Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

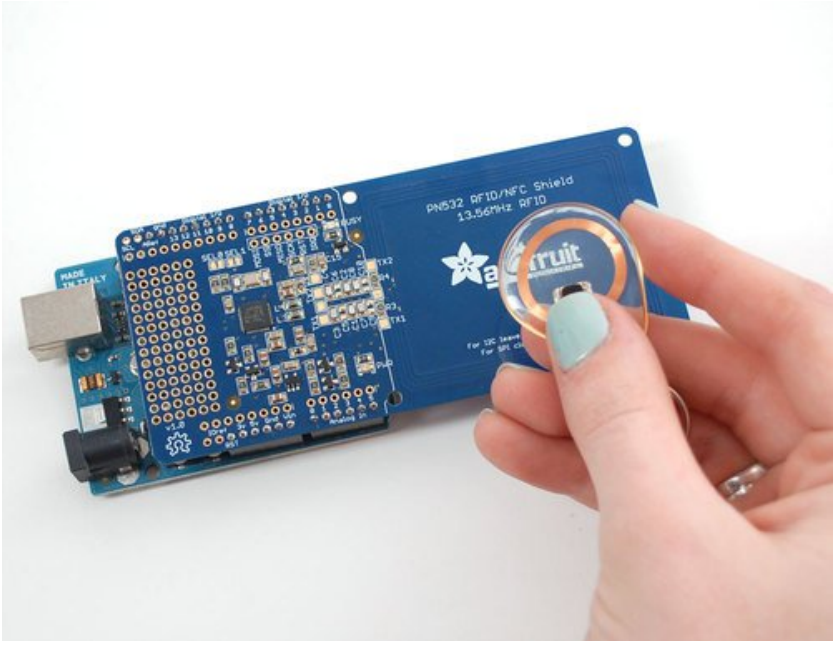
We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Adafruit PN532 RFID/NFC Breakout and Shield

