PV532 NFC RFID Module

User Manual

Hardware V3.5 | Firmware 1.6 | Document revision 1.4 | 11 November 2026

The ELECHOUSE PV532 V3.5 is a 13.56 MHz NFC module using the FSVP532 controller, compatible with PN5321. It accepts 3.3 V to 5 V DC and provides I2C, SPI and HSU UART with 3.3 V logic. It retains V4 mechanical dimensions and the four-pin UART/I2C connection footprint.



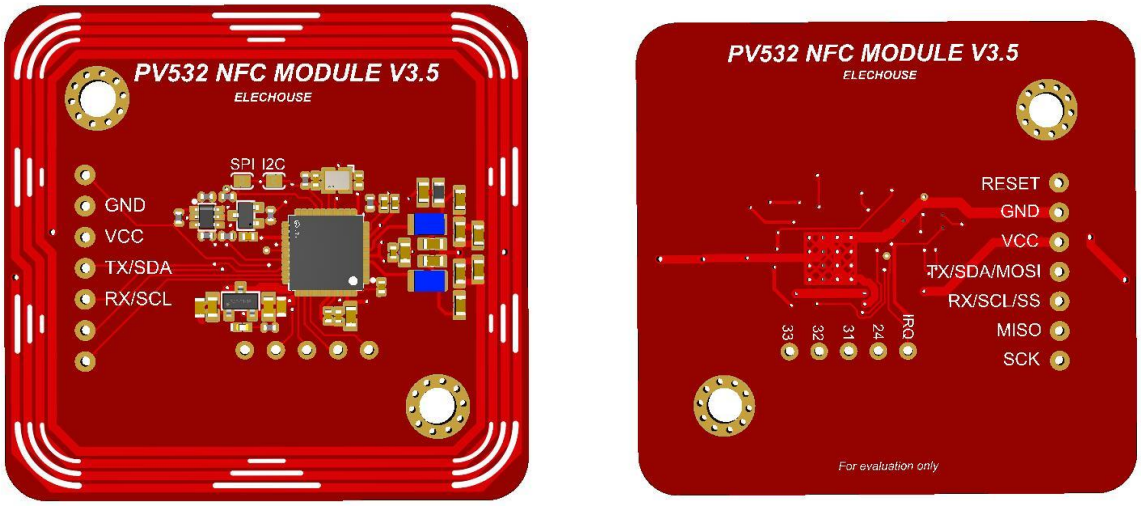
Actual PV532 V3.5 module showing the controller, mode jumpers, antenna and main connector.

Parameter	Specification
NFC controller	FSVP532, compatible with PN5321
Module firmware	Version 1.6
Operating frequency	13.56 MHz
Supply input at VCC	3.3 V to 5 V DC
Host signal level	3.3 V logic for I2C, SPI and UART
Host interfaces	I2C, SPI and HSU UART; select one at a time
Mode selection	Two solder jumpers marked I2C and SPI
Antenna	Integrated PCB antenna
Connector pitch	2.54 mm for H8 and H9
PCB outline	Approximately 42.7 x 40.4 mm

VCC supply voltage and communication logic voltage are separate requirements. Use 3.3 V signals even when VCC is supplied with 5 V.

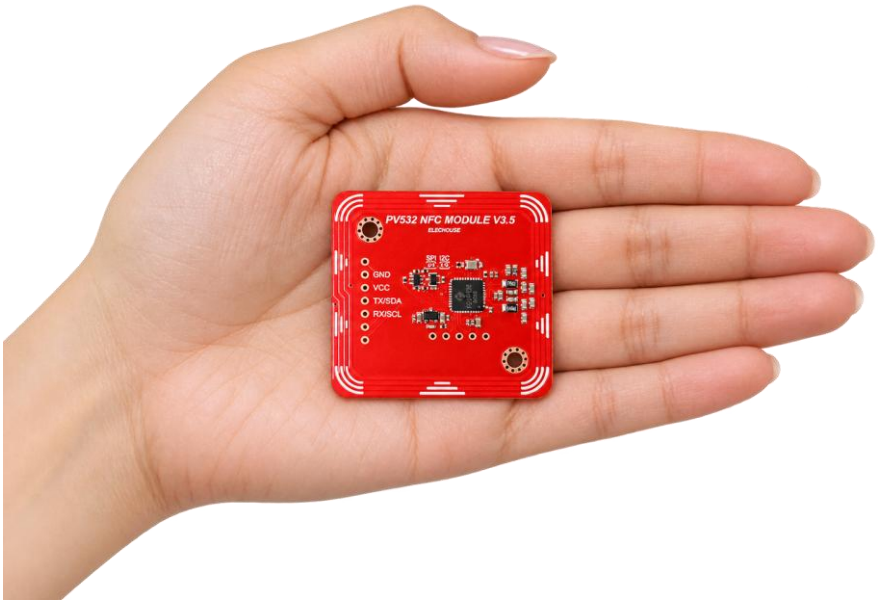
1 Product views and board identification

The component side carries the PV532 V3.5 product marking and the two mode jumpers. The reverse side identifies the full main header and the auxiliary signals. Use the view labels below when comparing pin positions.



