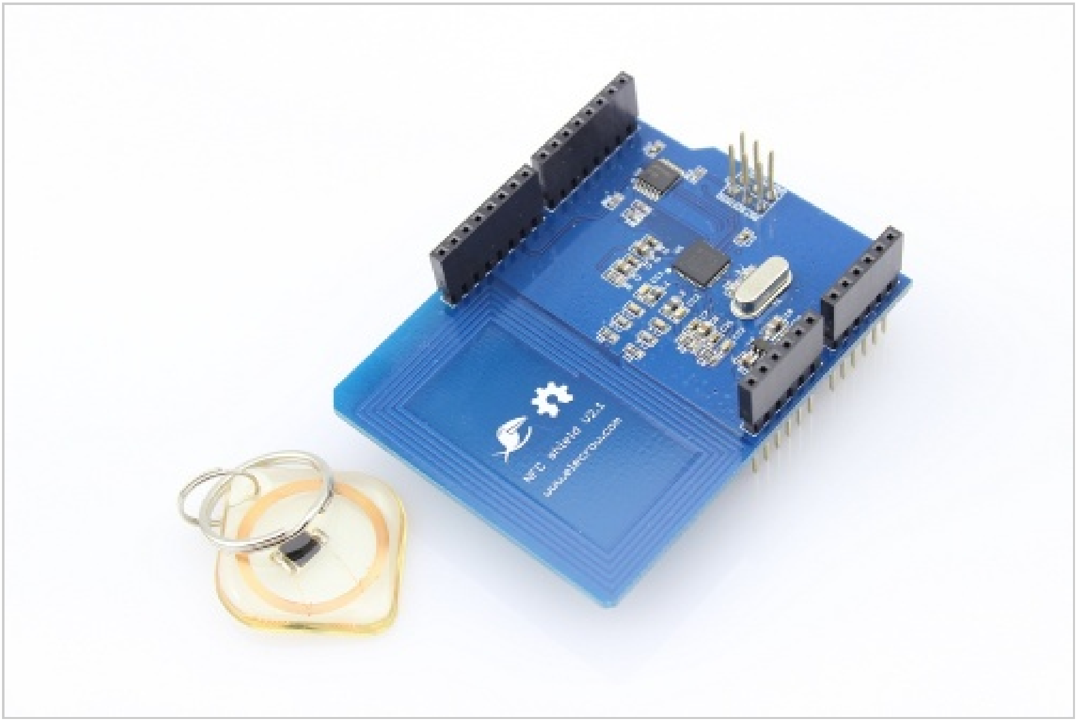
NFC Shield

From Elecrow

Contents

Introduction

The NFC(Near Field Communications) Shield uses a highly integrated transceiver module PN532 which handles contactless communication at 13.56MHz. With the arduino library, you can read and write a 13.56MHz tag with this shield or implement point to point data exchange with two NFC Shields. This shield is designed to use SPI communication protocols. you can choose the PD9 or PD10 as SS pin. so it can be used compatible with more Arduino Shields which also uses the SPI communication protocal.