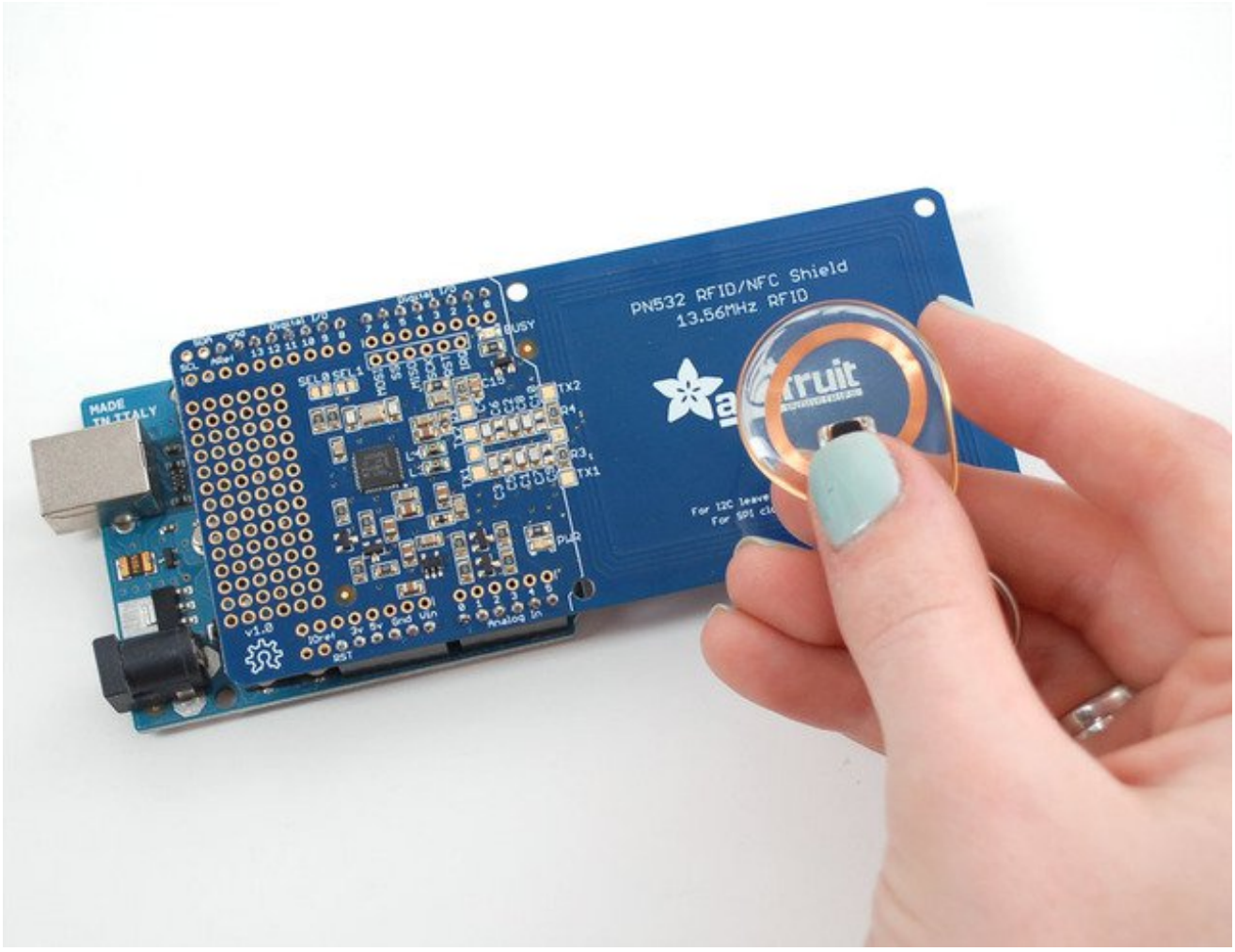
Created by lady ada



Last updated on 2016-10-12 06:38:31 PM UTC

Guide Contents

Overview

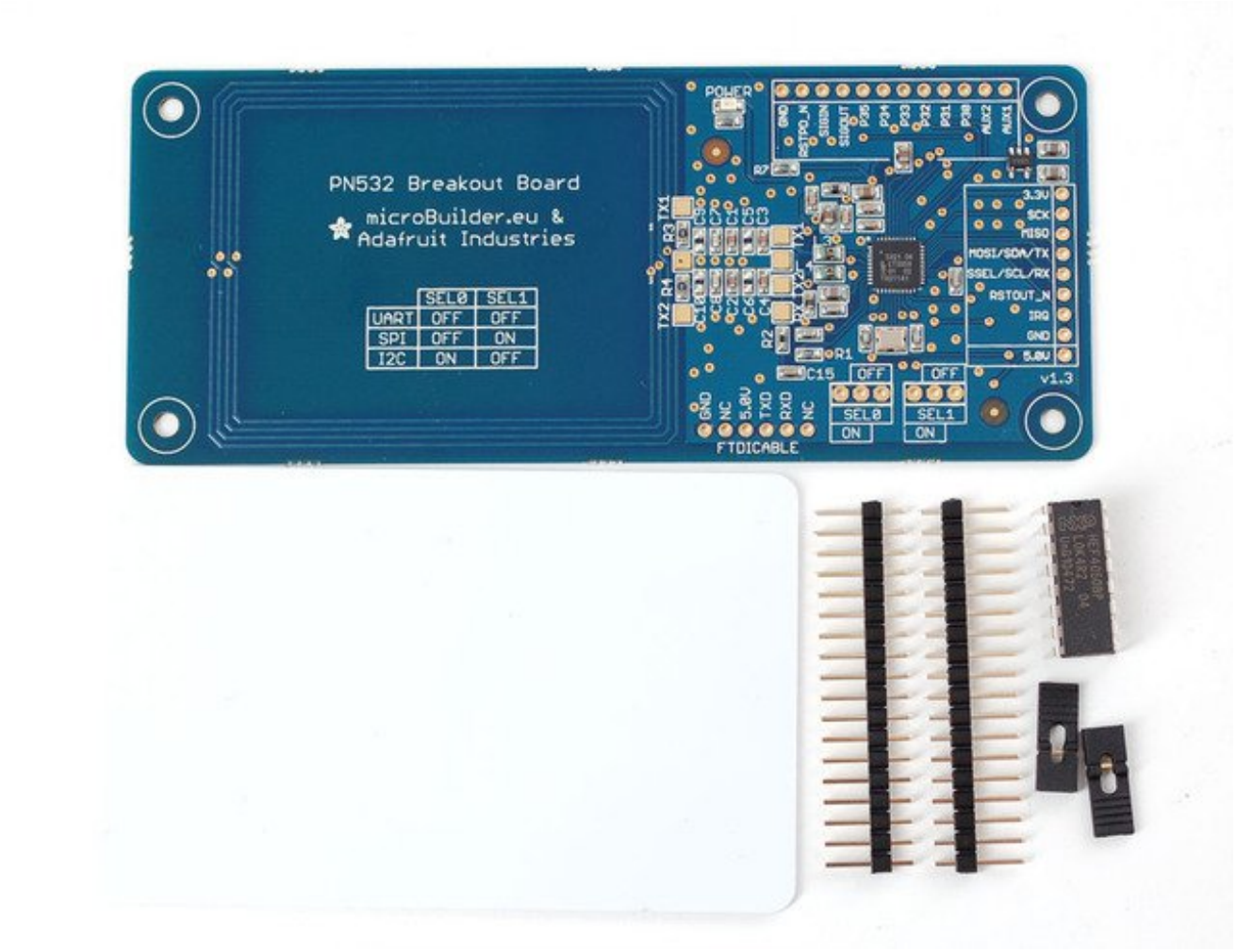


Hey! So this is not a full tutorial, its just a quickstart guide while we do more research into RFID/NFC. There's a lot of info here but not everything is explained in detail. We hope to fill out the tutorial but there's not a lot of good information about NFC so it's taking a bit of time!

Breakout Wiring

This part of the tutorial is specifically for the Breakout board. We show how to use it with SPI. The breakout also supports TTL serial and I2C but we don't have a tutorial for using it that way as SPI is the most cross-platform method to communicate