Left: component-side PCB render. Right: reverse-side PCB render, viewed directly from the back.

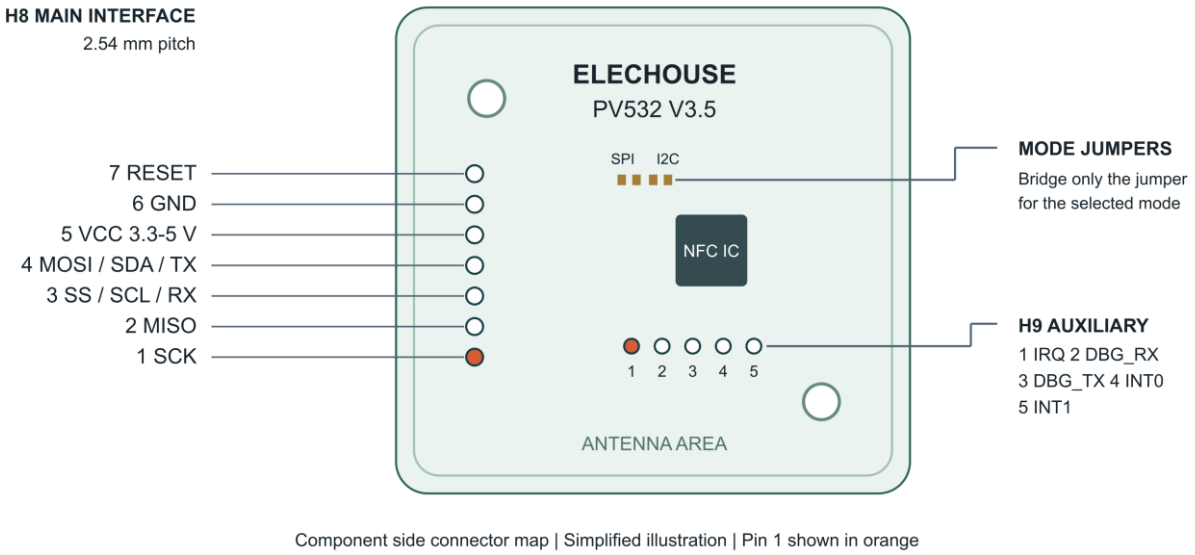
The reverse view is mirrored relative to the component-side connector map on the next page. The RX and TX names always refer to the module, not the host.



Actual PV532 V3.5 module in hand, shown for a sense of size. Use the dimension drawing for integration.

2 Main connector and signal definitions

This simplified component-side map corresponds to the actual front photograph on page 1. H8 pin 1 is SCK. The original V4 UART/I2C four-pin connection occupies H8 pins 3 through 6.



Component side connector map | Simplified illustration | Pin 1 shown in orange

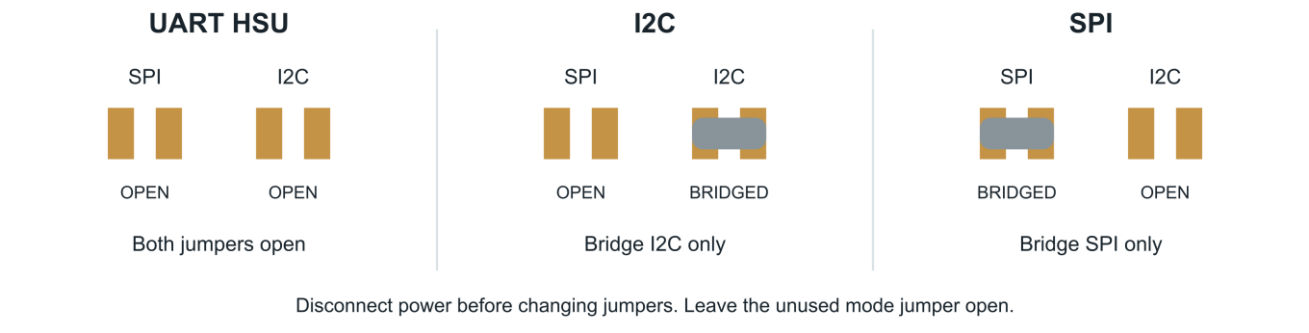
Component-side connector map. Orange marks pin 1; the board-side view must match the map when counting pins.