NFC Shield

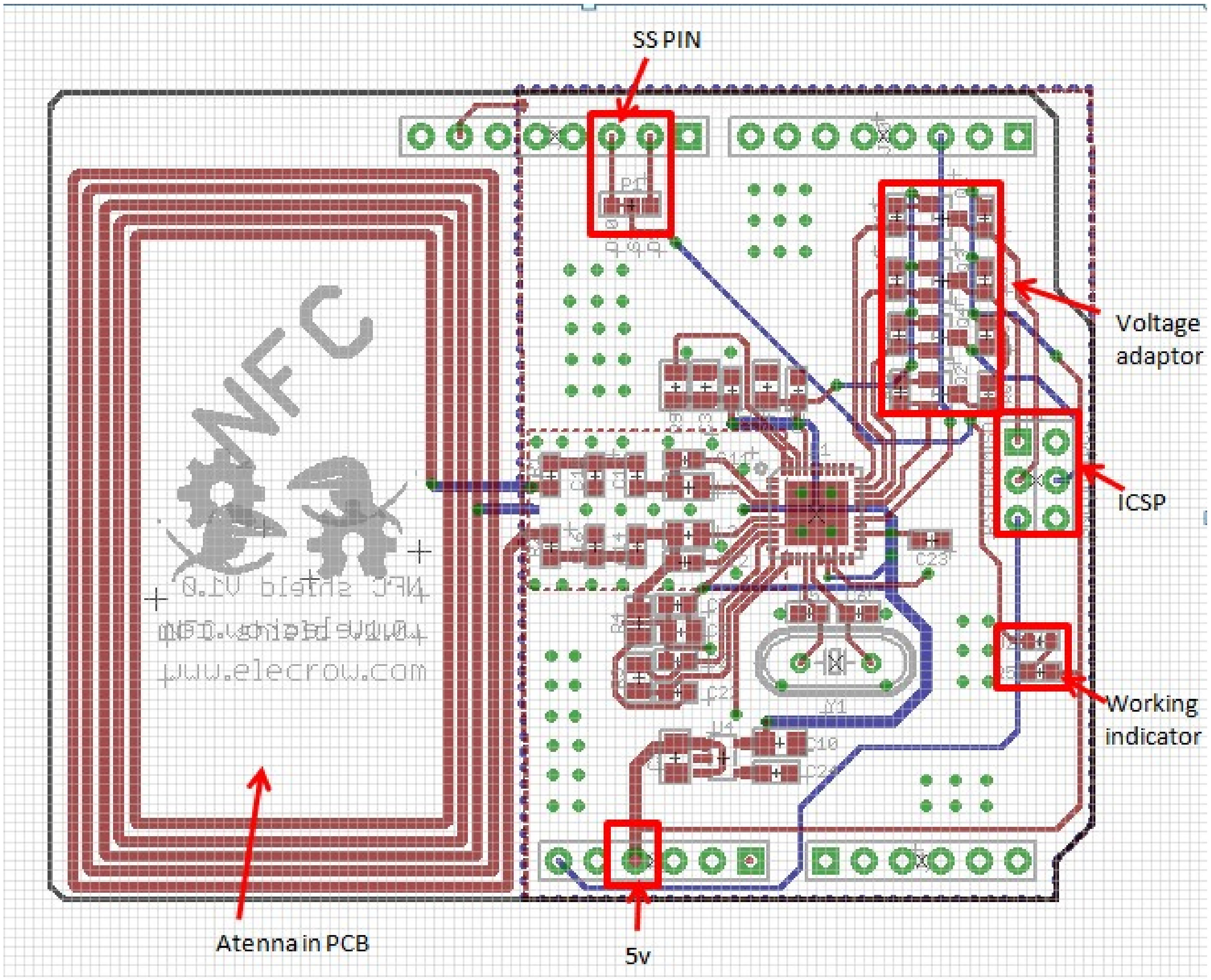
Features

- Arduino Shield compatible. No soldering required.
- **SPI** interface. Hence, most **Arduino** pins are available for other applications.
- Built in **PCB Antenna**.
- Supports both 3.3V and 5V operation.
- Socket to connect other shields.

Application Ideas

- Use as a RFID reader with **Mifare One tags** (ISO14443 Type-A) and cards (13.56Mhz).
- Build visiting card sharing system.
- Build attendance systems.
- Design authentication systems.
- Read Smart Posters.
- Securely exchange small data with other NFC devices
- And other endless possibility.