If you're using the shield, check the next page

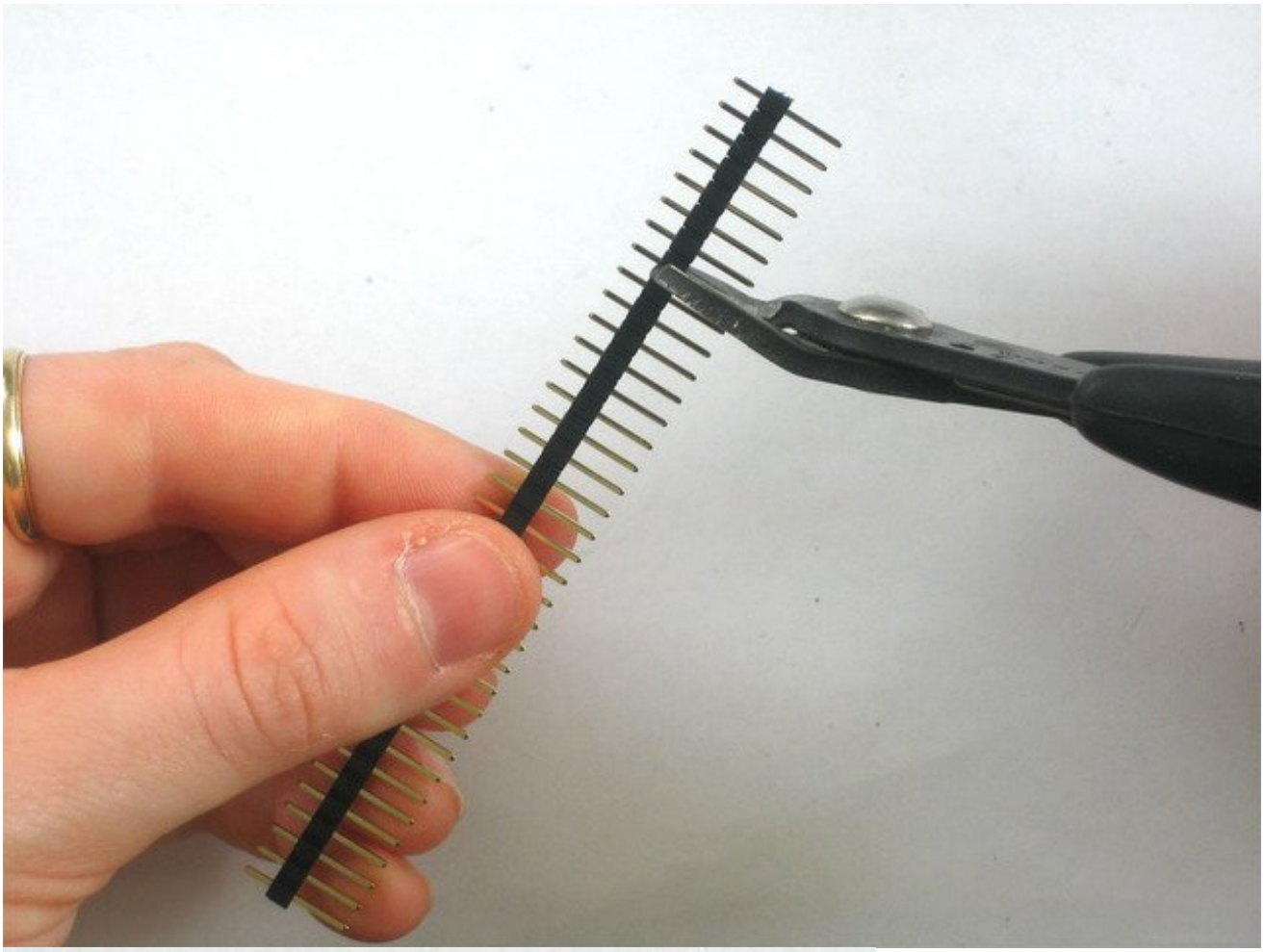


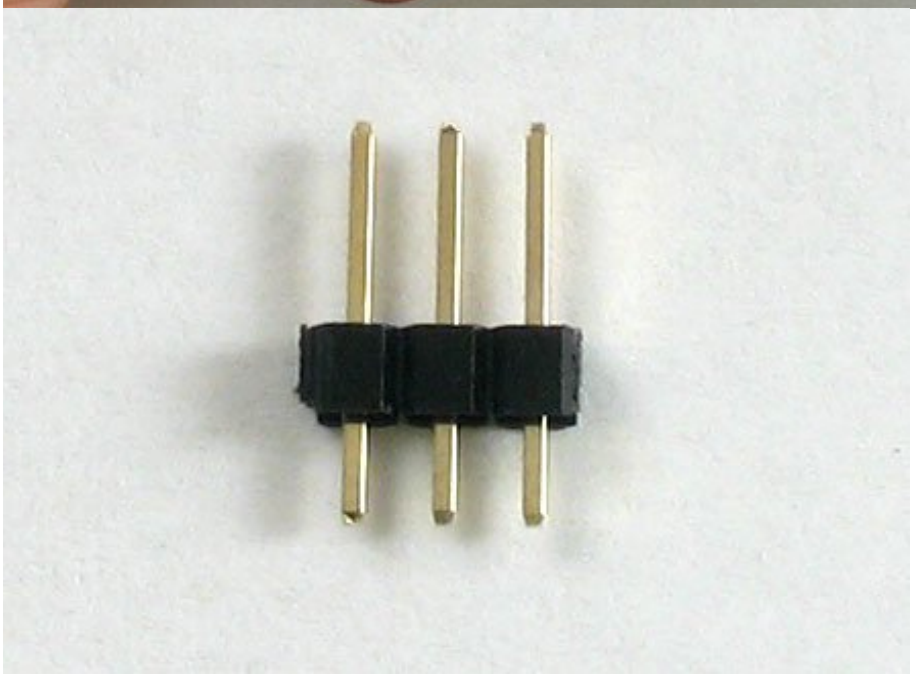
Wiring the Breakout for SPI

The PN532 chip and breakout is designed to be used by 3.3V systems. To use it with a 5V system such as an Arduino, a level shifter is required to convert the high voltages into 3.3V. If you have a 3.3V embedded system you won't have to use the shifter of course!

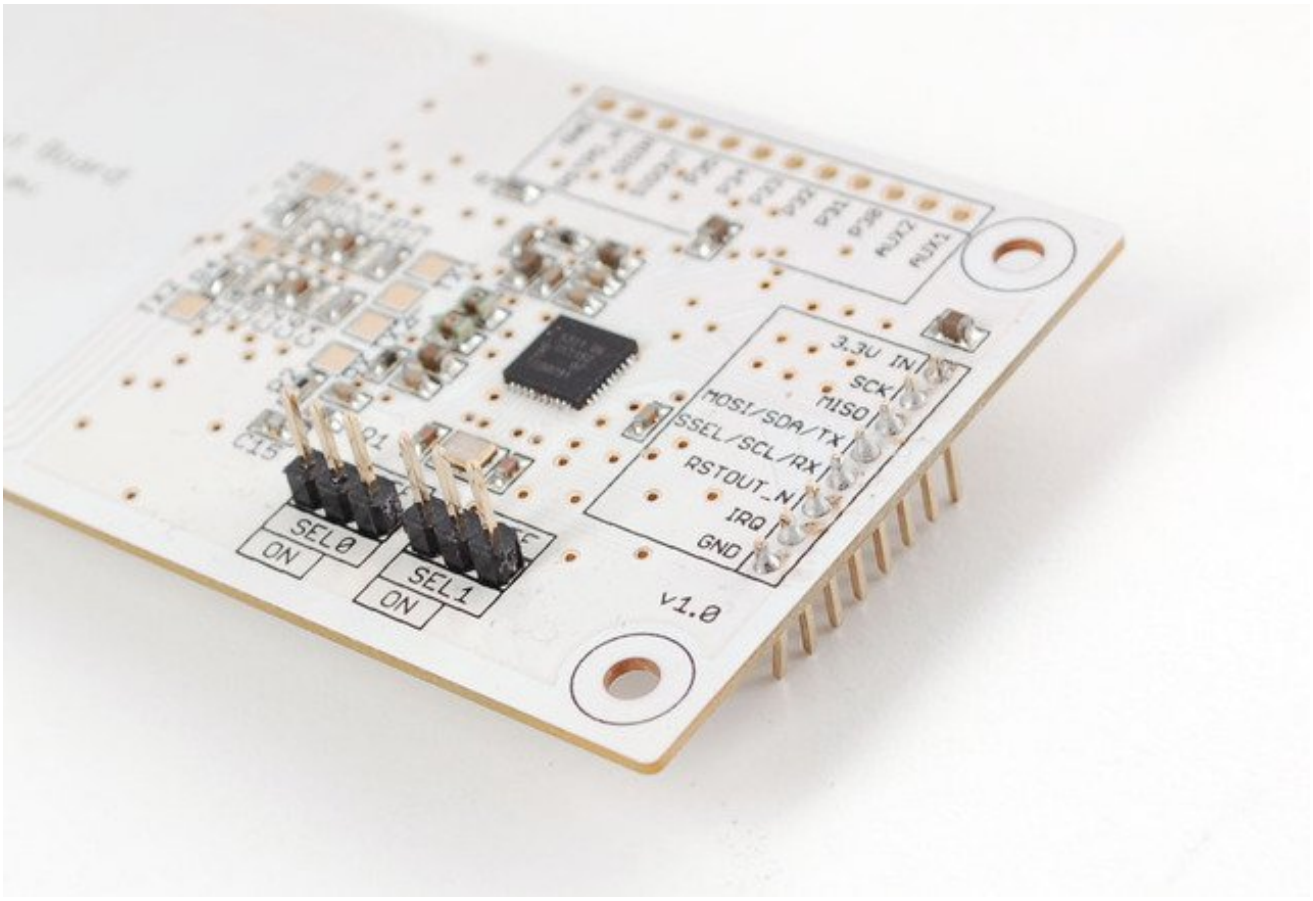
To begin, we'll solder in the header to the breakout board. You'll need two small 3-pin

pieces of header and one 8-pin piece. You can break these off of a large piece.





Solder the two small pieces to the**SEL0** and **SEL1** pads. These are interface selectors for the chip. Depending on how the jumpers are inserted the chip will talk in TTL serial, i2c or SPI. Also solder a strip to the end so you can plug it into a breadboard.



Wire up the 4050 level shifter chip to the Arduino as shown. The notch in the 4050 is at the 'top' in this image.

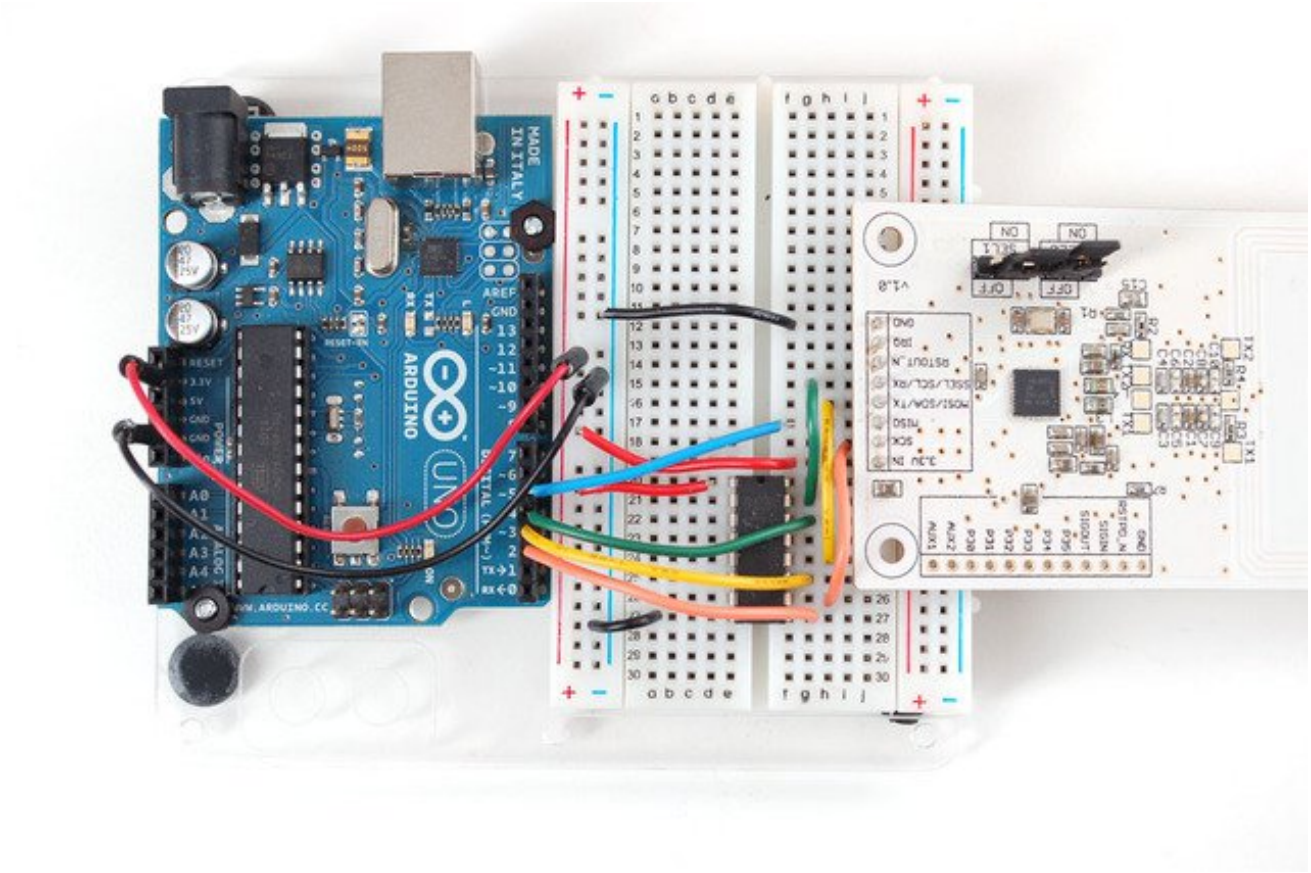
- Arduino digital pin **2** is connected to 4050 pin **9** (orange wire)
- Arduino digital pin **3** is connected to 4050 pin **11** (yellow wire)
- Arduino digital pin **4** is connected to 4050 pin **14** (green wire)