Pin	Signal	Function
1	SCK	SPI clock input
2	MISO	SPI data output to the host
3	SS / SCL / RX	SPI chip select / I2C clock / UART receive
4	MOSI / SDA / TX	SPI data input / I2C data / UART transmit
5	VCC	3.3 V to 5 V DC supply input
6	GND	Power and signal ground
7	RESET	RSTPD_N; active low reset and power down input

In UART mode, connect module RX to host TX and module TX to host RX. For basic operation, leave RESET unconnected. RESET is an active-low RSTPD_N input with an onboard pullup. If controlled by the host, use a 3.3 V compatible or open-drain output.

3 Select the communication mode

Locate the jumpers marked SPI and I2C on the actual module. Open means the two pads are separate; bridged means they are electrically joined with solder. Disconnect power before changing either jumper.



Mode diagram using the printed jumper names. For UART HSU, leave both jumpers open.

Selected mode	I2C jumper	SPI jumper
UART HSU	Open	Open
I2C	Bridged	Open
SPI	Open	Bridged
Do not use	Bridged	Bridged



Actual front photograph: the SPI and I2C jumper labels are above the controller area.

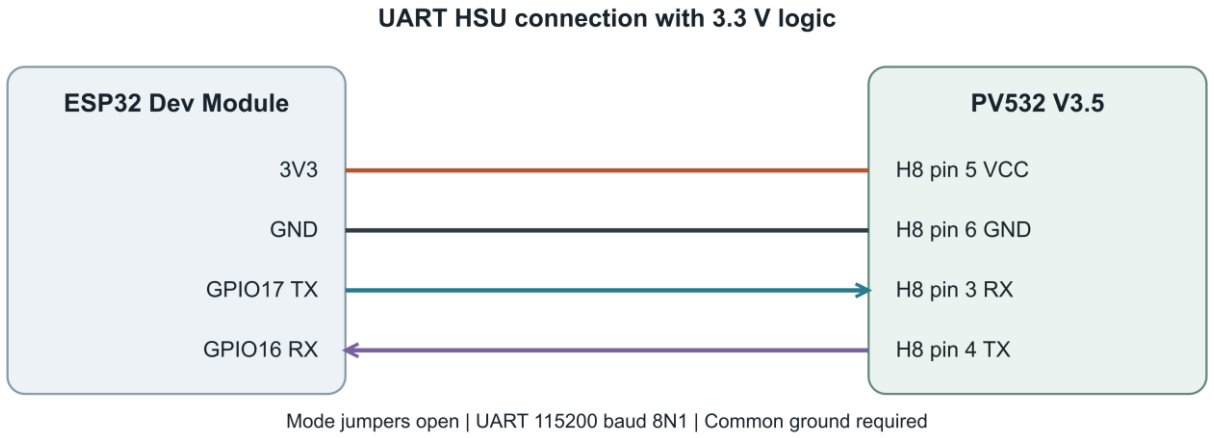
1. Set the jumpers for one interface only and inspect the solder joints.
2. Connect the corresponding host signals, then restore power. Power cycle after changing the mode.
3. Use PN532_HSU, PN532_I2C or PN532_SPI in software to match the selected mode.

UART HSU uses 115200 baud, 8 data bits, no parity and 1 stop bit. The I2C 7-bit address is 0x24 (8-bit write address 0x48). SPI uses mode 0 and least significant bit first.

4 Auxiliary signals and host wiring

H9 pin	Signal	Function
1	IRQ	Host interrupt signal; use if required by the driver
2	DBG_RX	Controller debug UART receive input
3	DBG_TX	Controller debug UART transmit output
4	INT0	Controller P32 external interrupt input
5	INT1	Controller P33 external interrupt input

DBG_RX and DBG_TX are the controller debug interface. Use H8 pins 3 and 4 for HSU host commands. Leave H9 unconnected for the supplied UID examples.



Classic ESP32 Dev Module wiring. Serial2 connects to PV532; Serial is used for the USB serial monitor.

H8 module pin	UART HSU	I2C	SPI
1 SCK	Unused	Unused	GPIO18
2 MISO	Unused	Unused	GPIO19
3 SS / SCL / RX	GPIO17 TX	GPIO22 SCL	GPIO5 SS
4 MOSI / SDA / TX	GPIO16 RX	GPIO21 SDA	GPIO23 MOSI
5 VCC	3V3	3V3	3V3
6 GND	GND	GND	GND
7 RESET	Leave open	Leave open	Leave open

Connect stable 3.3-5 V DC to VCC and share ground with the host. The table uses 3.3 V power. All communication signals, including any external I2C pullups, must stay in the 3.3 V logic domain. Add suitable level conversion for a 5 V host.

The examples use a classic ESP32 WROOM Dev Module. ESP32 C3, S3 and other variants have different pin availability; update pin assignments before using them. Keep leads short and away from the antenna.

5 Install the library and run UART

Extract the supplied Arduino library archive and copy PN532, PN532_HSU, PN532_I2C and PN532_SPI into the Arduino sketchbook libraries directory. Restart the Arduino tools. The archive includes the SPI header compatibility fix for ESP32 core 3.3.8. These library names are host API names; the product name is PV532.