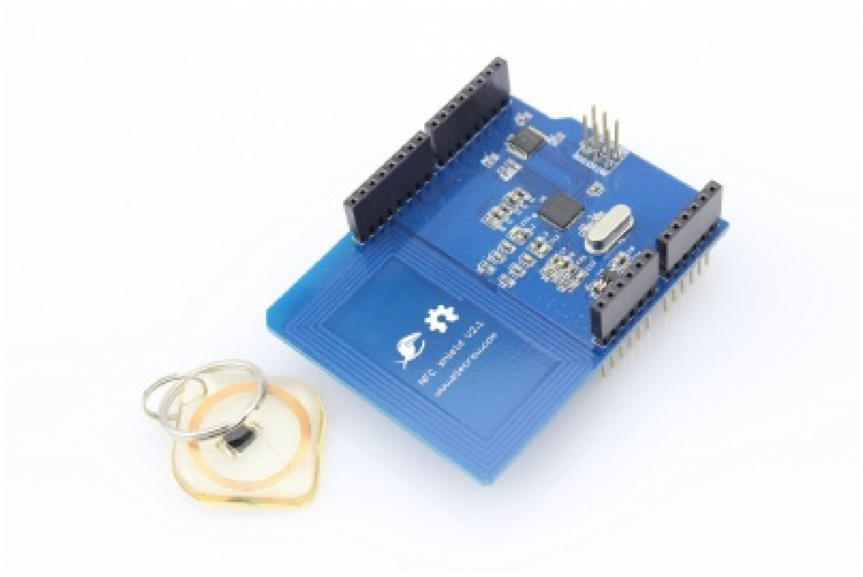
Interface



Usage

Hardware Installation

- Connect **NFC Shield** to **Crowduino** as shown below.



- Compile and upload the example sketch provided.

Programming

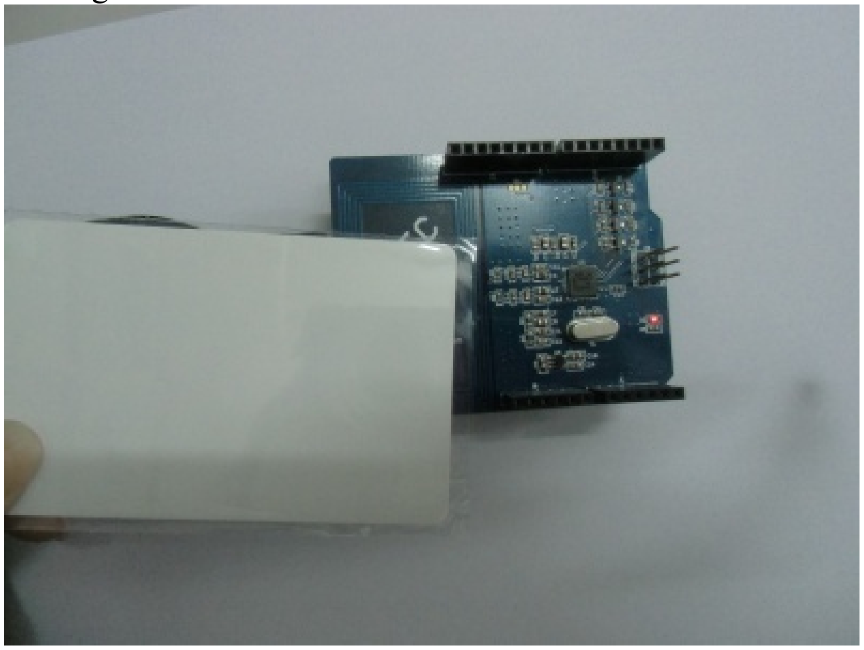
Donwload the PN532_SPI Library For NFC Shield (http://www.elecrow.com/wiki/index.php?title=File:PN532_SPI.zip) and install it to \arduino\libraries. you can either copy the folling codes into Arduino IDE or open Open the examples in the library to start.