On the breakout board

- **3.3Vin** is connected to the Arduino **3.3V** pn
- **SCK** is connected to 4050 pin **10** (orange wire)
- **MISO** is connected to Arduino pin **5** (blue wire)
- **MOSI** is connected to 4050 pin **12** (yellow wire)
- **SSEL** is connected to 4050 pin **15&** (green wire)
- **GND** connects to Arduino **ground** (black wire)

Also connect 4050 pin #1 to **3.3V** and pin #8 to **ground**.

Click to see a larger image. The red power wire should be connected to the **3.3v** pin on the Arduino!



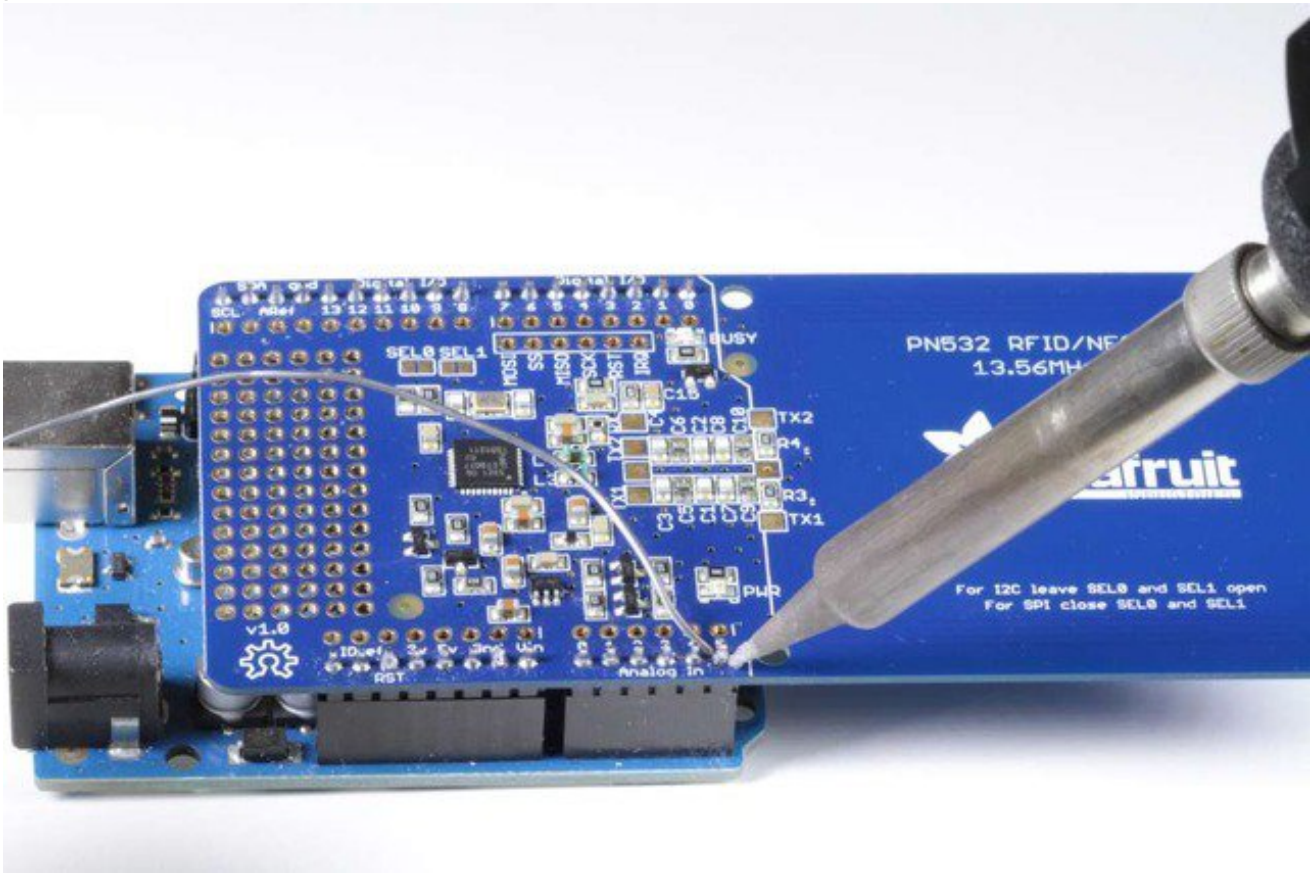
Also, we need to select SPI as the interface so on **SEL1** place the jumper in the **ON** position. for **SEL0** place the jumper in the **OFF** position.