Select ESP32 Dev Module, open the supplied HSU sketch and upload it. Leave both jumpers open and use the wiring on page 5. Open the serial monitor at 115200 baud. Module firmware is version 1.6; the example checks the firmware response and reads an ISO14443A UID.

```
#include <PN532_HSU.h>
#include <PN532.h>

PN532_HSU transport(Serial2);
PN532 nfc(transport);

void setup() {
  Serial.begin(115200);
  Serial2.setPins(16, 17);  // ESP32 RX, TX
  delay(100);
  nfc.begin();
  const uint32_t version = nfc.getFirmwareVersion();
  if (!version) {
    Serial.println("PV532 not found. Check power, mode and wiring.");
    while (true) delay(1000);
  }
  if (!nfc.SAMConfig()) {
    Serial.println("PV532 SAMConfig failed.");
    while (true) delay(1000);
  }
  nfc.setPassiveActivationRetries(0x02);
  Serial.println("PV532 ready. Present an ISO14443A tag.");
}

void loop() {
  uint8_t uid[10];
  uint8_t uidLength = 0;
  if (nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A,
                              uid, &uidLength, 1000)) {
    Serial.print("UID:");
    for (uint8_t i = 0; i < uidLength; ++i) {
      Serial.print(' ');
      if (uid[i] < 0x10) Serial.print('0');
      Serial.print(uid[i], HEX);
    }
    Serial.println();
    delay(500);
  }
}
```

6 Use the I2C and SPI examples

Three complete sketches are included with this manual. The I2C and SPI versions perform the same firmware check and UID reading as the UART example; their transport and bus initialization differ.

I2C initialization

Bridge I2C only, power cycle and connect SDA to GPIO21 and SCL to GPIO22. Use the I2C sketch from the package. Its transport setup is:

```
#include <Wire.h>
#include <PN532_I2C.h>
#include <PN532.h>

PN532_I2C transport(Wire);
PN532 nfc(transport);

// In setup(), before nfc.begin():
Wire.setPins(21, 22);  // SDA, SCL
```

SPI initialization

Bridge SPI only, power cycle and connect all four SPI signals as shown on page 5. Use the SPI sketch from the package. Its transport setup is:

```
#include <SPI.h>
#include <PN532_SPI.h>
#include <PN532.h>

PN532_SPI transport(SPI, 5);  // SS
PN532 nfc(transport);

// In setup(), before nfc.begin():
SPI.begin(18, 19, 23, 5);  // SCK, MISO, MOSI, SS
```

Example files

Sketch folder	Mode jumper state
PV532_V3_5_ESP32_HSU_UID	Both open
PV532_V3_5_ESP32_I2C_UID	I2C bridged, SPI open
PV532_V3_5_ESP32_SPI_UID	I2C open, SPI bridged

Expected operation

After initialization, the serial monitor displays "PV532 ready. Present an ISO14443A tag." Bring a supported tag close to the antenna. The UID appears as hexadecimal bytes. A tag held in place may be reported repeatedly; use application logic if one event per presentation is required.

The examples read UIDs only. They do not write card memory, format NDEF records or implement the other protocols listed in section 8.

7 ESP32 C3 UART card range test

The updated PN5321 GitHub project includes a standalone card-identification and manual range-test program for the LuatOS CORE-ESP32-C3. Keep the sketch and its Pn532Hsu.cpp / Pn532Hsu.h driver files together; no external NFC or NDEF library is needed.

Latest test software and guide
https://github.com/wilson-elechouse/PN5321/blob/arduino/extras/esp32c3-range-test/README.md

Documentation checked against commit bab0e97. Reference build: Arduino ESP32 core 3.3.8, ESP32C3 Dev Module, DIO, 40 MHz, 4 MB flash and USB CDC On Boot disabled. Use the matching sketch folder from GitHub or the supplied source snapshot.

Reference wiring and serial settings

PV532 V3.5	LuatOS CORE ESP32 C3
H8 pin 4 TX / SDA	GPIO4 / UART1 RX
H8 pin 3 RX / SCL	GPIO5 / UART1 TX
H8 pin 5 VCC	3.3 V supply
H8 pin 6 GND	GND

Leave both SPI and I2C jumpers open and power cycle the module. Both serial links use 115200 baud, 8N1. All module signals are 3.3 V. The reference host uses GPIO12 for D4 and GPIO13 for D5; adapt these LED pins for other boards. These GPIO4/5 connections differ from the classic ESP32 examples in sections 5 and 6.