Demo 1: Read the complete memory of a MIFARE card

1. Open and upload the example **readAllMemoryBlocks** example as show below.

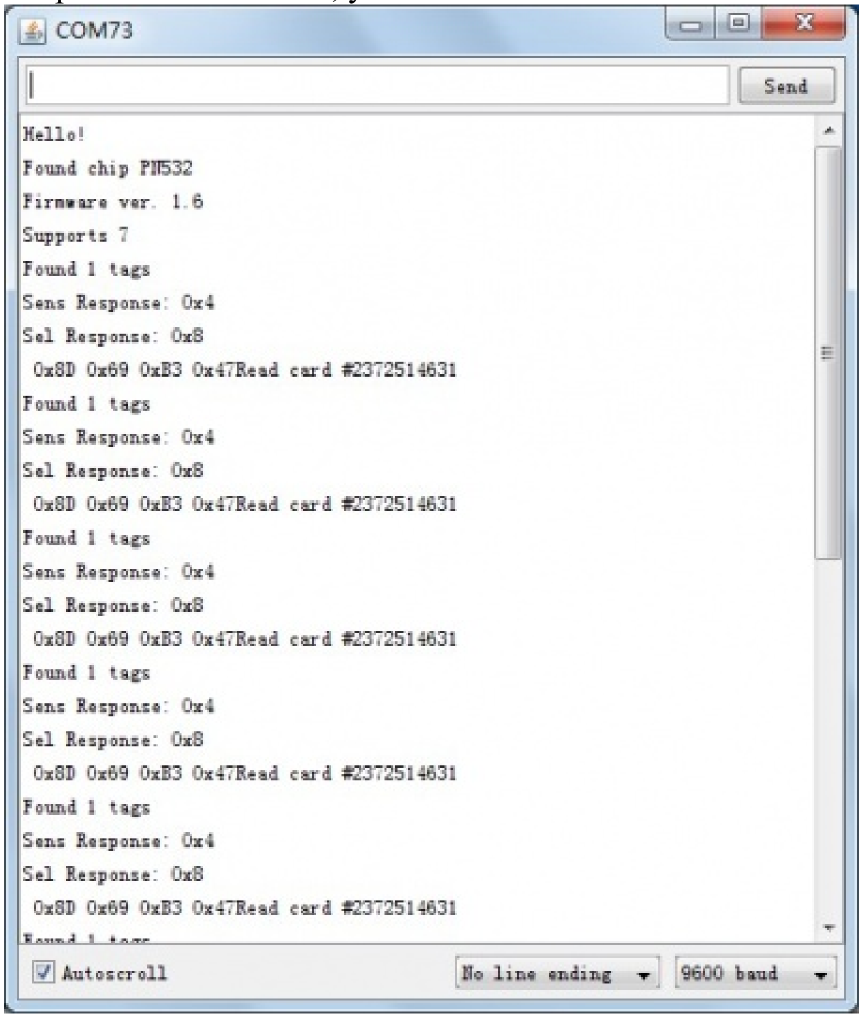
This sketch reads the complete memory of a MIFARE Standard card using default authentication keys. The output gives typical memory layout of fresh MIFARE Standard card.

2. Bring a Mifare Card near the NFC Antenna ON PCB.



Note: Blocks are classified as Manufacturer Block(read-only), Data Block (user/application writable area), and Sector Trailer(authentication and access bits for that sector)

3.Open the serial monitor, you can see the score as shown below:



```
Hello!
Found chip PN532
Firmware ver. 1.6
Supports 7
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631
Found 1 tags
```

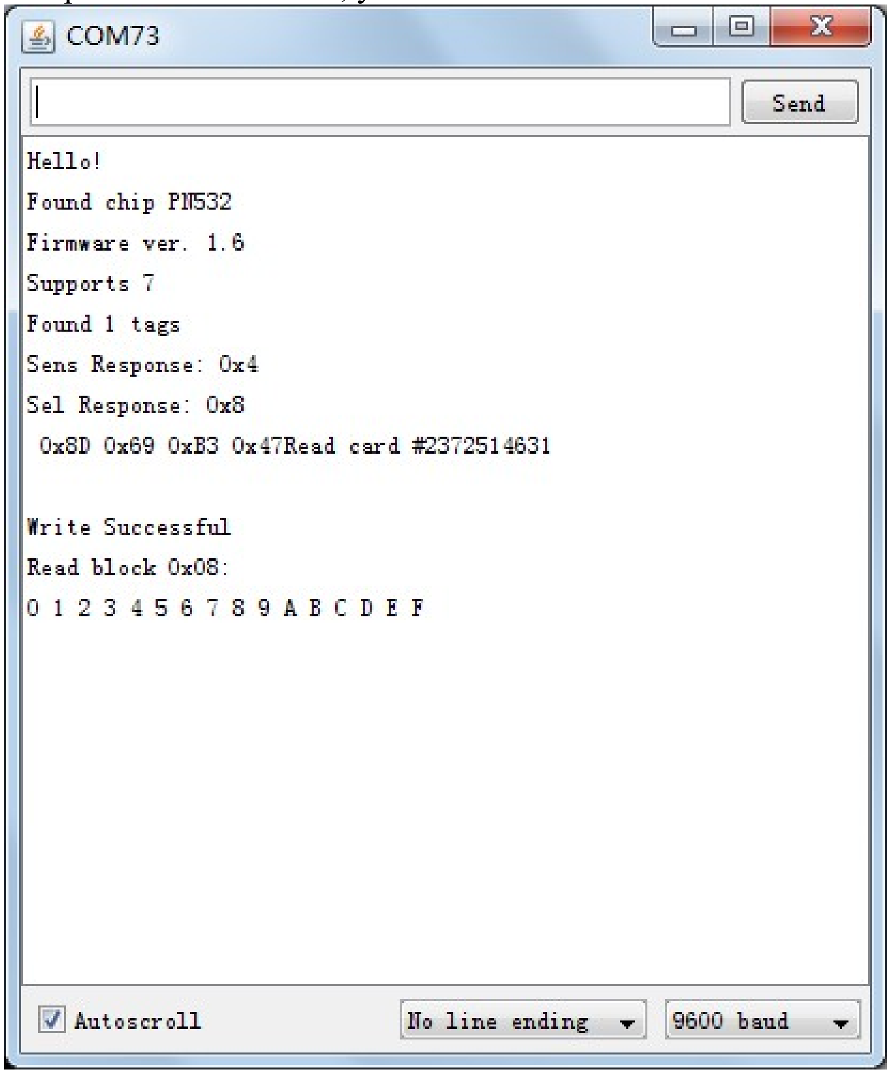
Demo 2: Write data to MIFARE Memory Block