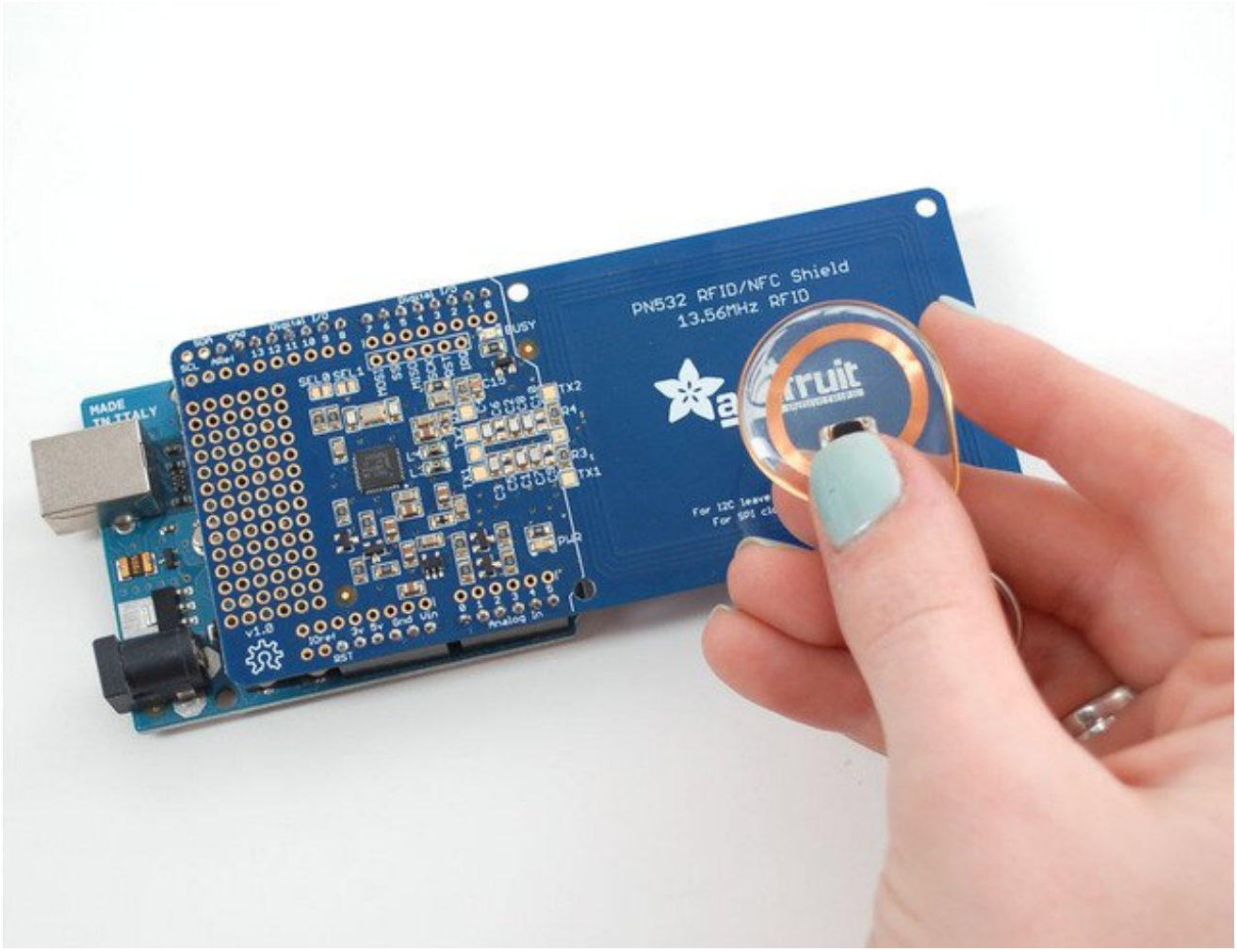
That's it! Later on you can change what Arduino pins you are using but for the beginning test we suggest matching our wiring.
If you are using the breakout in I2C mode, you will also need to add two 1.5K pullups on the SCL/SDA lines, since the breakout and the Arduino don't include the pullups. Simply solder or add a 1.5K resistor between SCL and 3.3V, and SDA and 3.3V, and then connect the breakout as you normally would.

Shield Wiring

Solder the Headers

The first step is to solder the headers to the shield. Cut the header strip to length and insert the sections (long pins down) into an Arduino. Then place the shield on top and solder each pin.





Using the Adafruit NFC Shield with I2C

The Adafruit NFC shield is designed to be used using the I2C by default. I2C only uses two pins (Analog 4 and 5 which are fixed in hardware and cannot be changed) to communicate and one pin as an 'interrupt' pin (Digital 2 - can be changed however). What is nice about I2C is that it is a 'shared' bus - unlike SPI and TTL serial - so you can put as many sensors as you'd like all on the same two pins, as long as their addresses don't collide/conflict. The Interrupt pin is handy because instead of constantly asking the NFC shield "is there a card in view yet? what about now?" constantly, the chip will alert us when a NFC target comes into the antenna range.

The shield is drop-in compatible with any Classic Arduino (UNO, Duemilanove, Diecimilla, etc using the ATmega168 or '328) as well as any Mega R3 or later.

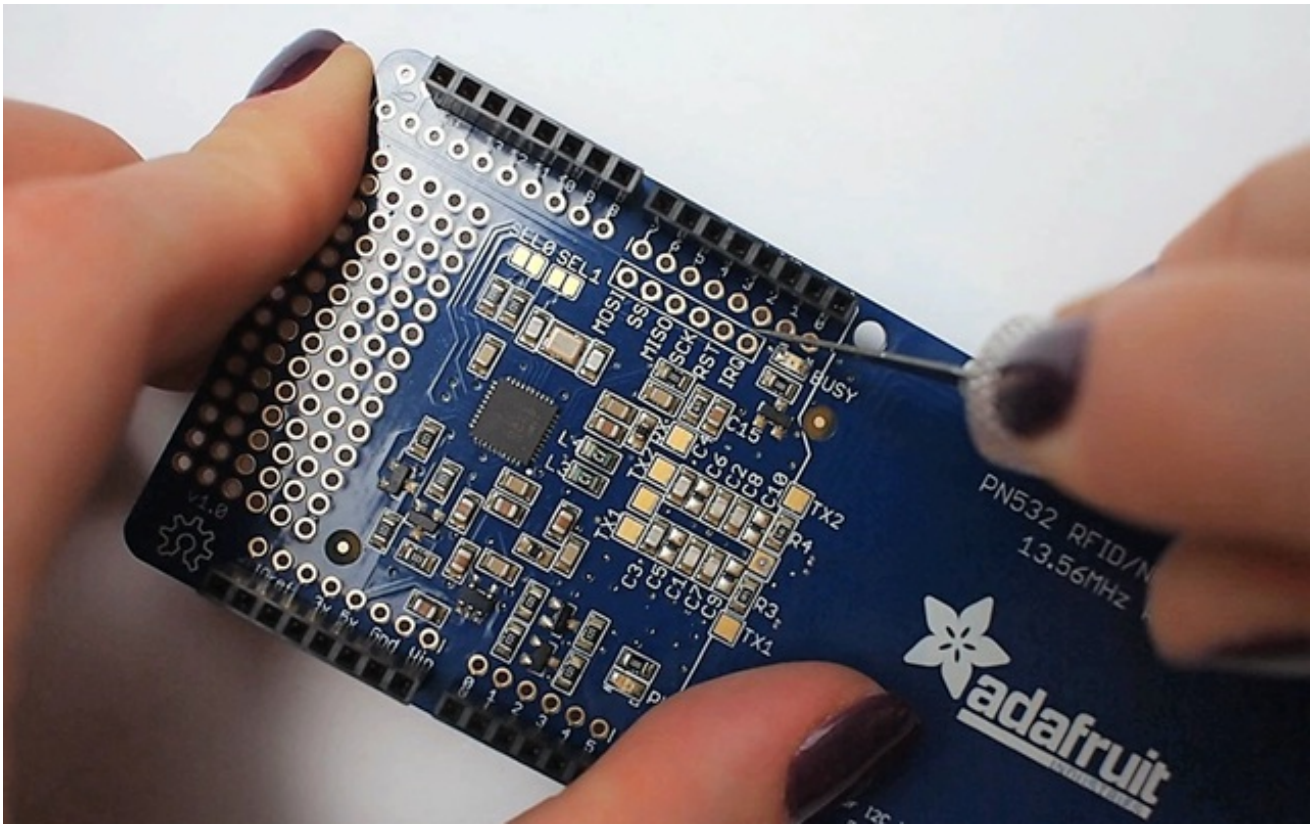
Mega R2 Arduinos work as well but you need to solder a wire from the (http://adafru.it/aUS)**SDA (http://adafru.it/aUS)** and (http://adafru.it/aUS)**SCL (http://adafru.it/aUS)** pin holes to the Mega's I2C pins on

