
AdafruitPN532 Library Documentation

Release 1.0

ladyada

Jun 15, 2019

Contents

1	Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
3	Contributing	7
4	Building locally	9
4.1	Zip release files	9
4.2	Sphinx documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	adafruit_pn532	12
5.2.1	Implementation Notes	12
5.3	adafruit_pn532.i2c	13
5.4	adafruit_pn532.spi	13
5.5	adafruit_pn532.uart	14
6	Indices and tables	15
	Python Module Index	17
	Index	19

CircuitPython driver for the [PN532 NFC/RFID Breakout](#) and [PN532 NFC/RFID Shield](#)

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

1.1 Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-pn532
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-pn532
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-pn532
```


CHAPTER 2

Usage Example

Check `examples/pn532_simpletest.py` for usage example

CHAPTER 3

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

4.1 Zip release files

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-pn532 --library_
↳ location .
```

4.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/pn532_simpletest.py

```
1  """
2  This example shows connecting to the PN532 with I2C (requires clock
3  stretching support), SPI, or UART. SPI is best, it uses the most pins but
4  is the most reliable and universally supported.
5  After initialization, try waving various 13.56MHz RFID cards over it!
6  """
7
8  import board
9  import busio
10 from digitalio import DigitalInOut
11 #
12 # NOTE: pick the import that matches the interface being used
13 #
14 from adafruit_pn532.i2c import PN532_I2C
15 #from adafruit_pn532.spi import PN532_SPI
16 #from adafruit_pn532.uart import PN532_UART
17
18 # I2C connection:
19 i2c = busio.I2C(board.SCL, board.SDA)
20
21 # Non-hardware
22 pn532 = PN532_I2C(i2c, debug=False)
23
24 # With I2C, we recommend connecting RSTPD_N (reset) to a digital pin for manual
25 # hardware reset
26 reset_pin = DigitalInOut(board.D6)
27 # On Raspberry Pi, you must also connect a pin to P32 "H_Request" for hardware
```

(continues on next page)

(continued from previous page)

```

28 # wakeup! this means we don't need to do the I2C clock-stretch thing
29 req_pin = DigitalInOut(board.D12)
30 pn532 = PN532_I2C(i2c, debug=False, reset=reset_pin, req=req_pin)
31
32 # SPI connection:
33 #spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
34 #cs_pin = DigitalInOut(board.D5)
35 #pn532 = PN532_SPI(spi, cs_pin, debug=False)
36
37 # UART connection
38 #uart = busio.UART(board.TX, board.RX, baudrate=115200, timeout=100)
39 #pn532 = PN532_UART(uart, debug=False)
40
41 ic, ver, rev, support = pn532.get_firmware_version()
42 print('Found PN532 with firmware version: {0}.{1}'.format(ver, rev))
43
44 # Configure PN532 to communicate with MiFare cards
45 pn532.SAM_configuration()
46
47 print('Waiting for RFID/NFC card...')
48 while True:
49     # Check if a card is available to read
50     uid = pn532.read_passive_target(timeout=0.5)
51     print('.', end="")
52     # Try again if no card is available.
53     if uid is None:
54         continue
55     print('Found card with UID:', [hex(i) for i in uid])

```

5.2 adafruit_pn532

This module will let you communicate with a PN532 RFID/NFC shield or breakout using I2C, SPI or UART.

- Author(s): Original Raspberry Pi code by Tony DiCola, CircuitPython by ladyada

5.2.1 Implementation Notes

Hardware:

- Adafruit PN532 Breakout
- Adafruit PN532 Shield

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

exception `adafruit_pn532.adafruit_pn532.BusyError`
Base class for exceptions in this module.

class `adafruit_pn532.adafruit_pn532.PN532(*, debug=False, reset=None)`
PN532 driver base, must be extended for I2C/SPI/UART interfacing

SAM_configuration()
Configure the PN532 to read MiFare cards.

call_function (*command*, *response_length=0*, *params=[]*, *timeout=1*)

Send specified command to the PN532 and expect up to *response_length* bytes back in a response. Note that less than the expected bytes might be returned! Params can optionally specify an array of bytes to send as parameters to the function call. Will wait up to *timeout* seconds for a response and return a bytearray of response bytes, or None if no response is available within the timeout.

get_firmware_version ()

Call PN532 GetFirmwareVersion function and return a tuple with the IC, Ver, Rev, and Support values.

mifare_classic_authenticate_block (*uid*, *block_number*, *key_number*, *key*)