1. Open the example **writeMifareMemory**. This example writes a MIFARE memory block 0x08 of a new MIFARE 1K cards using default authentication keys .Read the Block Memory after complete writing data to MIFARE Memory Block.

Note: Memory block 0 is readonly and contains manufacturer data. Do not write to Sector Trailer block unless you know what you are doing. Otherwise, the MIFARE card may be unusable in the future.

2. Compile and upload the example.

3. Bring a Mifare Card near the NFC Antenna.
4. Open the serial monitor, you can see the score as show below:



```
Hello!
Found chip PN532
Firmware ver. 1.6
Supports 7
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
 0x8D 0x69 0xB3 0x47Read card #2372514631

Write Successful
Read block 0x08:
0 1 2 3 4 5 6 7 8 9 A B C D E F
```

Reference code

The APIs make use of the commands to invoke the interfaces provided by PN532 firmware via SPI. All these commands are documented in PN532 User Manual. The following APIs are provided by PN532_SPI Library.

boolean SAMConfig(void)

This API invokes the **SAMConfiguration** command of PN532 and sets it to **Normal Mode**. **SAM** stands for Security Access Module (i.e the PN532 system). PN532 system can work in **Normal** mode, **Virtual Card** mode, **Wired Card** mode and **Dual Card** mode.

Usage:

```
nfc.SAMConfig(); // Call this before any read/write operation
```

uint32_t readPassiveTargetID(uint8_t cardbaudrate)

This method reads the Passive Target ID and returns it as a 32-bit number. At the moment only reading MIFARE ISO14443A cards/tags are supported. Hence use **PN532_MIFARE_ISO14443A** as parameter. *Returns* 32 bit card number

Usage:

```
uint32_t cid;
// look for MiFare type cards/tags
cid = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A);
```

uint32_t authenticateBlock(uint8_t cardnumber, uint32_t cid, uint8_t blockaddress ,uint8_t authtype, uint8_t * keys)

This method is used to authenticate a memory block with key before read/write operation. *Returns* **true** when successful.

- **cardnumber** can be 1 or 2
- **cid** is 32-bit Card ID
- **blockaddress** is block number (any number between 0 - 63 for MIFARE card)
- **authtype** is which key is to be used for authentication (either **KEY_A** or **KEY_B**)
- **keys** points to the byte-array holding 6 keys.

Usage:

```
uint8_t keys[]= {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};  // default key of a fresh card
nfc.authenticateBlock(1, id ,3,KEY_A,keys); ////authenticate block 3, id is 32-bit passive target i
```

uint32_t readMemoryBlock(uint8_t cardnumber,uint8_t blockaddress, uint8_t * block)

This method reads a memory block after authentication with the key. *Returns* **true** when successful.

- **cardnumber** can be 1 or 2
- **blockaddress** is block number (any number between 0 - 63 for MIFARE card) to read. Each block is 16bytes long in case of MIFARE Standard card.
- **block** points to buffer(byte-array)to hold 16 bytes of block-data.