Digital #20 and #21 (http://adafru.it/aUS)

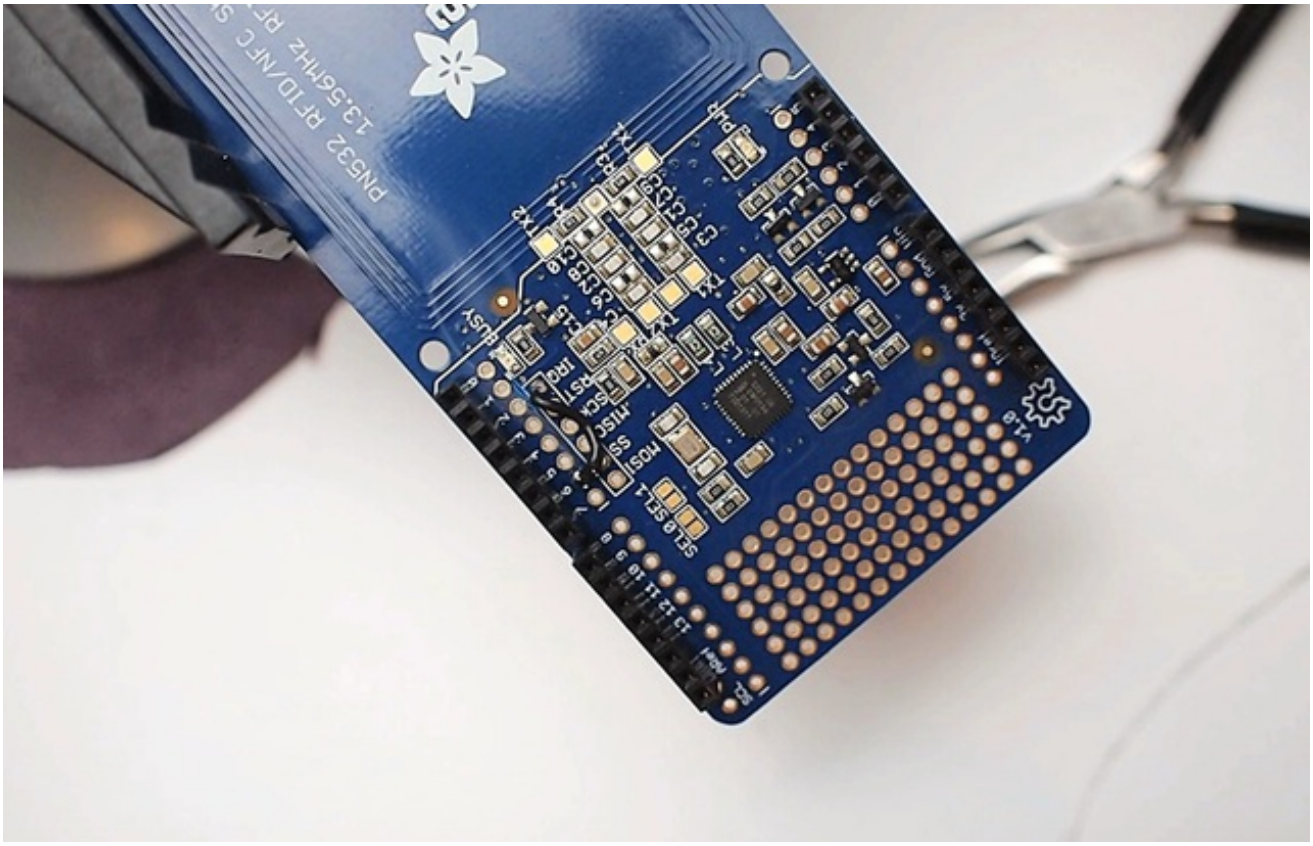
Using with the Arduino Leonardo and Yun

The IRQ pin is tied to Digital pin #2 by default. However, on the Arduino Leonardo and Yun, digital #2 is used for I2C which will not work. If using with a Leonardo or Yun, cut the trace beween the IRQ pin and Digital #2 and solder a wire from IRQ pin to Digital #4 or higher. Then change the example code so the the IRQ pin is declared as the new pin (say #6) not #2

Here are some photos of setting the IRQ pin to digital 6. First, use a sharp hobby knife to cut the trace from IRQ to 2



Solder a wire from IRQ to #6


PN532 RFID/NFC
13.56MHZ

Arduino Library

Which Library?

In the past there were two separate Arduino libraries for using the Adafruit NFC boards. One library supported the breakout over a SPI connection, and the other library supported the breakout or shield over an I2C connection. However both of these libraries have been merged into a single Arduino library, Adafruit-PN532 (http://adafru.it/eHi).

The Adafruit PN532 library has the ability to read MiFare cards, including the hard-coded ID numbers, as well as authenticate and read/write EEPROM chunks. It can work with both the breakout and shield using either a SPI or I2C connection.

Library Installation

Download the Adafruit PN532 library from github (http://adafru.it/aSX). Uncompress the folder and rename the folder **Adafruit_PN532**. Inside the folder you should see the **Adafruit_PN532.cpp** and **Adafruit_PN532.h** files. Install the **Adafruit_PN532** library foler by placing it in your ***arduinosketchfolder*/libraries** folder. You may have to create the **libraries** subfolder if this is your first library. You can read more about installing libraries in our tutorial (http://adafru.it/aYG).

Restart the Arduino IDE. You should now be able to select **File > Examples > Adafruit_PN532 > readMifare** sketch.

If you're using the NFC breakout with a SPI connection that uses the wiring shown on previous pages you can immediately upload the sketch to the Arduino and skip down to the Testing MiFare (http://adafru.it/kAc) section.

If you're using the NFC shield, or are using the breakout with an I2C connection then you must make a small change to configure the example for I2C. Scroll down to these lines near the top of the sketch:

```
// Uncomment just _one_ line below depending on how your breakout or shield
// is connected to the Arduino:

// Use this line for a breakout with a SPI connection:
Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

// Use this line for a breakout with a hardware SPI connection.  Note that
```

```
// the PN532 SCK, MOSI, and MISO pins need to be connected to the Arduino's
// hardware SPI SCK, MOSI, and MISO pins.  On an Arduino Uno these are
// SCK = 13, MOSI = 11, MISO = 12.  The SS line can be any digital IO pin.
//Adafruit_PN532 nfc(PN532_SS);

// Or use this line for a breakout or shield with an I2C connection:
//Adafruit_PN532 nfc(PN532_IRQ, PN532_RESET);
```