Test commands and results

Serial command	Purpose
x / a / b / f / g / j / k	AUTO / Type A / Type B / FeliCa 212 / FeliCa 424 / Jewel / ST SR-family entry
u	Use ID_ONLY as the distance criterion
c / t / v	Classic block 4 read / Type 2 pages 4-7 read / GET_VERSION; separate from ID-only testing
n <sample> / d <mm>	Set a sample label / manually measured distance in millimetres
r / s / h	Clear and reinitialize / statistics / help

Send one command per line. For an NTAG215 sample, send a, then n NTAG215_01, then u, then d 30. Fix the protocol before comparing distances. Measure the antenna-to-card spacing with a ruler; the program does not measure distance automatically.

Read WINDOW20 pass / no_card / errors and STATS, and save RESULT CSV lines. Record card shape, dimensions, orientation, supply and surroundings. AUTO plus ID_ONLY is the default; optional reads do not write card data. The repository records live Type A validation; other modes, including ST SR, require matching real-card checks. See the GitHub guide for the current validation scope.

8 Tags and NFC functions

The PV532 module uses FSVP532, compatible with PN5321 with firmware 1.6. Its protocol family includes ISO14443A/MIFARE, ISO14443B, FeliCa and ISO18092 NFCIP-1. Application software determines which commands and tag operations are available.

Card or function	Protocol	Application notes
MIFARE Classic 1K and 4K	ISO14443A	UID reading; data access requires the appropriate authentication key
MIFARE Ultralight and NTAG21x	ISO14443A	UID and tag data; NDEF needs a suitable software library
MIFARE DESFire	ISO14443A and ISO14443-4	Protocol communication requires application commands and authentication support
ISO14443B cards	ISO14443B	Additional host software is needed; this is not covered by the UID examples
FeliCa	FeliCa	Requires software that implements the relevant commands
NFC peer to peer	ISO18092 NFCIP-1	Chip capability; depends on host software and the peer device
Card emulation	ISO14443A and FeliCa modes	Requires a suitable target mode implementation

Reading and writing data

Start with an ISO14443A UID example to verify the connection. MIFARE Classic data access requires the correct sector key. NDEF operations on NTAG or Ultralight need a compatible NDEF library. DESFire commands, authentication, card emulation and peer-to-peer workflows require suitable application software.

Firmware and software compatibility

Firmware 1.6 is the module firmware version. The examples use the ELECHOUSE PN532 host library API; match the transport to the jumpers and confirm the firmware response.

Measured reading distance

PV532 V3.5 reading distances at 3.3 V and 5 V supply are nearly identical. One reference result is shown for each tested card or tag.

Card type and shape	Measured reading distance (mm)
S70 - Rectangular (54 x 85.5 mm)	75
M1 - Circular (diameter 25 mm)	40
ISO14443B - Rectangular (54 x 85.5 mm)	45
NTAG215 - Rectangular (16 x 6 mm)	20
NTAG215 - Circular (diameter 25 mm)	45

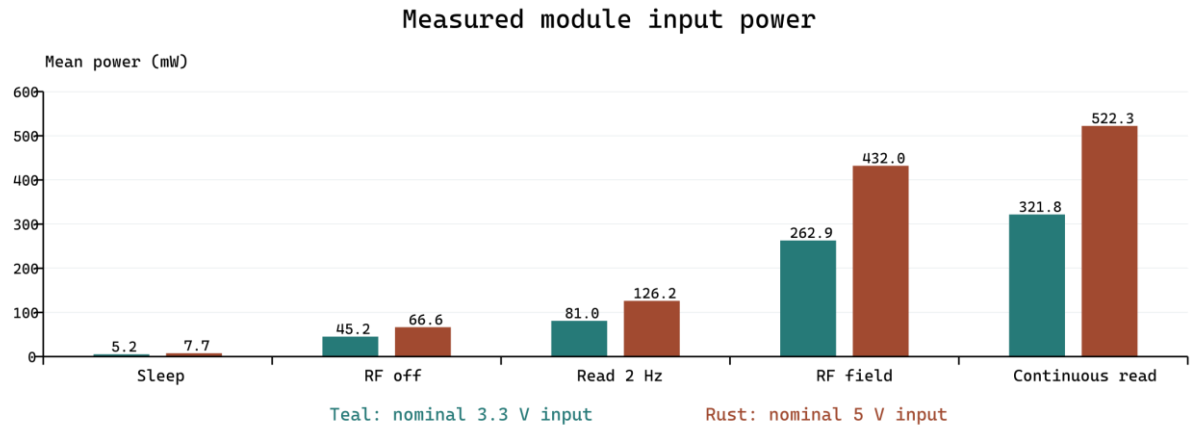
Measured reference values, not guaranteed limits. Actual distance depends on the tag antenna, orientation, enclosure and electrical environment.

Antenna placement

Place the tag over the antenna with the two antenna planes parallel. Keep large metal parts and cable bundles away from the antenna perimeter. Verify reading distance with the intended tags in the final installation.

9 Measured module power consumption

Measured on 11 November 2026 with a P5S1 v2.2 at 100 Hz. The measurement covers the complete module input, including onboard regulator and indicator loads. The ESP32-C3 host is excluded. HSU uses 115200 baud and 3.3 V logic; an S70 Classic 4K card remains on the antenna.