Usage:

```
uint8_t block[16];
nfc.readMemoryBlock(1,3,block); //Read can be performed only when authentication was successful.
```

uint32_t writeMemoryBlock(uint8_t cardnumber,uint8_t blockaddress, uint8_t * block)

This method writes data to a memory block after authentication with the key. *Returns* **true** when successful.

- **cardnumber** can be 1 or 2
- **blockaddress** is block number (any number between 0 - 63 for MIFARE card) to write. Each block is 16bytes long in case of MIFARE Standard card.
- **block** points to buffer(byte-array) which holds 16 bytes of block-data to write.**Usage:**

```
uint8_t writeBuffer[16];
    for(uint8_t ii=0;ii<16;ii++)
     {
       writeBuffer[ii]=ii; //Fill buffer with 0,1,2....F
     }
nfc.writeMemoryBlock(1,0x08,writeBuffer); //Write writeBuffer[] to block address 0x08. Read can be
```

uint32_t PN532::configurePeerAsInitiator(uint8_t baudrate)

This method implements a Peer to Peer Initiator. *Returns* **true** when successful.

- **baudrate** can be any number from 0-2. 0 for 106kbps or 1 for 201kbps or 2 for 424kbps. At the moment only 1 and 2 are supported.

This feature is experimental and supports NFC Peer to Peer communication with other NFC shields. Interaction with mobile devices are not tested.

Usage:

```
// Configure PN532 as Peer to Peer Initiator
if( nfc.configurePeerAsInitiator(2) ) //if connection is error-free
    {
     //Your Send  Receive code here
    }
```

uint32_t configurePeerAsTarget()

This method implements a Peer to Peer Target. *Returns* **true** when successful.

This feature is experimental and supports NFC Peer to Peer communication with other NFC shields. Interaction with mobile devices are not tested.

Usage:

```
// Configure PN532 as Peer to Peer Target
if(nfc.configurePeerAsTarget()) //if connection is error-free
   {
        //You code to trans-receive data
   }
```

uint32_t initiatorTxRx(char *DataOut,char *DataIn)

This method is used to transmit and receive data to and from target. This code is used by NFC Peer to Peer Initiator. *Returns* **true** when successful.

- **DataOut** is pointer and array of chars (16 bytes) transmit data.
- **DataIn** is pointer and array of chars (16 bytes) receive data.

This feature is experimental and supports NFC Peer to Peer communication with other NFC shields. Interaction with mobile devices are not tested.

Usage:

```
  // Configure PN532 as Peer to Peer Initiator in active mode
  if( nfc.configurePeerAsInitiator(2) ) //if connection is error-free
  {
      //trans-receive data
      if(nfc.initiatorTxRx(DataOut,DataIn))
      {
       Serial.print("Data Sent and Received: ");
       Serial.println(DataIn);
      }
  }
```

uint32_t targetTxRx(char *DataOut,char *DataIn)

This method is used to transmit and receive data to and from initiator. This code is used by NFC Peer to Peer Target to respond to Initiator. *Returns* **true** when successful.

- **DataOut** is pointer and array of chars (16 bytes) transmit data.
- **DataIn** is pointer and array of chars (16 bytes) receive data.

Usage:

```
  // Configure PN532 as Peer to Peer Target
  if(nfc.configurePeerAsTarget()) //if connection is error-free
  {
      //trans-receive data
      if(nfc.targetTxRx(DataOut,DataIn))
      {
       Serial.print("Data Received: ");
       Serial.println(DataIn);
      }
  }
```

Resources

- PN532_SPI Library For NFC Shield (http://www.elecrow.com/wiki/index.php?title=File:PN532_SPI.zip)
- NFC Shield - Schematic in PDF format (http://www.elecrow.com/wiki/index.php?title=File:NFC_Shield_V1.01.pdf)
- PN532 Datasheet (http://www.elecrow.com/wiki/index.php?title=File:PN532.pdf)
- NXP PN532 - User Manual (http://www.nxp.com/documents/user_manual/141520.pdf)
- NXP Mifare One S50 IC

(http://www.nxp.com/acrobat_download2/other/identification/M001053_MF1ICS50_rev5_3.pdf)
- NFC Forum (http://www.nfc-forum.org)

Retrieved from "http://www.elecrow.com/wiki/index.php?title=NFC_Shield&oldid=6615"

- This page was last modified on 29 September 2013, at 07:12.
- This page has been accessed 1,295 times.