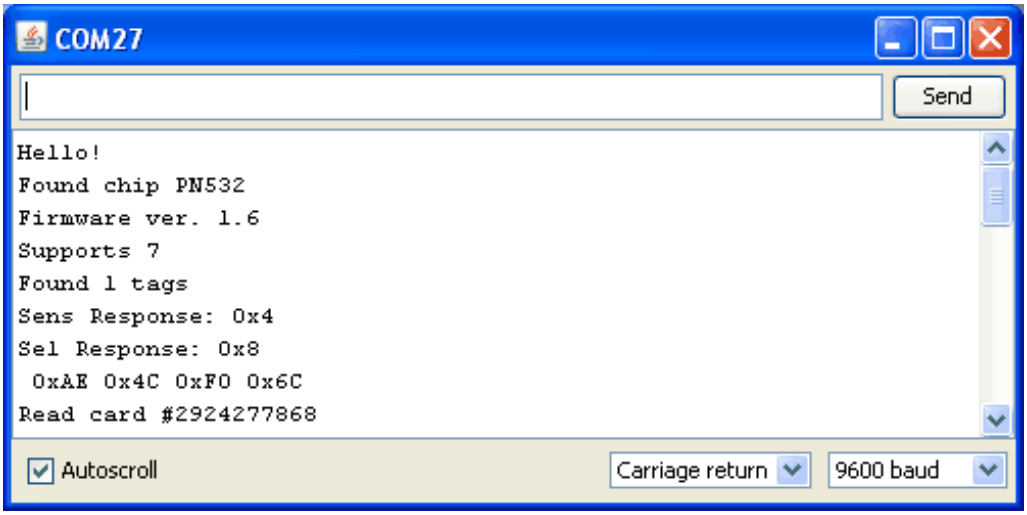
Change them so the second line is uncommented and the first line is commented. This will configure the sketch to make the library use I2C for communication with the NFC shield or breakout. The modified code should look like:

```
// Uncomment just _one_ line below depending on how your breakout or shield
// is connected to the Arduino:

// Use this line for a breakout with a SPI connection:
//Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

// Use this line for a breakout with a hardware SPI connection.  Note that
// the PN532 SCK, MOSI, and MISO pins need to be connected to the Arduino's
// hardware SPI SCK, MOSI, and MISO pins.  On an Arduino Uno these are
// SCK = 13, MOSI = 11, MISO = 12.  The SS line can be any digital IO pin.
//Adafruit_PN532 nfc(PN532_SS);

// Or use this line for a breakout or shield with an I2C connection:
Adafruit_PN532 nfc(PN532_IRQ, PN532_RESET);
```

Then upload the example to the Arduino and continue on. **Note that you need to make a similar change to pick the interface for any other NFC example from the library.**

Testing MiFare

In the serial monitor, you should see that it found the**PN532** chip. Then you can place your tag nearby and it will display the 4 byte ID code (this one is 0xAE 0x4C 0xF0 0x6C) and then the integer version of all four bytes together. You can use this number to identify each card. Recently NXP made so many cards that they actually ran through all 4 Bytes (2^32) so the number is not guaranteed to be absolutely unique. However, the chances are extremely slim you will have two cards with the same ID so as long as you aren't using these cards for anything terribly important (like money transfer) its fine to use the number as a unique identifier


COM27
Send
Hello!
Found chip PN532
Firmware ver. 1.6
Supports 7
Found 1 tags
Sens Response: 0x4
Sel Response: 0x8
0xAE 0x4C 0xF0 0x6C
Read card #2924277868
Autoscroll
Carriage return
9600 baud

About NFC

NFC (Near Field Communication)

NFC (Near Field Communication) is a set of short-range (typically up to 10cm) wireless communication technologies designed to offer light-weight and secure communication between two devices. While NFC was invented by NXP (Phillips at the time), Nokia and Sony, the main body behind the NFC 'standard' today is the NFC Forum (http://adafru.it/aSy), who are responsible for publishing and maintaining a variety of standards relating to NFC technology.

NFC operates at 13.56MHz, and is based around an "initiator" and "target" model where the initiator generates a small magnetic field that powers the target, meaning that the target does not require a power source. This means of communication is referred to as **Passive Communication**, and is used to read and write to small, inexpensive 13.56MHz RFID tags based on standards like ISO14443A. **Active communication** (peer-to-peer) is also possible when both devices are powered, where each device alternately creates its own magentic field, with the secondary device as a target and vice versa in continuous rotation.

Passive Communication: ISO14443A Cards (Mifare, etc.)

While the PN53x family of transceivers from NXP are compatible with a number of 13.56MHz RFID card standards, by far the most popular standard is ISO14443A. A variety of manufacturers produce ISO14443A compatible cards or chips, but the most common are based around the **Mifare** family from NXP. Mifare Classic and Mifare Ultralight are probably the most frequently encountered and useful for basic projects, though many tags with improved security and encryption also exist (Mifare DESFire, etc.). All of the tags sold at adafruit.com are Mifare Classic 1K, meaning that they contains 1K (1024 bytes) of programmable EEPROM memory which can be read and modified in passive mode by the initiator device (the PN532).

While all ISO14443A cards share certain common characteristics on the highest level (defined by the four part standard), each set of Mifare chips (Classic, Ultralight, Plus, DESFire, etc.) has it's own features and peculiarities. The two most common formats are described below.

- Mifare Classic (http://adafru.it/cl7): These cards are extremely common, and contain 1K or 4K of EEPROM, with basic security for each 64 byte (1K/4K cards) or 256 byte

(4K cards) sector.
- Mifare Ultralight (http://adafru.it/cl7): Contains 512 bytes of EEPROM, including 32-bits of OTP memory. These tags are inexpensive, often come in sticker format and are are frequently used for transportation ticketing, concert tickets, etc.

Active Communication (Peer-to-Peer)

Active or "Peer-to-Peer" communication is still based around the Initiator/Target model described earlier, but both devices are actively powered and switch roles from being an Initiator or a Target during the communication. When one device is initiating a conversation with the other, it enables it's magnetic field and the receiving device listens in (with it's own magnetic field disabled). Afterwards, the target/recipient device may need to respond and will in turn activate it's own magnetic field and the original device will be configured as the target. Despite two devices being present, only one magnetic field is active at a time, with each device constantly enabling or disabling its own magnetic field.

ToDo: Add better description of active mode, but I need to test it out a bit first myself!

NFC Data Exchange Format (NDEF)

The NFC Data Exchange Format (NDEF) is a standardised data format that can be used to exchange information between any compatible NFC device and another NFC device or tag. The data format consists of **NDEF Messages** and **NDEF Records**. The standard is maintained by the NFC Forum and is freely available for consultation but requires accepting a license agreement to download (http://adafru.it/aSA).

The NDEF format is used to store and exchange information like URIs, plain text, etc., using a commonly understood format. NFC tags like Mifare Classic cards can be configured as NDEF tags, and data written to them by one NFC device (NDEF Records) can be understood and accessed by any other NDEF compatible device. NDEF messages can also be used to exchange data between two active NFC devices in "peer-to-peer" mode. By adhering to the NDEF data exchange format during communication, devices that would otherwise have no meaningful knowledge of each other or common language are able to share data in an organised, mutually understandable manner.

The NDEF standard includes numerous **Record Type Definitions (RTDs)** that define how information like URIs should be stored, and each NDEF device, tag or message can contained multiple RTDs. Standard RTD definitions are described in "NFC Record Type Definition (RTD) Specification" maintained by the NFC Forum.

* NDEF Overview (http://adafru.it/cl7): This page offers a more detailed explanation of

NDEF, including how Mifare Classic cards can be used to store NDEF messages.

NOTE: The dedicated NDEF page is still a work in progress and some information is currently incomplete.