Mean input power for the tested operating modes. Supply labels are nominal; measured terminal voltages appear below.

Nominal 3V3 supply

Operating condition	Mean input V	Mean mA	Mean mW
PowerDown sleep	3.314	1.57	5.21
Ready with RF off	3.269	13.82	45.19
S70 read at 2 Hz	3.263	25.33	80.97
Continuous RF field	3.204	82.06	262.92
Continuous S70 read	3.169	101.56	321.80
Return to sleep	3.318	1.53	5.07

Nominal 5 V supply

Operating condition	Mean input V	Mean mA	Mean mW
PowerDown sleep	5.079	1.51	7.66
Ready with RF off	5.059	13.16	66.56
S70 read at 2 Hz	5.039	25.27	126.18
Continuous RF field	4.933	87.57	432.03
Continuous S70 read	4.901	106.57	522.25
Return to sleep	5.080	1.51	7.68

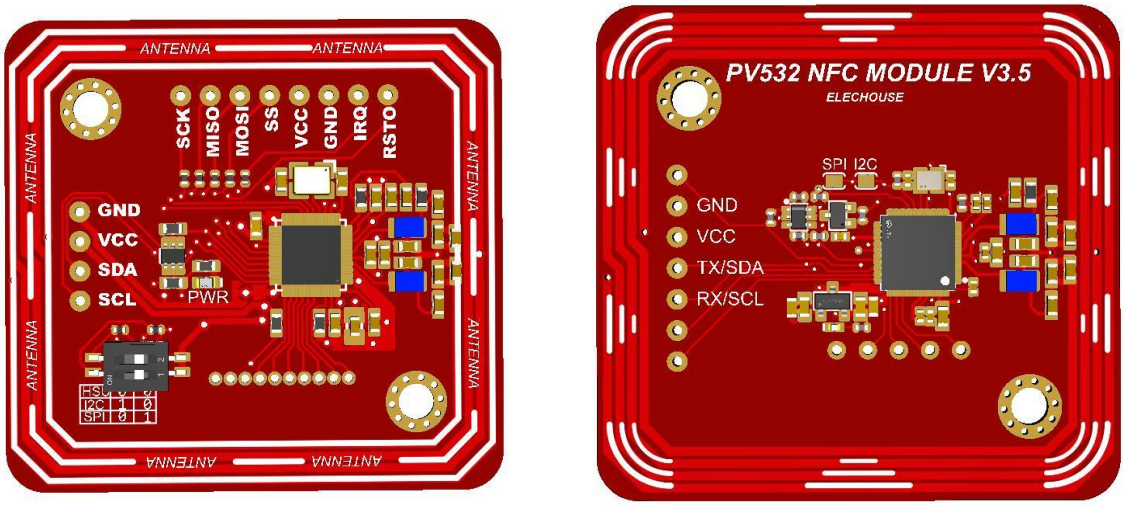
At 2 Hz, each 500 ms cycle selects and authenticates the card, reads block 4, then switches RF off. Continuous read authenticates once and repeatedly reads block 4. PowerDown is software sleep with HSU wakeup enabled; hardware reset power-down was not measured.

The 3.3 V path drooped under load: full-run voltage was 2.952-3.324 V and continuous-read mean voltage was 3.169 V. These results are not measurements at a constant 3.300 V module terminal. Sleep current was approximately 1.53-1.57 mA at nominal 3.3 V and 1.51 mA at nominal 5 V.

The 5 V capture had three estimated missing 100 Hz samples; the 3.3 V capture had none. Both had zero CRC errors. Sampled peaks do not bound transients shorter than 10 ms. Results describe the tested loads, not an absolute maximum rating. Full distribution statistics are in the supplied power data CSV.

10 Compatibility with the V4 module

PV532 V3.5 is fully compatible with V4 in board size, PCB thickness, mounting-hole positions and antenna position. The main four-pin UART/I2C interface retains its physical location and signal order, so the existing mechanical installation and four-wire connector footprint can be reused.



Left: V4 reference PCB render. Right: PV532 V3.5 component-side PCB render. Both show the same component-side orientation.

