
AdafruitPN532 Library Documentation

Release 1.0

ladyada

Jan 22, 2021

Contents

1	Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
3	Contributing	7
4	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	14
5.4	adafruit_pn532.spi	14
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.

CHAPTER 4

Documentation

For information on building library documentation, please check out [this guide](#).

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/pn532_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This example shows connecting to the PN532 with I2C (requires clock
6  stretching support), SPI, or UART. SPI is best, it uses the most pins but
7  is the most reliable and universally supported.
8  After initialization, try waving various 13.56MHz RFID cards over it!
9  """
10
11 import board
12 import busio
13 from digitalio import DigitalInOut
14
15 #
16 # NOTE: pick the import that matches the interface being used
17 #
18 from adafruit_pn532.i2c import PN532_I2C
19
20 # from adafruit_pn532.spi import PN532_SPI
21 # from adafruit_pn532.uart import PN532_UART
22
23 # I2C connection:
24 i2c = busio.I2C(board.SCL, board.SDA)
25
26 # Non-hardware
27 pn532 = PN532_I2C(i2c, debug=False)
```

(continues on next page)

(continued from previous page)

```

28
29 # With I2C, we recommend connecting RSTPD_N (reset) to a digital pin for manual
30 # hardware reset
31 reset_pin = DigitalInOut(board.D6)
32 # On Raspberry Pi, you must also connect a pin to P32 "H_Request" for hardware
33 # wakeup! this means we don't need to do the I2C clock-stretch thing
34 req_pin = DigitalInOut(board.D12)
35 pn532 = PN532_I2C(i2c, debug=False, reset=reset_pin, req=req_pin)
36
37 # SPI connection:
38 # spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
39 # cs_pin = DigitalInOut(board.D5)
40 # pn532 = PN532_SPI(spi, cs_pin, debug=False)
41
42 # UART connection
43 # uart = busio.UART(board.TX, board.RX, baudrate=115200, timeout=100)
44 # pn532 = PN532_UART(uart, debug=False)
45
46 ic, ver, rev, support = pn532.firmware_version
47 print("Found PN532 with firmware version: {0}.{1}".format(ver, rev))
48
49 # Configure PN532 to communicate with MiFare cards
50 pn532.SAM_configuration()
51
52 print("Waiting for RFID/NFC card...")
53 while True:
54     # Check if a card is available to read
55     uid = pn532.read_passive_target(timeout=0.5)
56     print(".", end="")
57     # Try again if no card is available.
58     if uid is None:
59         continue
60     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, irq=None, 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.

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

get_passive_target (timeout=1)
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. *listen_for_passive_target* must have been called first in order to put the PN532 into a listening mode.

It can be useful to use this when using the IRQ pin. Use the IRQ pin to detect when a card is present and then call this function to read the card's UID. This reduces the amount of time spend checking for a card.

listen_for_passive_target (card_baud=<sphinx.ext.autodoc.importer._MockObject object>, timeout=1)
Send command to PN532 to begin listening for a Mifare card. This returns True if the command was received successfully. Note, this does not also return the UID of a card! *get_passive_target* must be called to read the UID when a card is found. If just looking to see if a card is currently present use *read_passive_target* instead.

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.

power_down ()
Put the PN532 into a low power state. If the reset pin is connected a hard power down is performed, if not, a soft power down is performed instead. Returns True if the PN532 was powered down successfully or False if not.

process_response (*command*, *response_length=0*, *timeout=1*)

Process the response from the PN532 and expect up to *response_length* bytes back in a response. Note that less than the expected bytes might be returned! 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.

read_passive_target (*card_baud=<sphinx.ext.autodoc.importer._MockObject object>*, *time-*
out=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.

reset ()

Perform a hardware reset toggle and then wake up the PN532

send_command (*command*, *params=[]*, *timeout=1*)

Send specified command to the PN532 and wait for an acknowledgment. Will wait up to *timeout* seconds for the acknowledgment and return True. If no acknowledgment is received, False 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*, *, *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`, [14](#)
`adafruit_pn532.spi`, [14](#)
`adafruit_pn532.uart`, [14](#)

A

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

B

`BusyError`, 12

C

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

F

`firmware_version` (`adafruit_pn532.adafruit_pn532.PN532` attribute), 13

G

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

L

`listen_for_passive_target()`
(`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`), 14
`PN532_SPI` (class in `adafruit_pn532.spi`), 14
`PN532_UART` (class in `adafruit_pn532.uart`), 14
`power_down()` (`adafruit_pn532.adafruit_pn532.PN532` method), 13
`process_response()`
(`adafruit_pn532.adafruit_pn532.PN532` method), 13

R

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

S

`SAM_configuration()`
(`adafruit_pn532.adafruit_pn532.PN532` method), 13
`send_command()` (`adafruit_pn532.adafruit_pn532.PN532` method), 14