Reading

For more details about NFC/RFID and this chip we suggest the following fantastic resources:

- RFID selection guide (http://adafru.it/aSC) - a lot of details about RFID in general
- Nokia's Introduction to NFC (http://adafru.it/aSD)- a lot of details about NFC in general
- NXP S50 chip datasheet (http://adafru.it/aSE) , the chip *inside* MiFare classic tags
- NXP PN532 Short Form Datasheet (http://adafru.it/aSF)
- NXP PN532 Long Form Datasheet (http://adafru.it/aSG)
- NXP PN532 User Manual (http://adafru.it/aSH)
- NXP PN532 App Note (http://adafru.it/aSI)
- Using PN532 with libnfc (http://adafru.it/aSJ)

- NFC Glossary (http://adafru.it/aSK)

MiFare Cards & Tags

MiFare is one of the four 13.56MHz card 'protocols' (FeliCa is another well known one) All of the cards and tags sold at the Adafruit shop use the inexpensive and popular MiFare Classic chipset

MiFare Classic Cards

MIFARE Classic cards come in 1K and 4K varieties. While several varieties of chips exist, the two main chipsets used are described in the following publicly accessible documents:

- MF1S503x Mifare Classic 1K data sheet (http://adafru.it/aSL)
- MF1S70yyX MIFARE Classic 4K data sheet (http://adafru.it/aSM)

Mifare Classic cards typically have a **4-byte NUID** that uniquely (within the numeric limits of the value) identifies the card. It's possible to have a 7 byte IDs as well, but the 4 byte models are far more common for Mifare Classic.

EEPROM Memory

Mifare Classic cards have either 1K or 4K of EEPROM memory. Each memory block can be configured with different access conditions, with two seperate authentication keys present in each block.

Mifare Classic cards are divided into section called **sectors** and **blocks**. Each "sector" has individual access rights, and contains a fixed number of "blocks" that are controlled by these access rights. Each block contains 16 bytes, and sectors contains either 4 blocks (1K/4K cards) for a total of 64 bytes per sector, or 16 blocks (4K cards only) for a total of 256 bytes per sector. The card types are organised as follows:

- **1K Cards** - 16 sectors of 4 blocks each (sectors 0..15)
- **4K Cards** - 32 sectors of 4 blocks each (sectors 0..31) and 8 sectors of 16 blocks each (sectors 32..39)

4 Block Sectors

1K and 4K cards both use 16 sectors of 4 blocks each, with the bottom 1K of memory on the 4K cards being organised identically to the 1K models for compatability reasons. These

individual 4 block sectors (containing 64 byts each) have basic security features are can each be configured with seperate read/write access and two different 6-byte authentication keys (the keys can be different for each sector). Due to these security features (which are stored in the last block, called the **Sector Trailer**), only the bottom 3 blocks of each sector are actually available for data storage, meaning you have 48 bytes per 64 byte sector available for your own use.

Each 4 block sector is organised as follows, with four rows of 16 bytes each for a total of 64-bytes per sector. The first two sectors of any card are shown:

```
Sector  Block   Bytes                                          Description
 ------ -----  -----                                         -----------
            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15

 1      3     [-------KEY A-------]  [Access Bits]  [-------KEY B-------]  Sector Trailer
        2     [                    Data                    ]  Data
        1     [                    Data                    ]  Data
        0     [                    Data                    ]  Data

 0      3     [-------KEY A-------]  [Access Bits]  [-------KEY B-------]  Sector Trailer
        2     [                    Data                    ]  Data
        1     [                    Data                    ]  Data
        0     [                Manufacturer Data           ]  Manufacturer Block
```

Sector Trailer (Block 3)
The sector trailer block contains the two secret keys (Key A and Key B), as well as the access conditions for the four blocks. It has the following structure:

```
Sector Trailer Bytes
----------------------------------------------------------------
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
[      Key A      ]  [Access Bits]  [      Key B      ]
```

For more information in using Keys to access the clock contents, see Accessing Data Blocks further below.

Data Blocks (Blocks 0..2)
Data blocks are 16 bytes wide and, depending on the permissions set in the access bits, can be read from and written to. You are free to use the 16 data bytes in any way you wish. You can easily store text input, store four 32-bit integer values, a 16 character uri, etc.

Data Blocks as "Value Blocks"
An alternative to storing random data in the 16 byte-wide blocks is to configure them as "Value Blocks". Value blocks allow performing electronic purse functions (valid commands are: read, write, increment, decrement, restore, transfer).

Each Value block contains a single signed 32-bit value, and this value is stored 3 times for

data integrity and security reasons. It is stored twice non-inverted, and once inverted. The last 4 bytes are used for a 1-byte address, which is stored 4 times (twice non-inverted, and twice inverted).

Data blocks configured as "Value Blocks" have the following structure:

```
Value Block Bytes
-----------------------------------------------------------------
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
[  Value  ] [  ~Value ] [  Value  ] [A ~A A  ~A]
```

Manufacturer Block (Sector 0, Block 0)
Sector 0 is special since it contains the Manufacturer Block. This block contains the manufacturer data, and is read-only. It should be avoided unless you know what you are doing.

16 Block Sectors

16 block sectors are identical to 4 block sectors, but with more data blocks. The same structure described in the 4 block sectors above applies.

```
Sector  Block   Bytes                                   Description
------  -----   -----                                   ----------
            0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15

32      15      [-------KEY A-------]   [Access Bits]   [-------KEY B-------]   Sector Trailer 32
        14      [                       Data                    ]   Data
        13      [                       Data                    ]   Data
        ...
        2       [                       Data                    ]   Data
        1       [                       Data                    ]   Data
        0       [                       Data                    ]   Data
```

Accessing EEPROM Memory

To access the EEPROM on the cards, you need to perform the following steps:

1. You must retrieve the 4-byte NUID of the card (this can sometimes be 7-bytes long as well, though rarely for Mifare Classic cards). This is required for the subsequent authentication process.
2. You must authenticate the sector you wish to access according to the access rules defined in the Sector Trailer block for that sector, by passing in the appropriate 6 byte Authentication Key (ex. 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF for new cards).
3. Once authenication has succeeded, and depending on the sector permissions, you can then read/write/increment/decrement the contents of the specific block. Note that

you need to re-authenticate for each sector that you access, since each sector can have it's own distinct access keys and rights!

Note on Authentication

Before you can do access the sector's memory, you first need to "authenticate" according to the security settings stored in the Sector Trailer. By default, any new card will generally be configured to allow full access to every block in the sector using Key A and a value of 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF. Some other common keys that you may wish to try if this doesn't work are:

```
0XFF 0XFF 0XFF 0XFF 0XFF 0XFF
0XD3 0XF7 0XD3 0XF7 0XD3 0XF7
0XA0 0XA1 0XA2 0XA3 0XA4 0XA5
0XB0 0XB1 0XB2 0XB3 0XB4 0XB5
0X4D 0X3A 0X99 0XC3 0X51 0XDD
0X1A 0X98 0X2C 0X7E 0X45 0X9A
0XAA 0XBB 0XCC 0XDD 0XEE 0XFF
0X00 0X00 0X00 0X00 0X00 0X00
0XAB 0XCD 0XEF 0X12 0X34 0X56
```

Example of a New Mifare Classic 1K Card

The follow memory dump illustrates the structure of a 1K Mifare Classic Card, where the data and Sector Trailer blocks can be clearly seen:

```
[-------------------------Start of Memory Dump-------------------------]
------------------------Sector 0-------------------------
Block 0  8E 02 6F 66 85 08 04 00 62 63 64 65 66 67 68 69  ?.of?...bcdefghi
Block 1  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 3  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 1-------------------------
Block 4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 5  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 7  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 2-------------------------
Block 8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 11 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 3-------------------------
Block 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 15 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 4-------------------------
```

```
Block 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 19 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 5-------------------------
Block 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 23 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 6-------------------------
Block 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 26 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 27 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 7-------------------------
Block 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 31 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 8-------------------------
Block 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 35 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 9-------------------------
Block 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 37 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 39 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 10------------------------
Block 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 43 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 11------------------------
Block 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 46 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 47 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 12------------------------
Block 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 51 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 13------------------------
Block 52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 55 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 14------------------------
Block 56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```