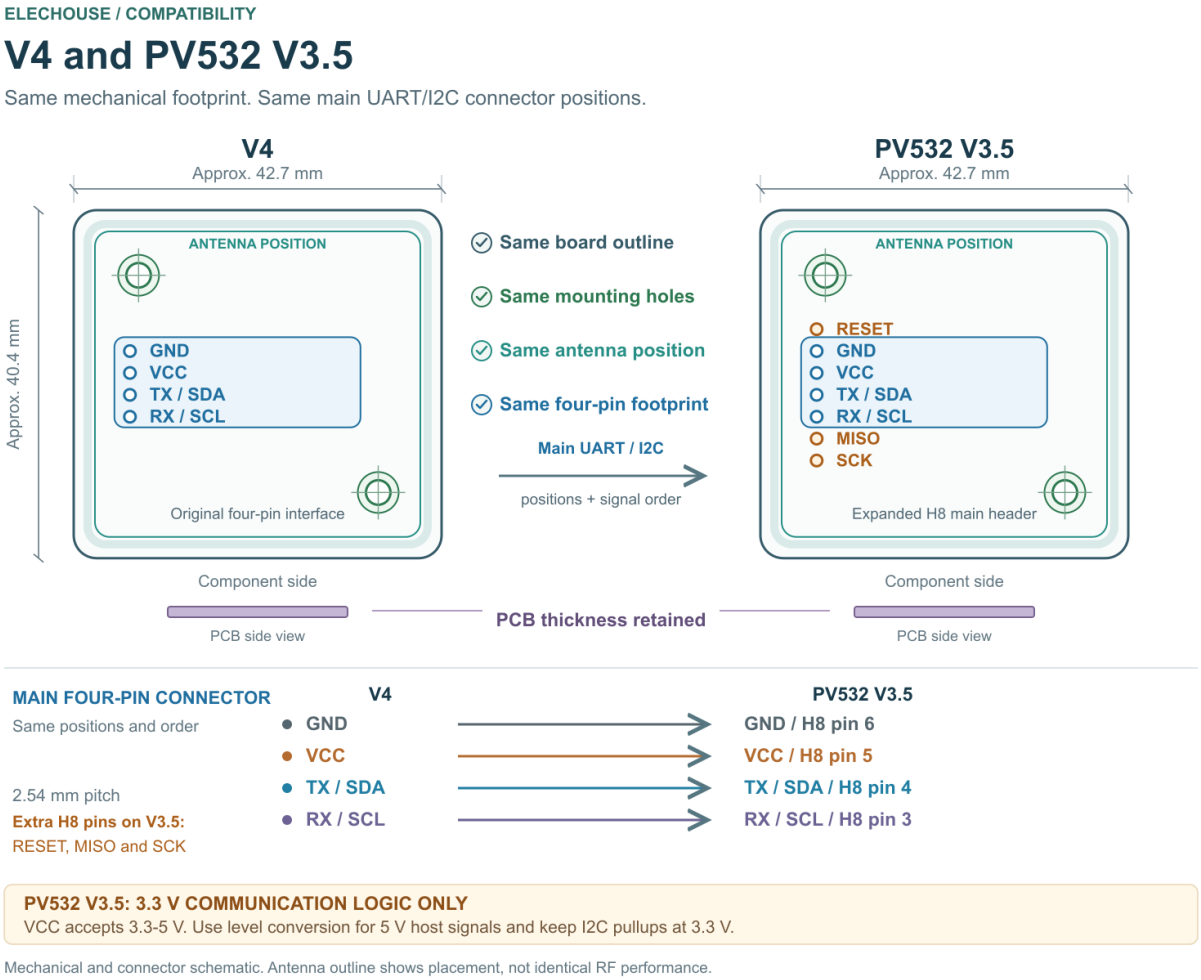
Compatibility item	PV532 V3.5
Board outline	Fully compatible with V4 board size and shape
PCB thickness	Compatible with V4 PCB thickness
Mounting holes	Same mounting-hole positions
Antenna position	Same antenna placement within the board footprint
UART and I2C main interface	Same four-pin location and signal order; H8 pins 3-6 on PV532
Signal level	PV532 communication signals use 3.3 V logic
SPI and auxiliary headers	Use the PV532 pin definitions; these headers have been rearranged

Four-wire sequence from the component-side view, top to bottom: GND, VCC, TX/SDA, RX/SCL. UART TX and RX are named from the module perspective.

Mechanical and connector compatibility does not change the 3.3 V signal requirement. Review any 5 V host signals or I2C pullups before replacing a V4 module. Antenna position compatibility does not specify an identical read distance.