Authenticate specified block number for a MiFare classic card. *Uid* should be a byte array with the UID of the card, *block number* should be the block to authenticate, *key number* should be the key type (like MIFARE_CMD_AUTH_A or MIFARE_CMD_AUTH_B), and *key* should be a byte array with the key data. Returns True if the block was authenticated, or False if not authenticated.

mifare_classic_read_block (*block_number*)

Read a block of data from the card. *Block number* should be the block to read. If the block is successfully read a bytearray of length 16 with data starting at the specified block will be returned. If the block is not read then None will be returned.

mifare_classic_write_block (*block_number*, *data*)

Write a block of data to the card. *Block number* should be the block to write and *data* should be a byte array of length 16 with the data to write. If the data is successfully written then True is returned, otherwise False is returned.

ntag2xx_read_block (*block_number*)

Read a block of data from the card. *Block number* should be the block to read. If the block is successfully read a bytearray of length 16 with data starting at the specified block will be returned. If the block is not read then None will be returned.

ntag2xx_write_block (*block_number*, *data*)

Write a block of data to the card. *Block number* should be the block to write and *data* should be a byte array of length 4 with the data to write. If the data is successfully written then True is returned, otherwise False is returned.

read_passive_target (*card_baud=<sphinx.ext.autodoc.importer._MockObject object>*, *timeout=1*)

Wait for a MiFare card to be available and return its UID when found. Will wait up to *timeout* seconds and return None if no card is found, otherwise a bytearray with the UID of the found card is returned.

5.3 adafruit_pn532.i2c

This module will let you communicate with a PN532 RFID/NFC shield or breakout using I2C.

- **Author(s):** Original Raspberry Pi code by Tony DiCola, CircuitPython by ladyada, refactor by Carter Nelson

class `adafruit_pn532.i2c.PN532_I2C` (*i2c*, *, *irq=None*, *reset=None*, *req=None*, *debug=False*)

Driver for the PN532 connected over I2C.

5.4 adafruit_pn532.spi

This module will let you communicate with a PN532 RFID/NFC shield or breakout using SPI.

- **Author(s):** Original Raspberry Pi code by Tony DiCola, CircuitPython by ladyada, refactor by Carter Nelson

class adafruit_pn532.spi.**PN532_SPI** (*spi, cs_pin, *, irq=None, reset=None, debug=False*)

Driver for the PN532 connected over SPI. Pass in a hardware or bitbang SPI device & chip select digitalInOut pin. Optional IRQ pin (not used), reset pin and debugging output.

adafruit_pn532.spi.**reverse_bit** (*num*)

Turn an LSB byte to an MSB byte, and vice versa. Used for SPI as it is LSB for the PN532, but 99% of SPI implementations are MSB only!

5.5 adafruit_pn532.uart

This module will let you communicate with a PN532 RFID/NFC shield or breakout using UART.

- **Author(s): Original Raspberry Pi code by Tony DiCola, CircuitPython by ladyada,** refactor by Carter Nelson

class adafruit_pn532.uart.**PN532_UART** (*uart, *, irq=None, reset=None, debug=False*)

Driver for the PN532 connected over Serial UART

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`adafruit_pn532.adafruit_pn532`, [12](#)

`adafruit_pn532.i2c`, [13](#)

`adafruit_pn532.spi`, [13](#)

`adafruit_pn532.uart`, [14](#)

A

`adafruit_pn532.adafruit_pn532` (module), 12
`adafruit_pn532.i2c` (module), 13
`adafruit_pn532.spi` (module), 13
`adafruit_pn532.uart` (module), 14

B

`BusyError`, 12

C

`call_function()` (`adafruit_pn532.adafruit_pn532.PN532`
method), 13

G

`get_firmware_version()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13

M

`mifare_classic_authenticate_block()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13
`mifare_classic_read_block()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13
`mifare_classic_write_block()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13

N

`ntag2xx_read_block()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13
`ntag2xx_write_block()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13

P

`PN532` (class in `adafruit_pn532.adafruit_pn532`), 12

`PN532_I2C` (class in `adafruit_pn532.i2c`), 13
`PN532_SPI` (class in `adafruit_pn532.spi`), 13
`PN532_UART` (class in `adafruit_pn532.uart`), 14

R

`read_passive_target()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 13
`reverse_bit()` (in module `adafruit_pn532.spi`), 14

S

`SAM_configuration()`
(`adafruit_pn532.adafruit_pn532.PN532`
method), 12