11 V4 compatibility diagram

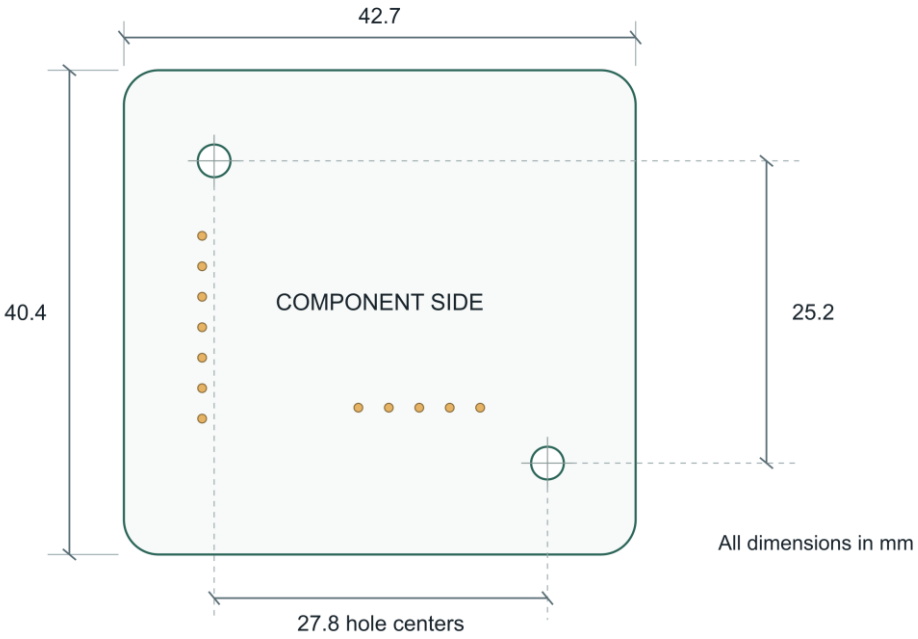
The shared footprint, antenna position and main UART/I2C connector positions are shown below. Orange marks additional pins on the PV532 V3.5 main header.



Component-side schematic. Mechanical compatibility does not change the 3.3 V communication logic requirement.

Use section 12 for the dimension reference and sections 2-4 for the complete pin definitions and host wiring.

12 Mechanical reference and installation



Nominal PCB outline approximately 42.7 x 40.4 mm. Connector pitch 2.54 mm.

Reference dimensions for integration. Confirm connector height and enclosure clearances on the supplied module.

Nominal dimensions from the board layout. Refer to the actual module photos for orientation and the V4 comparison on page 11.

Item	Reference
Nominal PCB outline	Approximately 42.7 x 40.4 mm
Mounting-hole center separation	27.8 mm horizontally and 25.2 mm vertically
Main and auxiliary pin pitch	2.54 mm
PCB thickness	Compatible with V4; use the supplied board for final stack-up checks
Antenna position	Compatible with V4 antenna placement

Installation

Retain the V4 mounting and antenna position when reusing an existing enclosure. Check the assembled connector height and cable access. Keep conductive hardware and large metal surfaces away from the antenna perimeter, and avoid laying wires across the antenna traces.

Use the reverse-side labels to confirm SCK, MISO, SS, MOSI and RESET when wiring SPI. The reverse-side photo view is mirrored relative to the component-side connection map.

13 Troubleshooting and resources

Symptom	Likely cause	Action
No firmware response	Wrong mode, crossed wiring, or missing ground	Match the jumper state and transport class. Recheck H8 pin numbers and common ground.
UART gives no response	TX and RX are not crossed	Host TX goes to H8 pin 3 RX. Host RX goes to H8 pin 4 TX.
Debug pins do not accept commands	H9 was used for the host connection	Use H8 pins 3 and 4 for HSU; DBG_RX and DBG_TX serve the chip debug interface.
I2C address is not found	Wrong mode or bus wiring	Bridge I2C only, power cycle, and check SDA and SCL. Use 7 bit address 0x24.
SPI gives no response	Wrong chip select or SPI settings	Check all four signals, SPI mode 0 and least significant bit first. Use the supplied SPI example.
Firmware responds but no tag appears	Unsupported tag or weak coupling	Start with an ISO14443A tag. Center it over the antenna and keep the antenna planes parallel.
Detection becomes unstable near an enclosure	Metal, long wiring or supply noise	Retest away from metal with short leads and a stable supply. Recheck the final installation.
An example does not compile	Wrong library or board selection	Use the ELECHOUSE library folders and classic ESP32 Dev Module target. Remove conflicting PN532 library copies.

Downloads and references

PN5321 test software on GitHub
https://github.com/wilson-elechouse/PN5321

ESP32 C3 range test guide
https://github.com/wilson-elechouse/PN5321/blob/arduino/extras/esp32c3-range-test/README.md

ELECHOUSE technical support
https://www.elechouse.com/contact/

The latest standalone test program is maintained in the PN5321 GitHub repository. The supplied classic ESP32 UID examples use a separate matching library bundle. This manual references range-test commit bab0e97; check GitHub for later updates. PN532 names in software refer to the compatible command API.

Document history

Rev	Date	Description
1.4	11 November 2026	Added the approved V4 compatibility schematic and the updated GitHub ESP32-C3 UART card range test
1.3	11 November 2026	Replaced the V3.5 photo in the V4 comparison with the supplied component-side PCB render
1.2	11 November 2026	Added measured V3.5 reading distances; combined card type and shape; noted nearly identical results at 3.3 V and 5 V
1.1	11 November 2026	PV532 branding, firmware 1.6, actual photos, V4 mechanical compatibility and measured power
1.0	10 November 2026	Initial interface and software instructions